

Numerical Approximation Methods and Comparison with RK-4 Method for a Linear Differential Equation with Initial Conditions Using Scilab 6.1.1

Ravindra Singh^{1*}, Omwati Rana², Yogesh Kumar Sharma³, Shiv Shankar Gaur¹

¹Department of Physics, Shivaji College (University of Delhi), Raja Garden New Delhi-110027.

²Department of Physics, Daulat Ram College (University of Delhi), Maurice Nagar, New Delhi-110007.

³Department of Chemistry, Kalindi College (University of Delhi), East Patel Nagar New Delhi-110008.

Volume 1, Issue 6, December 2024

Received: 2 October, 2024; Accepted: 16 November, 2024

DOI: <https://doi.org/10.63015/5c-2447.1.6>

*Corresponding Author Email: ravindrasingh@shivaji.du.ac.in

Abstract: Numerical approximation methods have been developed to linear differential equation with initial condition. The comparison of the results has been done among Euler's method, modified Euler's method and RK-4 with the help of Scilab software 6.1.1. RK-4 method is effective enough to reach more accuracy in the result. The Runge-Kutta method attempts to overcome the problem of the Euler's method, and modified Euler's method and the study shows that in all cases RK-4 method improves to a great extent, than those by the Euler method and Modified Euler's Method. For RK-4 the approximation accuracy is proportional to the fourth power of the step size, thus making it a powerful and widely used numerical method also this method gives us higher accuracy without performing more calculations. Three different values of the step size have been taken. It is observed that smaller values of step size give better result; in all cases RK-4 method fits best as compared to others. The nature of the plot obtained by directly matches with the approximation solutions. It is observed that RK-4 method is suitable for obtaining the accurate solution of ODEs when the taken step sizes are too much small; since smaller h reduces the error.

Keywords: Ordinary Differential Equation, Initial Conditions, Step size, Accuracy Analysis, Absolute Error and Scilab software 6.1.1.

1. Introduction: Scilab is a free and open-source, cross-platform with highly numerical calculation package. It has multiple uses and wide applications in all branch of sciences and engineering. Numerical methods are techniques by which mathematical problems are formulated so that they can be analyzed with arithmetic operations although there are many kinds of numerical methods, they have one common characteristic: they invariably involve large numbers of tedious arithmetic calculations. To solve differential equations we have many methods but in this paper we have opted only three method named Euler's method, modified Euler's method and RK-4 method. Operations with ODE having

significant role when it was done through scilab, the programming starts with the detailed modeling an ODE with three different styles [1]. In a comparison of performance of the CP scheme, Polygon, Harmonic-Polygon, and Cube-Polygon schemes it was found that all enhance Euler methods. The CP scheme achieves higher accuracy while requiring less computing time when applied to the RCL circuit equation for second-order ODE [2]. A System of non-homogeneous equations using matrix exponential method was studied and analyzed using Scilab software 6.1.1 [3]. All the results obtained by ODEs with initial conditions can be analyzed with the help of softwares and then the results

compared. These days Scilab software so popular, we used it in this paper (Scilab software 6.1.2 with window operating system). Result Analysis with accuracy at given initial conditions for ODEs done by M. A. Arefin et al. using modified Euler method [4]. A demonstration study done on RK-4 method by using MATLAB Programming [5]. The numerical solutions for Euler's method and RK4 for ordinary differential equations (ODEs) were solved using Scilab [6]. Multiple numerical solutions to intuitionistic fuzzy differential equations have been studied as well as the traffic flow problem using RK4 method and [7-9]. Many times the linear differential equations of second order with different ways and different methods had been compared by Runge-Kutta methods [10-12]. Numerical methods such as RK4 etc. are commonly used for solving mathematical problems that are formulated in science and engineering where it is difficult or even impossible to obtain exact solutions these methods use mathematical modelling forms an important part of many tertiary courses in mathematics and engineering. Numerical methods provide a vehicle for you to reinforce your understanding of mathematics. RK-4 methods is to reduce higher mathematics to basic arithmetic operations it is a rich collection of numerical algorithms covering many aspects of scientific computing problems such as ordinary differential equations which can solve by Matlab also [13]. Non-linear partial differential equations such as a specific fluid flow problem can be easily solved by RK-4 method as well as it is capable solve a complex problem, physically or geometrically. The Various methods such as RK-4 and numerical techniques the basic fundamentals for the differential equations as well as mathematical modelling and engineering problem done by S. C. Chapra, R. P. Canale and D. Houcque [14-16]. The short list of its capabilities for matrices, polynomials, linear equation, signal processing, ODE's for numerical methods always involve with the topics related to

generate the matrices which are solving the linear equation, ordinary differential equation (ODE's) and numerical integration [17-18].

IMPORTANCE & APPLICATIONS OF RK-4 METHOD

RK-4 METHOD: Runge Kutta was not a single person but two names collectively and the Runge Kutta method was named after these two German mathematicians-Carl David Tolmé Runge and Wilhelm Kutta. Runge Kutta method is an iterative method, where the successive approximations are generated to reach the final result and each iteration requires function evaluations at several points within the step interval. This method can be used for 'n' iterations to solve the differential equation; in this method the approximation accuracy is proportional to the fourth power of the step size, thus making it a powerful and widely used numerical method.

RK-4 is called **fourth-order Runge-Kutta method** for approximating the solution to the problem with initial value $y' = f(x, y)$ with initial condition $y(x_0) = y_0$ at the points

$x_{n+1} = x_0 + nh$ in an interval with slope as an arithmetic average

where $n = 0, 1, 2, 3 \dots$

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

where

$$k_1 = hf(x_0, y_0)$$

$$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_0 + h, y_0 + k)$$

Applications:

1. It controls the accuracy and adjustment of the step size.
2. It suited very well to solve linear and non-linear initial value fuzzy problems.
3. It can be used to solve a complex problem, physically or geometrically.

4. It us higher accuracy without performing more calculations.
5. It is also used to solve both higher order ordinary differential equations and coupled (simultaneous) differential equations.
6. It can be used to solve non-linear partial differential equations such as a specific fluid flow problem.
7. It gives more accurate result as compared to above two methods. The solution curves were infact generated using a Runge-Kutta approximation.
8. We can take different value of step size, h.

SCILAB PROGRAMMING/CODING FOR EULER'S METHOD:

```
funcprot(0);
function dy=f(x, y)
dy=y-x;
endfunction
C=1;
x0=0; //initial value of x
y0=1/2; //initial value of y
h=0.1; //step size It can be 0.01;0.05 etc
xn=1;
n=(xn-x0)/h; //no. of steps
x=zeros(n+1,1)
y=zeros(n+1,1)
x(1)=x0;
y(1)=y0;
disp('x y Exact Soln Abs Error');
for i=1:n
    x(i+1)=x(i)+h;
    y(i+1)=y(i)+h*f(x(i),y(i))
end
yexact=C+x-(1/2)*exp(x); //it is an exact
soln after solving DE manually
plot(x,y,'r',x,yexact,'-b','linewidth',2)
//plots
xlabel('x→','fontsize',6)
ylabel('y↑','fontsize',6,'rot',0)
title('h=0.1','fontsize',4)
legend('Euler Solution','Exact Solution',2)
error=yexact-y;
a=yexact-y;
abs(a);
disp([x,y,yexact,abs(a)])
```

SCILAB PROGRAMMING/CODING FOR EULER'S MODIFIED METHOD:

```
funcprot(0);
function dy=f(x, y)
dy=y-x;
endfunction
x0=0;//initial value of x
y0=1/2;//initial value of y
h=0.1;//step size It can be 0.01;0.05 etc
xn=1;
n=(xn-x0)/h;
x=zeros(n+1,1)
y=zeros(n+1,1)
x(1)=x0;
y(1)=y0;
disp('x y Exact Soln Abs Error');
for i=1:n
    x(i+1)=x(i)+h;
    yc(i+1)=y(i)+h*f(x(i),y(i))
    y(i+1)=y(i)+0.5*h*(f(x(i),y(i))+f(x(i+1),yc(
    i+1)))
end
yexact=1+x-(1/2)*exp(x);//it is the exact
soln after solving DE manually
plot(x,y,'r',x,yexact,'-b','linewidth',2)
xlabel('x→','fontsize',6)
ylabel('y↑','fontsize',6,'rot',0)
title('h=0.05','fontsize',4)
legend('Eulers Modified','Exact Solution',2)
error=yexact-y;
a=yexact-y;
abs(a);
disp([x,y,yexact,abs(a)])
```

SCILAB PROGRAMMING/CODING FOR RK-4 METHOD:

```
funcprot(0);
function dy=f(x, y)
dy=y-x;
endfunction
x0=0;
y0=1/2;
h=0.1; //step size It can be 0.01;0.05 etc
n=(xn-x0)/h; //no. of steps
x=zeros(n+1,1)
y=zeros(n+1,1)
x(1)=x0;
y(1)=y0;
```

```

disp('x y Exact Soln Abs Error');
for i=1:n;
x(i+1)=x(i)+h;
k1=h*f(x(i),y(i));
k2=h*f(x(i)+h/2,y(i)+k1/2);
k3=h*f(x(i)+h/2,y(i)+k2/2);
k4=h*f(x(i)+h,y(i)+k3);
y(i+1)=y(i)+(1/6)*(k1+2*k2+2*k3+k4);
end
yexact=(1+x)-(1/2)*exp(x);%it is the exact
soln after solving DE manually
plot(x,y,'r',x,yexact,'-b','linewidth',2)
xlabel('x→','fontsize',6)
ylabel('y↑','fontsize',6,'rot',0)
title('h=0.05','fontsize',4)
legend('RK4','Exact Solution',2)
error=yexact-y;
a=yexact-y;
abs(a);
disp([x,y,yexact,abs(a)])

```

Ordinary method:

```

clc
clf
funcprot(0);
function dy=f(x, y)
    dy = y-x
endfunction
x0 = 0;
y0 = 1/2;
x=0:0.01:1;
y=ode(y0,x0,x,f)
plot(x,y,'linewidth',2)
xlabel('x→','fontsize',6)
ylabel('y↑','fontsize',6,'rot',0)

```

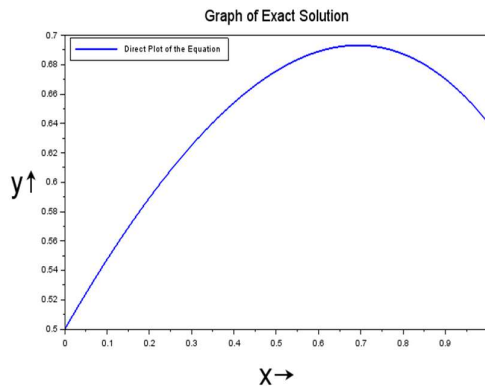


Figure 1. Solution by direct method

title('Graph of Exact Solution','fontsize',4)
legend('Direct Plot of the Equation',2)

Now continuing with a systematic manner, we generate the approximations listed in Tables 1-3 where we have rounded the calculations to seven decimal places.

Table 1. [Figure 2(a)-2(c)] Approximate solution for different step sizes with exact solution and absolute value.

EULER			
h=0.1			
x	y	Exact Solution	Abs Error
0.	0.5	0.5	0.
0.1	0.55	0.5474145	0.0025855
0.2	0.595	0.5892986	0.0057014
0.3	0.6345	0.6250706	0.0094294
0.4	0.66795	0.6540877	0.0138623
0.5	0.694745	0.6756394	0.0191056
0.6	0.7142195	0.6889406	0.0252789
0.7	0.7256415	0.6931236	0.0325178
0.8	0.7282056	0.6872295	0.0409761
0.9	0.7210262	0.6701984	0.0508277
1.	0.7031288	0.6408591	0.0622697
Euler's Modified			
0.	0.5	0.5	0.
0.1	0.5475	0.5474145	0.0000855
0.2	0.5894875	0.5892986	0.0001889
0.3	0.6253837	0.6250706	0.0003131
0.4	0.654549	0.6540877	0.0004613
0.5	0.6762766	0.6756394	0.0006373
0.6	0.6897857	0.6889406	0.0008451
0.7	0.6942132	0.6931236	0.0010895
0.8	0.6886055	0.6872295	0.001376
0.9	0.6719091	0.6701984	0.0017107
1.	0.6429596	0.6408591	0.0021005
RK-4			
0.	0.5	0.5	0.
0.1	0.5474146	0.5474145	4.237D-08
0.2	0.5892987	0.5892986	9.365D-08
0.3	0.6250708	0.6250706	0.0000002
0.4	0.6540879	0.6540877	0.0000002
0.5	0.6756397	0.6756394	0.0000003
0.6	0.688941	0.6889406	0.0000004
0.7	0.6931242	0.6931236	0.0000005
0.8	0.6872302	0.6872295	0.0000007
0.9	0.6701993	0.6701984	0.0000008
1.	0.6408601	0.6408591	0.000001

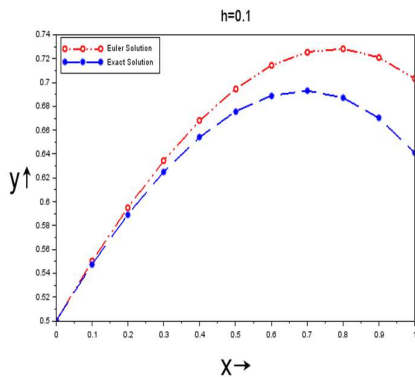


Figure 2(a). Approximate curve for step size $h=0.1$

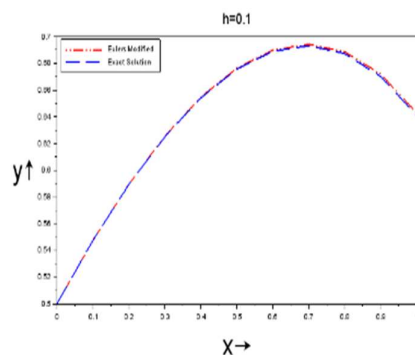


Figure 2(b). Approximate curve for step size $h=0.1$

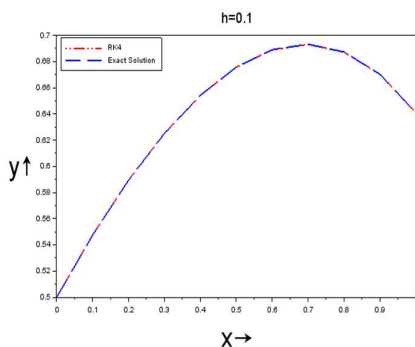


Figure 2(c). Approximate curve for step size $h=0.1$

Table 2. [Figure 3(a)-3(c)] Approximate solution for different step sizes with exact solution and absolute value

EULER			
h=0.05			
x	y	Exact Solution	Abs Error
0.05	0.525	0.5243645	0.0006355
0.1	0.54875	0.5474145	0.0013355
0.15	0.5711875	0.5690829	0.0021046
0.2	0.5922469	0.5892986	0.0029483
0.25	0.6118592	0.6079873	0.0038719
0.3	0.6299522	0.6250706	0.0048816
0.35	0.6464498	0.6404662	0.0059836
0.4	0.6612723	0.6540877	0.0071846
0.45	0.6743359	0.6658439	0.008492
0.5	0.6855527	0.6756394	0.0099133
0.55	0.6948303	0.6833735	0.0114568
0.6	0.7020718	0.6889406	0.0131312
0.65	0.7071754	0.6922296	0.0149458
0.7	0.7100342	0.6931236	0.0169106
0.75	0.7105359	0.6915	0.0190359
0.8	0.7085627	0.6872295	0.0213332
0.85	0.7039908	0.6801766	0.0238143
0.9	0.6966904	0.6701984	0.0264919
0.95	0.6865249	0.6571452	0.0293797
1.	0.6733511	0.6408591	0.0324921
Euler's Modified			
0.	0.5	0.5	0.
0.05	0.524375	0.5243645	0.0000105
0.1	0.5474367	0.5474145	0.0000222
0.15	0.5691179	0.5690829	0.000035
0.2	0.5893476	0.5892986	0.000049
0.25	0.6080517	0.6079873	0.0000644
0.3	0.6251519	0.6250706	0.0000813
0.35	0.6405659	0.6404662	0.0000997
0.4	0.6542074	0.6540877	0.0001197
0.45	0.6659855	0.6658439	0.0001416
0.5	0.6758048	0.6756394	0.0001654
0.55	0.6835648	0.6833735	0.0001913
0.6	0.68916	0.6889406	0.0002194
0.65	0.6924794	0.6922296	0.0002498
0.7	0.6934065	0.6931236	0.0002828
0.75	0.6918186	0.6915	0.0003186
0.8	0.6875868	0.6872295	0.0003572
0.85	0.6805756	0.6801766	0.000399
0.9	0.6706426	0.6701984	0.0004441
0.95	0.657638	0.6571452	0.0004929
1.	0.6414045	0.6408591	0.0005454
RK-4			
0.	0.5	0.5	0.
0.05	0.5243645	0.5243645	1.313D-09
0.1	0.5474145	0.5474145	2.761D-09
0.15	0.5690829	0.5690829	4.353D-09
0.2	0.5892986	0.5892986	6.102D-09
0.25	0.6079873	0.6079873	8.019D-09
0.3	0.6250706	0.6250706	1.012D-08
0.35	0.6404662	0.6404662	1.241D-08
0.4	0.6540877	0.6540877	1.491D-08
0.45	0.6658439	0.6658439	1.763D-08
0.5	0.6756394	0.6756394	2.059D-08
0.55	0.6833735	0.6833735	2.381D-08
0.6	0.6889406	0.6889406	2.731D-08
0.65	0.6922296	0.6922296	3.110D-08
0.7	0.6931237	0.6931236	3.521D-08
0.75	0.6915	0.6915	3.966D-08
0.8	0.6872296	0.6872295	4.447D-08
0.85	0.6801766	0.6801766	4.968D-08
0.9	0.6701985	0.6701984	5.530D-08
0.95	0.6571452	0.6571452	6.136D-08
1.	0.6408592	0.6408591	6.790D-08

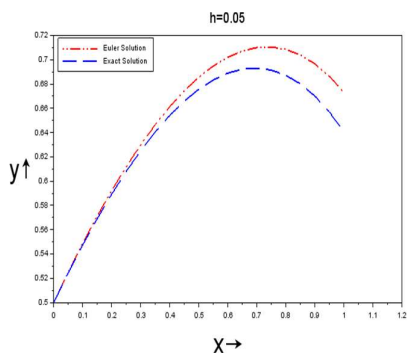


Figure 3(a). Approximate curve for step size $h=0.05$

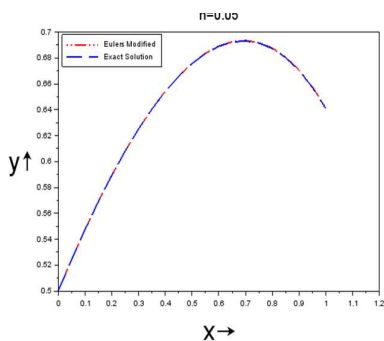


Figure 3(b). Approximate curve for step size $h=0.05$

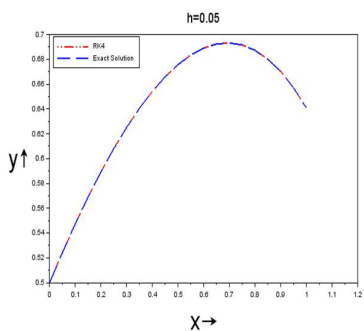


Figure 3(c). Approximate curve for step size $h=0.05$

Table 3. [Figure 4(a)-4(c)] Approximate solution for different step sizes with exact solution and absolute value

EULER			
h=0.01			
x	y	Exact Solution	Abs Error
0.	0.5	0.5	0.
0.01	0.505	0.5049749	0.0000251
0.02	0.50995	0.5098993	0.0000507
0.03	0.5148495	0.5147727	0.0000768
0.04	0.519698	0.5195946	0.0001034
0.05	0.524495	0.5243645	0.0001305
0.06	0.5292399	0.5290817	0.0001582
0.07	0.5339323	0.5337459	0.0001864
0.08	0.5385716	0.5383565	0.0002152
0.09	0.5431574	0.5429129	0.0002445
0.1	0.5476889	0.5474145	0.0002744
0.2	0.589905	0.5892986	0.0006064
0.3	0.6260755	0.6250706	0.0010049
0.4	0.6555681	0.6540877	0.0014805
0.5	0.6776841	0.6756394	0.0020447
0.6	0.6916517	0.6889406	0.0027111
0.7	0.6966183	0.6931236	0.0034947
0.8	0.6916424	0.6872295	0.0044129
0.9	0.6756837	0.6701984	0.0054852
1.	0.6475931	0.6408591	0.006734
Euler's Modified			
0.	0.5	0.5	0.
0.05	0.524375	0.5243645	0.0000105
0.1	0.5474367	0.5474145	0.0000222
0.15	0.5691179	0.5690829	0.000035
0.2	0.5893476	0.5892986	0.000049
0.25	0.6080517	0.6079873	0.0000644
0.3	0.6251519	0.6250706	0.0000813
0.35	0.6405659	0.6404662	0.0000997
0.4	0.6542074	0.6540877	0.0001197
0.45	0.6659855	0.6658439	0.0001416
0.5	0.6758048	0.6756394	0.0001654
0.55	0.6835648	0.6833735	0.0001913
0.6	0.68916	0.6889406	0.0002194
0.65	0.6924794	0.6922296	0.0002498
0.7	0.6934065	0.6931236	0.0002828
0.75	0.6918186	0.6915	0.0003186
0.8	0.6875868	0.6872295	0.0003572
0.85	0.6805756	0.6801766	0.000399
0.9	0.6706426	0.6701984	0.0004441
0.95	0.657638	0.6571452	0.0004929
1.	0.6414045	0.6408591	0.0005454
RK-4			
0.	0.5	0.5	0.
0.01	0.5049749	0.5049749	4.174D-13
0.02	0.5098993	0.5098993	8.431D-13
0.03	0.5147727	0.5147727	1.277D-12
0.04	0.5195946	0.5195946	1.720D-12
0.05	0.5243645	0.5243645	2.172D-12
0.06	0.5290817	0.5290817	2.633D-12
0.07	0.5337459	0.5337459	3.102D-12
0.08	0.5383565	0.5383565	3.581D-12
0.09	0.5429129	0.5429129	4.069D-12
0.1	0.5474145	0.5474145	4.567D-12
0.2	0.5892986	0.5892986	1.009D-11
0.3	0.6250706	0.6250706	1.673D-11
0.4	0.6540877	0.6540877	2.466D-11
0.5	0.6756394	0.6756394	3.406D-11
0.6	0.6889406	0.6889406	4.518D-11
0.7	0.6931236	0.6931236	5.825D-11
0.8	0.6872295	0.6872295	7.357D-11
0.9	0.6701984	0.6701984	9.147D-11
1.	0.6408591	0.6408591	1.123D-10

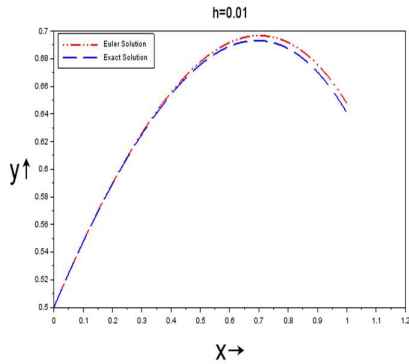


Figure 4(a): Approximate curve for step size $h=0.01$

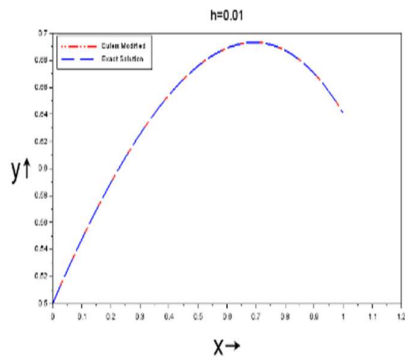


Figure 4(b). Approximate curve for step size $h=0.01$

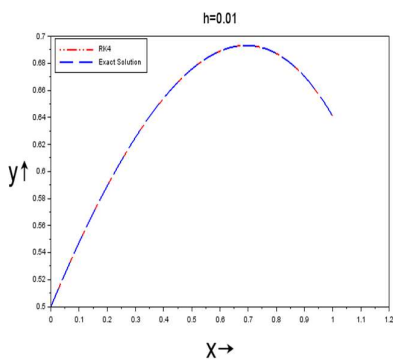


Figure 4(c). Approximate curve for step size $h=0.01$

3. Result Analysis: The programming has been done through Scilab and the codes successfully run. All the graphs have been plotted with the help of software at different values of step size, $h=0.1$, 0.05 and 0.01 for Euler’s method, Euler’s modified method

and RK-4 method respectively. The few properties like line, marker, style, fontsize, title, labels, linewidth, rotation and legend etc. have been used in the graph for betterment of the figures. The obtained resulting values are displayed in tabulated form for above three method and their comparison. For $h=0.1$ it is observed that there is some error in the result for Euler’s method as compared to Euler’s modified but RK-4 method gives better result among above three. Same observation is noticed when $h=0.05$ and 0.01 . The importance of RK-4 method is that it fits best in all cases. When decreasing the step size, h to be so small that absolute errors or round-off errors started to increase the accuracy, the calculations become very cumbersome for large values of n or alternatively large no. of step size; in other words we can say that the small value of h gives the better result for above methods. For RK-4 method at $h=0.01$ the plot from exact solution and by approximations coincides it means RK-4 gives better result as compared to others. In this paper for all cases, it has been observed that the convergence rate of approximate solutions to exact solutions is not so high but the amount of error is high when the considering step size is comparatively large; here again RK-4 gains high accuracy. So it is the reason why we opt the small value of step size, h . From the graphs and tables we can observe it easily. So step size, h play very important role to reducing the errors. Figure 1 is obtained by solving the differential equation directly or by ode command using Scilab. In figure 2(a)-2(c) the value of step size, h is 0.1 ; for figure 3(a)-3(c) the value of step size, h is 0.05 and finally for figure 4(a)-4(c) the step size, h is taken 0.01 . The curve of figures 1 is same as obtained by approximations shown in other figures. Finally, it has been observed that we can reach up to the accurate solutions of IVPs of ODEs by utilizing the approximations methods when the step size is so much small. Finally, the result comes out in favour of RK-4 for high accuracy among all.

4. Conclusions: Finally, the results are calculated and tabulated with approximation solution. Exact Solution and absolute errors done using Scilab coding. RK-4 method fits best in all cases; it is more effective while comparing to other numerical methods as it reduces the minimum error. Finally, the results of high accuracy found in favour of RK-4 among all. In all cases the role of step size, h is very important and if it is low then accuracy goes to high or in other words we can say that smaller value of step size, h is needed for better result. The best value of the solution is found to be at $x=0.4$ in all cases. We can take different value of step size, h . The curve of figure 1 as obtained by direct solution is same as by approximations for all three methods. The results have been compared and it has been found that RK-4 method gives best result among all for smaller h . It controls the accuracy and adjustment of the step size. All the graphs have been plotted using Scilab. It has an advantage that is free of cost, open-source and easily available to all for any operating systems. RK-4 has some applications. It having much applications in Mechanics, aerodynamics and in non-linear dynamics etc. and in was This method was adopted in the study of fluid mechanics where some applications on laminar and turbulent aerodynamics (fully implicit schemes) described by M.H. et al. in 2005. Finally, a turbulent Navier- Stokes used to show the nature of order reduction that encountered in high Reynolds number applications [19]. RK-4 was used for solving a set of non-linear transcendental power flow equations of power and the feasibility of this method is tested on 5-bus, IEEE 14-bus, 39-bus and 57-bus test system. this method is capable to obtain the global optimal solution of each power network by solving the dynamics of HNN. The operations of an analog RC electrical circuit can be analyzed using RK-4 method [20].

Conflict of Interest: Authors declare No conflicts of interest.

5. References:

- [1] Revathi. R. et al, Multicolored Approaches for Ordinary Differential Equation using Scilab, Journal of Propulsion Technology, 45 (2), 2024,1977-1985.
- [2] Nur Shahirah Zulkifli et al, Improving Euler Method using Centroidal-Polygon Scheme for Better Accuracy in Resistor Capacitor Circuit Equation, J. Phys.: Conf. Ser. 2319 012023, 2022, 1-9.
- [3] Ravindra Singh et al, The Study of A System of Non-Homogeneous Equations Using Matrix Exponential Method and Analysis With Scilab Software 6.1.1, J. Phys.: Conf. Ser. **2223** 012001, 2022, 1-12.
- [4] M. A. Arefin at.el , Accuracy Analysis for the Solution of Initial Value Problem of ODEs Using Modified Euler Method, I. J. Mathematical Sciences and Computing, 7(2), 2021, 31-41.
- [5] P. L. Sharma & A. Kumar, Demonstration Study on Runge-Kutta Fourth Order Method by Using MATLAB Programming, IOSR Journal of Mathematics 17(4), 2021, 1-9,
- [6] P. Yaswanth & L. Naveen, Numerical Solution of Ordinary Differential Equation using Scilab, International Journal of Scientific Development and Research, 4(10), 2019, 93-97.
- [7] V. Nirmala *et al*, Application of Runge-Kutta method for finding multiple numerical solutions to intuitionistic fuzzy differential equations, J. Phys.:Conf. Ser. 1139 012012, 1139, 2018, 1-8.
- [8] S. N. Kumar et al, A Study on Application of Runge- Kutta Method with Respect to Traffic Flow Problem, Journal of Emerging Technologies and Innovative Research (JETIR), 6(6), 2019, 121-126.
- [9] V. Chauhana & P. K. Srivastavab, Computational Techniques Based on Runge-Kutta Method of Various Order and Type for Solving Differential Equations, International Journal of Mathematical, Engineering and Management Sciences 4(2), 2019, 375–386,
- [10] G. C. Paul & S. S. Kumar, Exploration on initial structures of extrasolar

protoplanets via new explicit RKAHeM(4.4) method, The Egyptian Journal of Remote Sensing and Space Science, 18(1), 2015, 1-8.

[11] S. A. Agam & Y. A. Yahaya, A highly efficient implicit Runge-Kutta method for first order ordinary differential equations. African Journal of Mathematics and computer science research.7(5), 2014, 55-60.

[12] S. S. Kumar & G. C. Paul, Application of new RKAHeM(4.4) technique to analyze the structure of initial extrasolar giant protoplanets, Earth Science Informatics, 5(1), 2012, 23-31.

[13] M. Podisuk, International Journal of Mathematical Models and Methods in Applied Science, 5(2), 2011, 387-394.

[14] Steven C. Chapra & Raymond P. Canale, Numerical Methods for Engineers, Mc Graw Hill, International Edition (6th Edition), 2010, ISBN 978-0-07-339792-4.

[15] D. Houcque, Matlab Application Ordinary Differential Equation, MathWorks Inc. 2010, 1-12.

[16] S. C. Chapra & R. P. Canale, Numerical Methods For Engineers, McGraw Hill, New York Edition-5, 2006. ISBN: 0071244298.

[17] S. Rao, Applied Numerical Methods for Engineers and Scientist, Pearson Prentice Hall Education, 2002, ISBN: 978-0130894809.

[18] Gilberto. E. Urroz, Ordinary Differential Equations with SCILAB, Infor Clearinghouse.com, 2001.

[19] M. H. Carpanter et.al., Fourth-Order Runge-Kutta Schemes for Fluid Mechanics Applications, Journal of Scientific Computing, 25(1), 157-194, 2005.

[20] V. Veerasamy et.al., A novel RK4-Hopfield Neural Network for Power Flow Analysis of power system, Applied Soft Computing, 93, 2020, 106346