AN ANALOGOUS ILLUSTRATION OF THE EULER METHOD AND SECOND-ORDER RUNGE KUTTA METHOD FOR INITIAL VALUE PROBLEMS

Himangshu Barman[1*], Ranjana Mary Rajesh[2]

[1,2] School of Mathematics & Statistics, University of Hyderabad, Gachibowli, Telangana, India*

[1]Corresponding Author's Email ID: bhimangshu43@gmail.com*

ABSTRACT

Euler method and Runge-Kutta methods are discussed in this paper. Our aim is to find the best iterative numerical method between the Euler method and the Runge-kutta method for determining the solution of an initial value problem of an ordinary differential equation. Therefore, we have taken one initial value problem and found out the exact solution [8]. Later we find the approximate solution of the problem using the above-mentioned methods [12]. The approximate solution obtained for the problem is compared with the exact solution. The graphs consist of exact solutions and approximate solutions are drawn for each method for better investigation of the results.

KEYWORDS: *Euler method, Runge-kutta method, initial value problem, ordinary differential equation.*

INTRODUCTION

Numerical Analysis is one of the pretty branches of mathematics where we mostly discuss the approximate solution of a problem using many methods and techniques. But when we find the approximate solution to a problem using any numerical methods, errors always occur. That's why analysis of the numerical methods is a very difficult mathematical scheme by means of approximations. Two of numerical iterative methods are the Euler method and the Runge-Kutta method [11].

One of the simplest and oldest iterative methods, Euler method was discovered by a Swiss mathematician Leonhard Euler (1707-1783). This method was first found in the book "Institutionum Calculi Integrals" which was written by Leonhard Euler and the book was published in 1768-1770. The process of the Euler method is very slow [Sastry, 2013]. We need to take very small values of h for a better accuracy and that's why this method is less efficient in practical problems [Sastry, 2013]. This method can be used to find the approximate solution for first-order ordinary differential equations [9][13].

Runge-kutta method was discovered by German mathematicians Carl Runge (1856-1927) and Wilhelm Kutta (1867-1944) around 1900. This method has four orders first, second, third, and fourth. The Runge-Kutta method [9] of first-order is nothing but the Euler method. This method can also used to determine the approximate solution of an initial value problem of the ordinary differential equation [10].

EULER METHOD

Let us assume the problem $y' = f(x, y)$, $y(x_0) = y_0$.[Logan, 2011; Sastry, 2023]. Let (x_{n+1}, y_{n+1}) be the approximation obtained in the n^{th} iteration. "We want to find the value of y_n at $x = x_n$, where $x_n = x_0 + ns$, where $s = 0,1,2,3, \dots, (n-1)$. "[1][3]

We assume that $f(x, y) \approx f(x_0, y_0)$ in $x_0 \leq x \leq x_1$ [Rao,2006].

Integrating equation (1), we get,

$$\int_{x_0}^{x_1} dy = \int_{x_0}^{x_1} f(x,y)dx$$

$\Rightarrow y(x_1) - y(x_0) \approx \int_{x_0}^{x_1} f(x_0,y_0)dx$ [Replace $f(x,y)$ by $f(x_0,y_0)$]

$\Rightarrow y_1 - y_0 \approx f(x_0,y_0)\int_{x_0}^{x_1} dx$ [We assume $y(x_n) = y_n$]

$\Rightarrow y_1 \approx y_0 + f(x_0,y_0)(x_1 - x_0)$ [Rao, 2006]

$\Rightarrow y_1 \approx y_0 + s\, f(x_0,y_0)$ [Since each sub-interval is of length s], [1][3]

Similarly, by assuming that $f(x,y) \approx f(x_1,y_1)$ in $x_1 \le x \le x_2$, we get,

$$\Rightarrow y_2 \approx y_1 + s\, f(x_1,y_1)$$

“By proceeding in this way, we can conclude that

$\Rightarrow y_{n+1} \approx y_n + s\, f(x_n,y_n)\, ,\, n = 0,1,2,3, \dots$ “[1]

See the books [1][2]

SECOND-ORDER RUNGE-KUTTA METHOD

See the books [Issacson & Keller, 1966; Scheid, 1988; Sastry, 2013]]

ANALOGY BETWEEN TWO METHODS

Let’s take $y' = x + y,\ y(0) = 1$ and we want to determine the value of $y(0.8)$[1].

The given equation can be written as $y' - y = x$, which is nothing but a first order first degree ordinary linear differential equation. Therefore, integrating factor $= e^{\int(-1)dx} = e^{-x}$

Therefore, the solution is,

$$y.(\mathrm{I.F.}) = \int \mathrm{x.(I.F.)dx} + C$$

$$\Rightarrow y.e^{-x} = \int x.e^{-x}dx + C$$

$$\Rightarrow y.e^{-x} = -xe^{-x} - e^{-x} + C$$

$$\Rightarrow y(x) = -x - 1 + Ce^{x}$$

$y(0) = 1$, so $1 = -1 + C \Rightarrow C = 2$

Therefore $y(x) = -x - 1 + 2e^{x}$

So $y(0.8) = 2e^{0.8} - 1.8 = 2.651081$

C-program for Euler method:

```
“# include < stdio. h >

# include < conio. h >

# include < math. h >
```

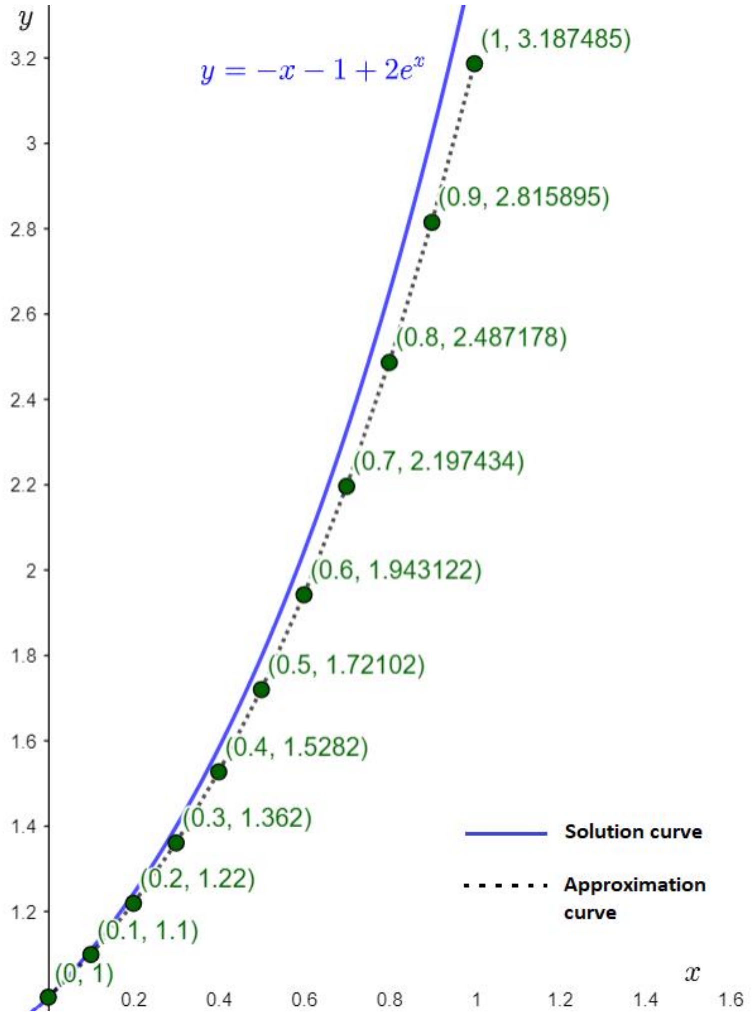
```
$\# define\ f(x, y)\ \ x + y$

$int\ main(\,)$

{

    $float\ x_0, y_0, x_n, y_n, s;$

    $int\ i, n;$

    $clrscr(\,);$

    $printf(\text{“}Enter\ x_0, y_0, x_n, n");$

    $scanf(\text{“}\%f\ \ \%f\ \ \%f\ \ \%d\text{”},\ \&x_0,\ \&y_0,\ \&x_n,\ \&n);$

    $s = \frac{x_n - x_0}{n}$

    $printf(\text{“}\backslash nx_0\backslash ty_0\backslash ty_n\backslash n");$

   $for(i = 0; i < n; i++)$

   {

        $y_n = y_0 + s * f(x_0, y_0);$

  $printf(\text{“}\%.6f\ \backslash t\%.6f\ \backslash t\%.6f\backslash n\text{”}, x_0, y_0, y_n);$

        $y_0 = y_n;$

        $x_0 = x_0 + s;$

   }

  $getch(\,);$

  $return\ 0;$

}” [4][5][6]
```

OUTPUT:

```
    x_0   y_0   x_n   n   0   1   1   10

    x_0                  y_0                   y_n

    0                1.000000             1.100000

    0.1              1.100000             1.220000
```

0.2	1.220000	1.362000
0.3	1.362000	1.528200
0.4	1.528200	1.721020
0.5	1.721020	1.943122
0.6	1.943122	2.197434
0.7	2.197434	2.487178
0.8	2.487178	2.815895
0.9	2.815895	3.187485
1	3.187485	

Graph for Euler method:



The approximate value of $y(0.8) = 2.487178$

Hence the error is

$$|Error| = |Exact\ value - approximate\ value|$$

$$\approx |2.651081 - 2.487178|$$

$$\approx 0.163903$$

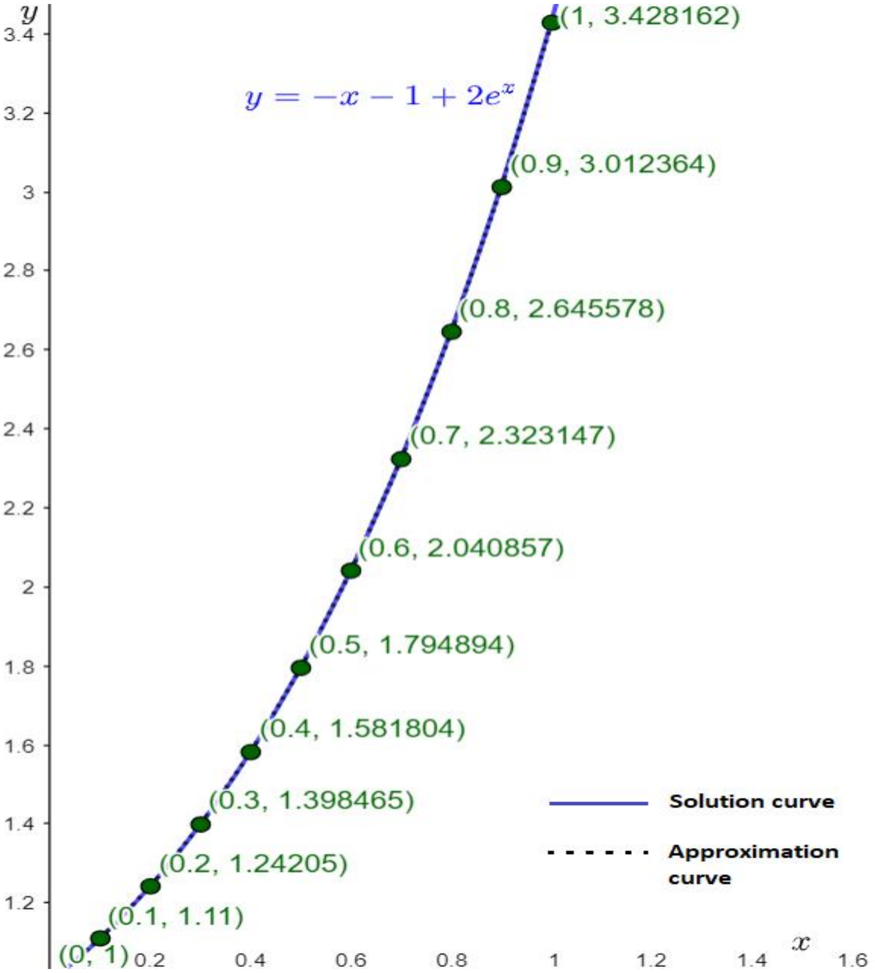
C-Program for Runge-Kutta Method

```
$“\# include < stdio.h >$
$\# include < conio.h >$
$\# include < math.h >$
$\# define\ f(x,y)\ \ x + y$
$int\ main(\,)$
${$
    $float\ x_0, y_0, x_n, y_n, s, K_1, K_2, K;$
    $int\ i, n;$
    $clrscr(\,);$
    $printf(“Enter\ x_0, y_0, x_n, n");$
    $scanf(“\%f\ \ \%f\ \ \%f\ \ \%d”,\ \&x_0,\ \&y_0,\ \&x_n,\ \&n);$
    $s = \frac{x_n - x_0}{n}$
    $printf(“\backslash nx_0\backslash ty_0\backslash ty_n\backslash n");$
   $for(i = 0; i < n; i++)$
   ${$
        $K_1 = s * f(x_0, y_0);$
        $K_2 = s * f(x_0 + s, y_0 + K_1);$
        $K = \frac{K_1 + K_2}{2};$
        $y_n = y_0 + K;$
        printf("%.6f \t%.6f \t%.6f\n", x0, y0, yn);

            x0 = x0 + s;

            y0 = yn;

         }

         getch( );

        return 0;

    }" [4][5][6]
```

OUTPUT:

```
    x0   y0   xn   n    0    1    1    10

    x0                y0                yn

    0                 1.000000          1.110000

    0.1               1.110000          1.242050

    0.2               1.242050          1.398465

    0.3               1.398465          1.581804

    0.4               1.5281804         1.794894

    0.5               1.794894          2.040857

    0.6               2.040857          2.323147

    0.7               2.323147          2.645578

    0.8               2.645578          3.012364

    0.9               3.012364          3.428162

    1                 3.428162
```

Graph For Runge-Kutta Method



The approximate value of $y(0.8) = 2.645578$

Hence the error is

$$|Error| = |Exact\ value - approximate\ value|$$

$$\approx |2.651081 - 2.645578|$$

$$\approx 0.005506$$

CONCLUSION

Euler method and Runge-Kutta method hold a very significant spot in numerical Analysis. We can find approximate solution using these methods [14]. The graphs have been drawn for each method showing the variation of exact solution and approximate solution obtained from c-programs of the respective methods. From the graph, it is clear that error is minimum in the second-order Runge-Kutta method and hence is the best method between them.

REFERENCE

[1] Sastry S. S. "Introductory Methods of Numerical Analysis", Fifth Edition, PHI Learning Private Limited Delhi-110092, November 2013, Page No: 302-315.

[2] Issacson E. and H.B. Keller, "Analysis of Numerical methods", John Willey & Sons, New York, 1996.

[3] Rao G. S. "Numerical Analysis", Revised Third Edition, New Age International limited Publishers, 2006, Page No: 217-240.

[4] http://www.codesansar.com

[5] http://www.bragitoff.com

[6] http://mcatutorials.com

[7] Scheid F. "Numerical Analysis", Schaum's Outline Series of Theory and Problems, Second Edition, McGraw-Hill, 1988, Page No: 197-232.

[8] Logan J.D. "A First Course in Differential Equations", Springer Science and Business Media LLC, 2011.

[9] http://fac.ksu.edu.sa

[10] www.akamaiuniversity.us

[11] www.cs.ru.nl

[12] Inc M., Evans D. J. "The Decomposition Method For Solving of A Class of Singular Two-Point Boundary Value Problems", International Journal of Computer Mathematics, 2003.

[13] Tanaka N. "On Hamiltonian Particle Dynamics", International Journal of Computational Fluid Dynamics, 2001.

[14] Hoffmann M., True H. "Dynamics of Two-axle railway freight wagons with UIC standard suspension", Vehicle System Dynamics, 2006.