The Fourth Order Runge-Kutta Method for Solving First Order Linear Equations Using C++ and Java

Oshinubi, Kayode Isaac
Institut de Mathématiques et de Sciences Physiques
School for Operations Research Techniques
Porto Novo
Benin Republic, West Africa
oshinubik@gmail.com

Adebesin Adesoji Ademola
Department of Computer Science/Engineering
Moshood Abiola Polytechnic
Abeokuta, Nigeria

Longe, O. Babatope PhD, Olowoyo Eniola Lois, Quadri Oluwajuwon & Oluwaseyi Pelumi
Department of Computer Science
Caleb University
Imota, Lagos State, Nigeria
E-mail: longeolumide@fulbrightmail.org

ABSTRACT

The derivation of fourth order Runge-Kutta method involves in computation of many unknowns and the detailed step by step derivation and analysis can hardly be found in many literatures. Due to the vital role played by the method in the field of computation and applied science/engineering, we simplify and further reduce the complexity of its derivation and analysis by exploring some possibly well-known works and propose a step by step derivation of the method. In this paper, the improved Runge-Kutta Method of Order 4 with four (4) stages for solving First Order Ordinary Differential Equation is proposed. The method is based on classical Runge-Kutta (RK) method which is considered as special class of two-step method. Here, the coefficients of the method are obtained using the minimization of the error norm up to order five. The improved method with only 4-stages is more accurate than fourth order 4-stages RK Method. A number of test problems are solved and the numerical results compared with the analytical solution.

Keywords: Fourth Order Runge-Kutta Method, First Order Linear Equations, C++ and Java

iSTEAMS Conference Proceedings Paper Citation Format
Oshinubi, Kayode Isaac, Olowoyo Eniola Lois, Quadri Oluwajuwon & Oluwaseyi Pelumi (2018): The Fourth Order Runge-Kutta Method for Solving First Order Linear Equations Using C++ and Java. Proceedings of the 14th iSTEAMS International Multidisciplinary Conference, AlHikmah University, Ilorin, Nigeria, Vol. 14, Pp 77-94.

1. INTRODUCTION

In numerical analysis, the Runge-Kutta methods are a family of implicit and explicit iterative methods, which include the well-known routine called the Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations. These methods were developed around 1900 by the German mathematicians C. Runge and M. W. Kutta Runge-Kutta formulas are among the oldest and best understood schemes in numerical analysis. However, despite the evolution of a vast and comprehensive body of knowledge, it continues to be a source of active research. Runge-Kutta methods provide a popular way to solve the initial value problem for a system of ordinary differential equations. The most widely known member of the Runge-Kutta family is generally referred to as "RK4", "classical Runge-Kutta method" or simply as "the Runge-Kutta method".Let an initial value problem be specified as follows:

$\dot{y} = f(t,y), y(t_0) = y_0$

Here y is an unknown function (scalar or vector) of time t, which we would like to approximate; we are told that ẏ, the rate at which y changes, is a function of t and of y itself. At the initial time t_0 , the corresponding y value is y_0. The function f and the data t_0 , y_0 are given.

Now pick a step-size h > 0 and define

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

for n = 0, 1, 2, 3, ..., using

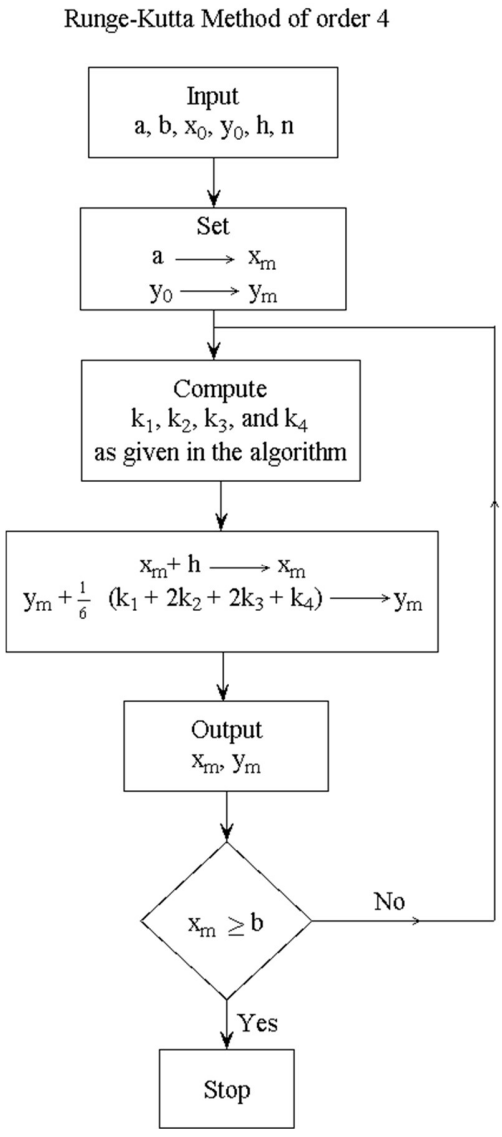
$$k_1 = f(t_n , y_n)$$
$$k_2 = f\left(t_n + \frac{h}{2} , y_n + h\frac{k_1}{2}\right)$$
$$k_3 = f\left(t_n + \frac{h}{2} , y_n + h\frac{k_2}{2}\right)$$
$$k_4 = f(t_n + h , y_n + h\,k_3)$$

Here y_{n+1} is the RK4 approximation of $y(t_{n+1})$, and the next value (y_{n+1}) is determined by the present value (y_n) plus the weighted average of four increments, where each increment is the product of the size of the interval, h, and an estimated slope specified by function f on the right-hand side of the differential equation.

- k_1 is the increment based on the slope at the beginning of the interval, using y (Euler's method);
- k_2 is the increment based on the slope at the midpoint of the interval, using y and k_1;
- k_3 is again the increment based on the slope at the midpoint, but now using y and k_2;
- k_4 is the increment based on the slope at the end of the interval, using y and k_3.

In averaging the four increments, greater weight is given to the increments at the midpoint. If f is independent of y, so that the differential equation is equivalent to a simple integral, then RK4 is Simpson's rule.
The RK4 method is a fourth-order method, meaning that the local truncation error is on the order of $O(h^5)$, while the total accumulated error is on the order of $O(h^4)$.

2. FLOWCHART OF FOURTH ORDER RUNGE-KUTTA



3. EXAMPLES, IMPLEMENTATION AND OUTPUT OF SOME EXAMPLES SOLVED WITH FOURTH ORDER RUNGE-KUTTA METHOD IN C++

Examples.
Given y(0) = 1 and take h = 0.2

1. $dy/dx = 1 + y^2$
2. $dy/dx = x + y$
3. $dy/dx = x – y$
4. $dy/dx = x^2 + xy$
5. $dy/dx = y – x^2 + 1$
6. $dy/dx = 8 - 3y$

The Examples Above Are Solved Below Using C++ Program

```
#include <iostream>  //Header file for cin & cout
#include <cmath>  //Header file for mathematical operations
#include <iomanip> //Header file for precession
#include <string>
#include <stdlib.h>
#include <conio.h>

using namespace std;  //calling the standard directory

int option, n, i, j;
float h;

//Given dy/dx
float f(float x, float y)
{
   return (1+pow(y,2));
}

float g(float a, float b)
{
        return (a+b);
}

float d(float u, float l)
{
   return (u-l);
}

float z(float r, float s)
{
        return (pow(r,2)+(r*s));
}

float m(float t, float o)
{
        return (o-(pow(t,2))+ 1);
}

float p(float q, float w)
{
   return (8-(3*w));
}
```

```
int main()
{
   cout << "INSTITUTION: CALEB UNIVERSITY, IMOTA LAGOS, NIGERIA." << endl;
   cout << "DEPARTMENT: COMPUTER SCIENCE, STATISTICS AND MATHEMATICS." << endl;
   cout << "COURSE: MAT 313 - COMPUTATIONAL SCIENCE AND NUMERICAL METHODS" << endl;
   cout << "----------" << endl;
   cout << "OBJECTIVE: To solve a First Order Ordinary Differential equation by Fourth Order Runge-Kutta's
Method" << endl;
   cout << "----------\n" << endl;
   cout << "Given Equation:" << endl;
   cout << "1: Solve dy/dx = 1 + y^2" << endl;
   cout << "2: Solve dy/dx = x + y" << endl;
   cout << "3: Solve dy/dx = x - y" << endl;
   cout << "4: Solve dy/dx = x^2 + x*y" << endl;
   cout << "5: Solve dy/dx = y - x^2 + 1" << endl;
   cout << "6: Solve dy/dx = 8 - (3 * y)\n" << endl;
   cout << "7: ABOUT THE PROGRAM" << endl;
   cout << "8: Exit\n" << endl;
   cout << "Select an Option: ";
   cin >> option;

   if(option == 1)
   {
      system("cls");
      int n, i, j;
      float h;

      cout << "Enter the Number of Solutions: ";
      cin >> n;

      cout << "Enter The Value of Step Size: ";
      cin >> h;

      long double y[n], x[n], k[n][n];

      cout << "Enter the value of x0: ";
      cin >> x[0];

      cout << "Enter The Value of y0: ";
      cin >> y[0];

      for(i=1;i<=n;i++)
      {
         x[i]=x[i-1]+h;
      }

      cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

      for(j=1;j<=n;j++)
      {
         cout << "when x = " <<h*j<<","<<endl;

         k[1][j] = h*f(x[j-1], y[j-1]);
         cout << "K[1] = "<< k[1][j]<< "\t";
```

```
        k[2][j] = h*f(x[j-1]+(h/2), y[j-1]+(k[1][j]/2));
        cout << "K[2] = "<< k[2][j]<< "\t";

        k[3][j] = h*f(x[j-1]+(h/2), y[j-1]+(k[2][j]/2));
        cout << "K[3] = "<< k[3][j]<< "\t";

        k[4][j] = h*f(x[j-1]+(h), y[j-1]+k[3][j]);
        cout << "K[4] = "<< k[4][j]<< "\n";

        y[j] = y[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
        cout << "y["<<h*j<<"]="<<setprecision(10)<<y[j]<<"\n"<<endl;
    }
}

else if(option == 2)
{
    system("cls");
    int n, i, j;
    float h;

    cout << "Enter the Number of Solutions: ";
    cin >> n;

    cout << "Enter The Value of Step Size: ";
    cin >> h;

    long double b[n],e[n],a[n],Y[n],L[n],k[n][n];

    cout << "Enter the value of x0: ";
    cin >> a[0];

    cout << "Enter The Value of y0: ";
    cin >> b[0];

    for(i=1;i<=n;i++)
    {
          a[i]=a[i-1]+h;
    }

    cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

    for(j=1;j<=n;j++)
    {
        cout << "when x = " <<h*j<<","<<endl;

        k[1][j] = h*g(a[j-1], b[j-1]);
        cout << "K[1] = "<< k[1][j]<< "\t";

        k[2][j] = h*g(a[j-1]+(h/2), b[j-1]+(k[1][j]/2));
        cout << "K[2] = "<< k[2][j]<< "\t";

        k[3][j] = h*g(a[j-1]+(h/2), b[j-1]+(k[2][j]/2));
        cout << "K[3] = "<< k[3][j]<< "\t";
```

```
        k[4][j] = h*g(a[j-1]+(h), b[j-1]+k[3][j]);
        cout << "K[4] = "<< k[4][j]<< "\n";

        b[j] = b[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
        cout << "y["<<h*j<<"]="<<setprecision(10)<<b[j]<<"\n"<<endl;

    }
}

else if(option == 3)
{
    system("cls");
    int n, i, j;
    float h;

    cout << "Enter the Number of Solutions: ";
    cin >> n;

    cout << "Enter The Value of Step Size: ";
    cin >> h;

    long double l[n],e[n],u[n],Y[n],L[n],k[n][n];

    cout << "Enter the value of x0: ";
    cin >> u[0];

    cout << "Enter The Value of y0: ";
    cin >> l[0];

    for(i=1;i<=n;i++)
    {
        u[i]=u[i-1]+h;
    }

    cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

    for(j=1;j<=n;j++)
    {
        cout << "when x = " <<h*j<<","<<endl;

        k[1][j] = h*d(u[j-1], l[j-1]);
        cout << "K[1] = "<< k[1][j]<< "\t";

        k[2][j] = h*d(u[j-1]+(h/2), l[j-1]+(k[1][j]/2));
        cout << "K[2] = "<< k[2][j]<< "\t";

        k[3][j] = h*d(u[j-1]+(h/2), l[j-1]+(k[2][j]/2));
        cout << "K[3] = "<< k[3][j]<< "\t";

        k[4][j] = h*d(u[j-1]+(h), l[j-1]+k[3][j]);
        cout << "K[4] = "<< k[4][j]<< "\n";

        l[j] = l[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
        cout << "y["<<h*j<<"]="<<setprecision(10)<<l[j]<<"\n"<<endl;
```

```
    }
}

else if(option == 4)
{
    system("cls");
    int n, i, j;
    float h;

    cout << "Enter the Number of Solutions: ";
    cin >> n;

    cout << "Enter The Value of Step Size: ";
    cin >> h;

    long double s[n],e[n],r[n],Y[n],L[n],k[n][n];

    cout << "Enter the value of x0: ";
    cin >> r[0];

    cout << "Enter The Value of y0: ";
    cin >> s[0];

    for(i=1;i<=n;i++)
    {
        r[i]=r[i-1]+h;
    }

    cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

    for(j=1;j<=n;j++)
    {
        cout << "when x = " <<h*j<<","<<endl;

        k[1][j] = h*z(r[j-1], s[j-1]);
        cout << "K[1] = "<< k[1][j]<< "\t";

        k[2][j] = h*z(r[j-1]+(h/2), s[j-1]+(k[1][j]/2));
        cout << "K[2] = "<< k[2][j]<< "\t";

        k[3][j] = h*z(r[j-1]+(h/2), s[j-1]+(k[2][j]/2));
        cout << "K[3] = "<< k[3][j]<< "\t";

        k[4][j] = h*z(r[j-1]+(h), s[j-1]+k[3][j]);
        cout << "K[4] = "<< k[4][j]<< "\n";

        s[j] = s[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
        cout << "y["<<h*j<<"]="<<setprecision(10)<<s[j]<<"\n"<<endl;

    }
}

else if(option == 5)
{
    system("cls");
```

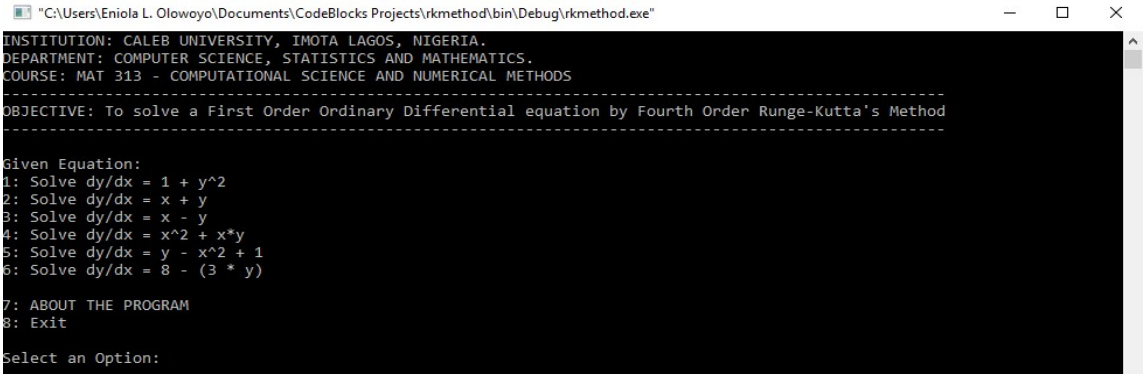
```
    int n, i, j;
    float h;

    cout << "Enter the Number of Solutions: ";
    cin >> n;

    cout << "Enter The Value of Step Size: ";
    cin >> h;

    long double o[n],e[n],t[n],Y[n],L[n],k[n][n];

    cout << "Enter the value of x0: ";
    cin >> t[0];

    cout << "Enter The Value of y0: ";
    cin >> o[0];

    for(i=1;i<=n;i++)
    {
       t[i]=t[i-1]+h;
    }

    cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

    for(j=1;j<=n;j++)
    {
       cout << "when x = " <<h*j<<","<<endl;

       k[1][j] = h*m(t[j-1], o[j-1]);
       cout << "K[1] = "<< k[1][j]<< "\t";

       k[2][j] = h*m(t[j-1]+(h/2), o[j-1]+(k[1][j]/2));
       cout << "K[2] = "<< k[2][j]<< "\t";

       k[3][j] = h*m(t[j-1]+(h/2), o[j-1]+(k[2][j]/2));
       cout << "K[3] = "<< k[3][j]<< "\t";

       k[4][j] = h*m(t[j-1]+(h), o[j-1]+k[3][j]);
       cout << "K[4] = "<< k[4][j]<< "\n";

       o[j] = o[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
       cout << "y["<<h*j<<"]="<<setprecision(10)<<o[j]<<"\n"<<endl;

    }
}

else if(option == 6)
{
    system("cls");
    int n, i, j;
    float h;

    cout << "Enter the Number of Solutions: ";
    cin >> n;
```

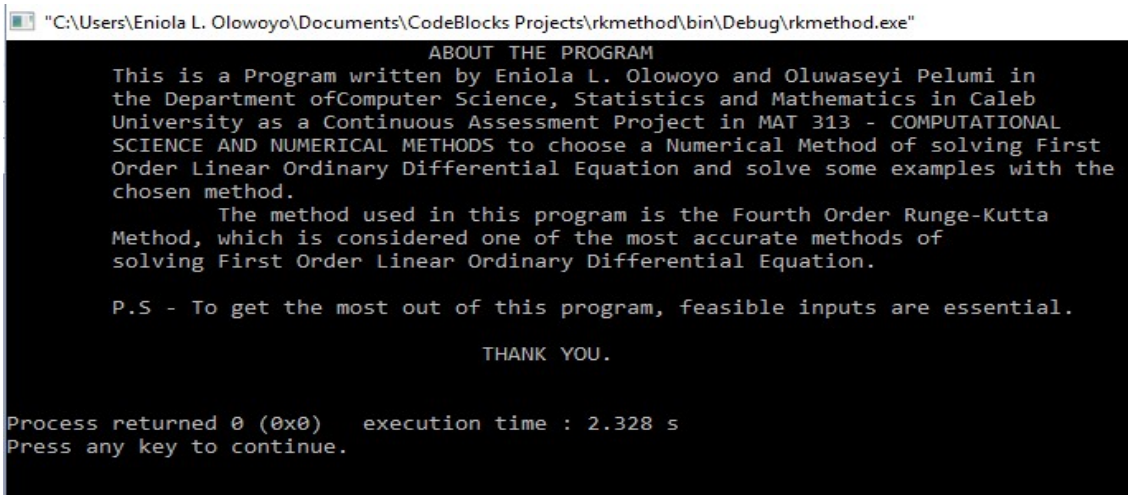
```
    cout << "Enter The Value of Step Size: ";
    cin >> h;

    long double w[n],e[n],q[n],Y[n],L[n],k[n][n];

    cout << "Enter the value of x0: ";
    cin >> q[0];

    cout << "Enter The Value of y0: ";
    cin >> w[0];

    for(i=1;i<=n;i++)
    {
       q[i]=q[i-1]+h;
    }

    cout << "\nSolution of the given Differential Equation by Fourth Order Runge-Kutta Method is: \n" << endl;

    for(j=1;j<=n;j++)
    {
       cout << "when x = " <<h*j<<","<<endl;

       k[1][j] = h*p(q[j-1], w[j-1]);
       cout << "K[1] = "<< k[1][j]<< "\t";

       k[2][j] = h*p(q[j-1]+(h/2), w[j-1]+(k[1][j]/2));
       cout << "K[2] = "<< k[2][j]<< "\t";

       k[3][j] = h*p(q[j-1]+(h/2), w[j-1]+(k[2][j]/2));
       cout << "K[3] = "<< k[3][j]<< "\t";

       k[4][j] = h*p(q[j-1]+(h), w[j-1]+k[3][j]);
       cout << "K[4] = "<< k[4][j]<< "\n";

       w[j] = w[j-1] + ((k[1][j]+2*k[2][j]+2*k[3][j]+k[4][j])/6);
       cout << "y["<<h*j<<"]="<<setprecision(10)<<w[j]<<"\n"<<endl;

    }
}

else if (option == 7)
{
    system("cls");

    cout << "\t\t\t\tABOUT THE PROGRAM\n";
    cout << "\tThis is a Program written by Eniola L. Olowoyo and Oluwaseyi Pelumi in\n";
    cout << "\tthe Department ofComputer Science, Statistics and Mathematics in Caleb\n";
    cout << "\tUniversity as a Continuous Assessment Project in MAT 313 - COMPUTATIONAL\n";
    cout << "\tSCIENCE AND NUMERICAL METHODS to choose a Numerical Method of solving First\n";
    cout << "\tOrder Linear Ordinary Differential Equation and solve some examples with the\n";
    cout << "\tchosen method.\n";
    cout << "\t\tThe method used in this program is the Fourth Order Runge-Kutta\n";
    cout << "\tMethod, which is considered one of the most accurate methods of\n";
    cout << "\tsolving First Order Linear Ordinary Differential Equation.\n" << endl;
    cout << "\tP.S - To get the most out of this program, feasible inputs are essential.\n" << endl;
```

```
        cout << "\t\t\t\t    THANK YOU.\n" << endl;
    }

    else if (option == 8)
    {
        exit(EXIT_SUCCESS);
    }

    else
    {
        cout << "--------------------------------------------------------------------------------------------------------" << endl;
    }

    return 0;

}
```

4.1 Output of the Programs Written

Interface





Outputs For the First Example

5. EXAMPLES, IMPLEMENTATION AND OUTPUT OF SOME EXAMPLES SOLVED WITH FOURTH ORDER RUNGE-KUTTA METHOD IN JAVA

Examples.

1. $dy/dx = 3e^{-x} - 0.4y$, $y(0)=5$. Find $y(3)$ using $h=1.5$
2. $dy/dx = -xy^2$. When $y(0)=4$ and $h=0.5$, find $y(5)$.
3. $dy/dx = 3e^{-x} - 1.5y$, $y(0)=5$. Find $y(4)$ using $h=0.4$

The examples above are solved below using java program

```
package runge.kutta;

import java.util.Scanner;

/**
 *
 * @author CULStudent
 */
public class RungeKutta {

    public static double equation(double xi, double yi)
    {
        return (3 * Math.exp(-xi)) - 0.4 * (yi);
    }
    public static double equation2(double x,double y)
    {
        return (-x*Math.pow(y,2));
    }
    public static double equation3(double x,double y)
    {
        return (3*Math.exp(-x)) - (1.5*y);
    }
    public static double equationSelection(int n,double x,double y)
    {
        double computation = 0;
        switch(n)
        {
            case 1:
                computation = equation(x,y);
                break;
            case 2:
                computation = equation2(x,y);
                break;
            case 3:
                computation = equation3(x,y);
                break;
        }
        return computation;
    }
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner input = new Scanner(System.in);
        System.out.println("Numerical  Method: Runge - kutta 4th Order method");
        System.out.println("QUESTIONS");
        System.out.println("dy/dx = 3e^(-x) - 0.4y, y(0) = 5 find y(3) using h = 1.5");
```

```
System.out.println("dy/dx = -xy^2, When y(0)=4 and h=0.5 find y(5)");
System.out.println("dy/dx = 3e^-x - 1.5y, y(0)=5 find y(4) using h=0.4");
System.out.println("Find y(3) using 1.5");
System.out.println("--------------------------------");
System.out.println("SOLUTION");
System.out.println("--------------------------------");

double x0;
double y0;
double h;
int iterations;
int eq;
System.out.print("Enter value for X0:");
x0 = input.nextDouble();
System.out.println();
System.out.print("Enter value for Y0:");
y0 = input.nextDouble();
System.out.println();
System.out.print("Enter value for Step size:");
h = input.nextDouble();
System.out.println();
System.out.print("Enter number of iterations: ");
iterations = input.nextInt();
System.out.println();
System.out.println("Please select equation to Use");
System.out.println("-----------------------------");
//3 * Math.exp(-xi)) - 0.4 * (yi)
//-x*Math.pow(y,2)
//(3*Math.exp(-x)) - (1.5*y)

while(true)
{
   System.out.println("(1)3e^-x -0.4y ");
   System.out.println("(2)-xy^2");
   System.out.println("(3)3e^-x - 1.5y");
   System.out.println();
   System.out.print("Enter option number: ");
   eq = input.nextInt();
   System.out.println();
   if(eq == 1 || eq == 2 || eq == 3)
      break;
   else
      System.out.println("Enter numers 1,2 or 3 only");
}

for(int i = 0; i<iterations;i++)
{
   //k1
   double k1 = equationSelection(eq,x0,y0);
   //
   double kn1 = (x0 + ((0.5)*h));
   double kn2 = (y0 + ((0.5) * k1 * h));

   //k2
   double k2 = equationSelection(eq,kn1,kn2);
```

```
    kn2 = (y0 + ((0.5) * k2 * h));

    //k3
    double k3 = equationSelection(eq,kn1,kn2);

    kn1 = (x0 + h);
    kn2 = (y0 + ((k3) * h));

    //k4
    double k4 = equationSelection(eq,kn1,kn2);
    System.out.println("K1 = "+k1);
    System.out.println("K2 = "+k2);
    System.out.println("K3 = "+k3);
    System.out.println("K4 = "+k4);


    double sum_of_ks = (k1 + (2*k2) + (2*k3) + k4);
    double j =  sum_of_ks * h;
    double g = (1/6.0)*j;
    double Yi = y0 + g;

    int num = i+1;
    System.out.println("Y("+num+") = "+Yi);
    System.out.println("---------------------------------");
    y0 = Yi;
    x0 = x0 + h;

}
```

4.1 Output of the Programs Written

For the First Example

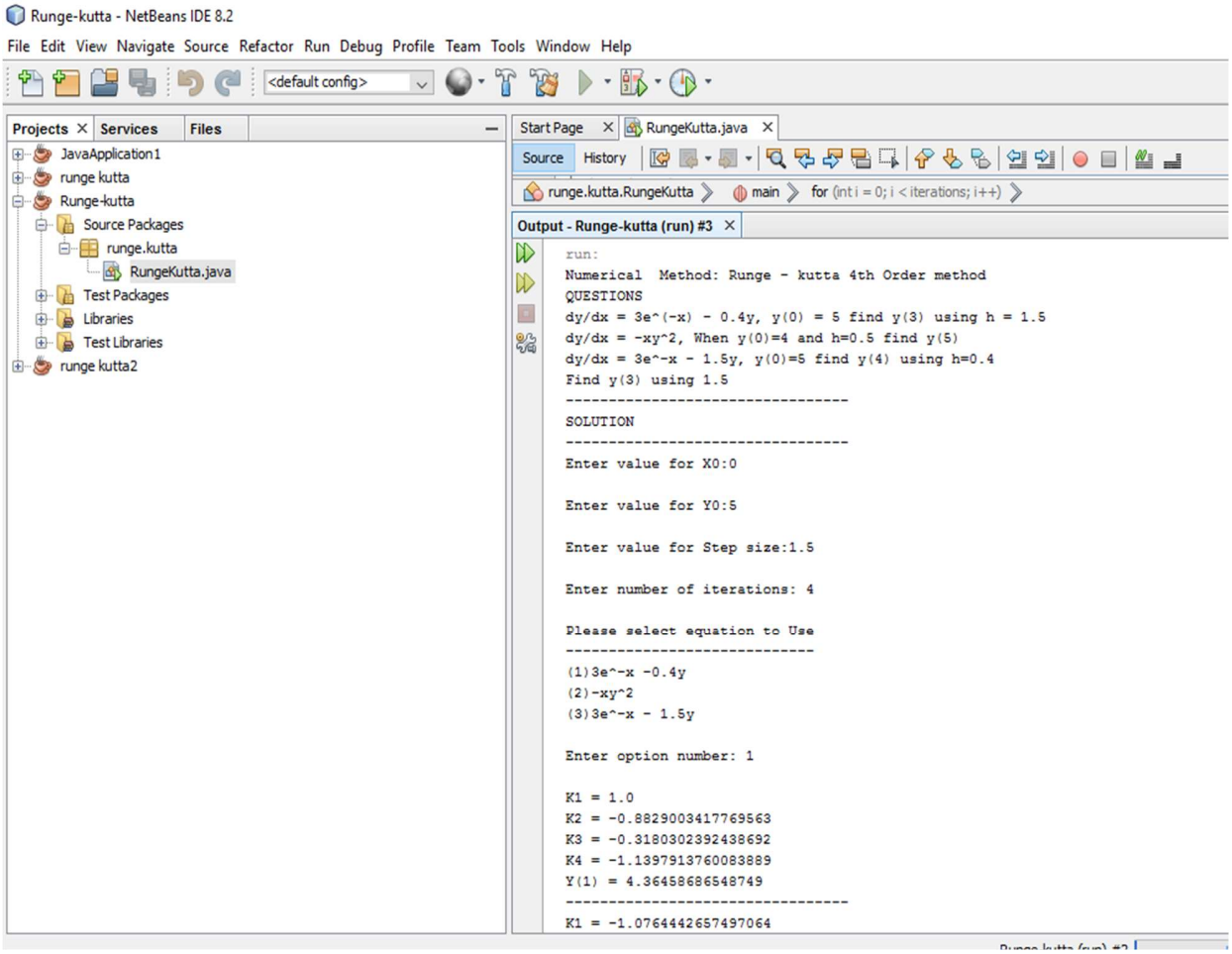


Fig 2: Output of the Programs Written For the First Example

5. ANALYTICAL SOLUTION OF THE EXAMPLES USED IN THE C++ IMPLEMENTATION

The examples used for implementation are those which technically cannot be solved by hand-written analytical solution methods. The use of Numerical Solutions makes the solution feasible. Hence:

S/N	Equations	Analytical Solution
1	dy/dx = 1 + y^2	$\tan(x + \frac{\pi}{4})$
2	dy/dx = x + y	$-x + 2e^{x} - 1$
3	dy/dx = x – y	$x - 1 + 2e^{-x}$
4	dy/dx = x^2 + xy	$-x + \frac{\sqrt{2}\sqrt{\pi}}{2} e^{\frac{x^2}{2}} \operatorname{erf}\left(\frac{\sqrt{2x}}{2}\right) + e^{\frac{x^2}{2}}$
5	dy/dx = y – x^2 + 1	$x^2 + 2x + 1$
6	dy/dx = 8 - 3y	$\frac{8}{3} - \frac{5}{3} e^{-3x}$

COMPARISON OF SOME ANALYTICAL SOLUTIONS AND THE NUMERICAL SOLUTION WITH COMPUTATION OF THE ERROR OF C++ EXAMPLES

S/N	Equation	Points of x	Analytical Solution	Numerical Solution	Error
1	dy/dx = x + y	0.2	1.242805516	1.242800004	5.512E-06
		0.4	1.588364934	1.583635926	0.004729008
		0.6	2.044237601	2.044212925	2.4676E-05
		0.8	2.651081857	2.651041677	4.018E-05
		1.0	3.436563657	3.436502338	6.1319E-05
2	dy/dx = x – y	0.2	0.837461506	0.837466666	-5.16E-06
		0.4	0.740640092	0.740648541	-8.499E-06
		0.6	0.697623272	0.697633647	-1.0375E-05
		0.8	0.698657928	0.698669256	-1.1328E-05
		1.0	0.735758882	0.735770478	-1.1596E-05
3	dy/dx = y – x^2 + 1	0.2	1.440000000	1.439993342	6.658E-05
		0.4	1.960000000	1.959985209	1.4791E-05
		0.6	2.560000000	2.559975283	2.4717E-05
		0.8	3.240000000	3.239963162	3.6838E-05
		1.0	4.000000000	3.999948329	5.1671E-05
4	dy/dx = 8 - 3y	0.2	1.751980607	1.751000010	0.000980597
		0.4	2.164676313	2.163599401	0.001076912
		0.6	2.391168520	2.390281495	0.000887025
		0.8	2.515470078	2.514820682	0.000649396
		1.0	2.583688219	2.583242500	0.000445719

6. ANALYTICAL SOLUTION OF THE EXAMPLES USED IN THE JAVA IMPLEMENTATION

The examples used for implementation are those which technically cannot be solved by hand-written analytical solution methods. The use of Numerical Solutions makes the solution feasible. Hence:

S/N	Equations	Analytical Solution
1	$dy/dx = 3e^{-x} - 0.4y$	$\left(10e^{\frac{3x}{5}} - 5\right)e^{-x}$
2	$dy/dx = -xy^2$	$\frac{4}{2x^2 + 1}$
3	$dy/dx = 3e^{-x} - 1.5y$	$6e^{-x} - e^{-\frac{3x}{2}}$

COMPARISON OF THE ANALYTICAL SOLUTIONS AND THE NUMERICAL SOLUTION WITH COMPUTATION OF THE ERROR OF JAVA EXAMPLES

S/N	Equation	Points of x	Analytical Solution	Numerical Solution	Error
1	$dy/dx = 3e^{-x} - 0.4y$	1.5	4.372465560	4.364586865	0.007878695
		3.0	2.763006777	2.758836440	0.004170337
		4.5	1.597443900	1.596239648	0.001204252
		6.0	0.894785772	0.894943830	-0.000158058
2	$dy/dx = -xy^2$	0.5	2.666666667	2.613932292	0.052734375
		1.0	1.333333333	1.331067568	0.002265765
		1.5	0.727272727	0.729646776	-0.002374049
		2.0	0.444444444	0.446113900	-0.001669456
		2.5	0.296296296	0.297248156	-0.000951860
3	$dy/dx = 3e^{-x} - 1.5y$	0.4	3.473108640	3.474227538	-0.001118898
		0.8	2.394779573	2.396215806	-0.001436233
		1.2	1.641866383	1.643245360	-0.001378977
		1.6	1.120661155	1.121835726	-0.001176571

7. CONCLUSION

In this paper, we have simplified the existing derivation and analysis of the fourth order Runge-Kutta Method for easy reference to students and plot the stability region. We also reduced the complexity of the method by proposing a step by step derivation approach for better understanding to students.

REFERENCES

[1] M.K. Jain, S.R.K. Iyengar, R.K. Jain, (2007), Numerical Methods for Scientific and Engineering Computing.
[2] J. D. Lambert, (1991), Numerical Methods for Ordinary Differential Systems, the initial value Problem, John Wiley & Sons Ltd.
[3] J.C. Butcher, (2003), Numerical Methods for Ordinary Differential Equations, John Wiley & Sons Ltd.
[4] John R. Dorman, (1996), Numerical Methods for Differential Equations, a Computational Approach, CRC Press, Inc.