RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 10

Date of Experiment: 06.05.2018
Date of Submission: 25.06.2018

Submitted to:
Shyla Afroge
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Name of the Experiment: Implementation of Runge-Kutta Method (2nd order and 4th order)

Theory:

Suppose that we wish to solve $y' = f(x, y)$ for values of y at $x = x_r = x_0 + rh$ $(r = 1,2, \dots)$. Integrating the equation, we get,

$$y_1 = y_0 + \int_{x_0}^{x_1} f(x, y)dx$$

Approximating the integral given in the equation above by means of Trapezoidal rule to obtain,

$$y_1 = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1)]$$

Substituting $y_1 = y_0 + hf(x_0, y_0)$ on the right side of the equation above, we obtain,

$$y_1 = y_0 + \frac{h}{2}[f_0 + f(x_0 + h, y_0 + hf_0)] \qquad where, f_0 = f(x_0, y_0)$$

$If\ now\ we\ set, k_1 = hf_0 \quad and \quad k_2 = hf(x_0 + h, y_0 + k_1) \quad then\ the\ above\ equation\ becomes,$

$$y_1 = y_0 + \frac{1}{2}(k_1 + k_2)$$

This is the second-order Runge-Kutta formula.

The forth-order Runge-Kutta formula is as follows-

$$y_1 = y_0 + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)$$

$where,$

$$k_1 = hf(x_0, y_0)$$
$$k_2 = hf\left(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_1\right)$$
$$k_3 = hf\left(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_2\right)$$
$$k_4 = hf(x_0 + h, y_0 + k_3)$$

Code:

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;
double func_value(double x, double y)
{
    return (y - x);
}
int second_order(void)
{
    double temp;
    double h;
    int choice, i;
    double x_check;
    while (1)
    {
        vector<double> x;
        vector<double> y;
        printf("1. New value\n2. New h\n0. Terminate\n  Enter choice:
");
        cin >> choice;
        switch (choice)
        {
        case 0:
            return 1;
```

```
        case 1:
            printf("Enter initial value of x: ");
            cin >> temp;
            x.push_back(temp);
            printf("Enter initial value of y: ");
            cin >> temp;
            y.push_back(temp);
        xx:
            printf("Enter value of h: ");
            cin >> h;
            printf("Enter the value of x to find y: ");
            cin >> x_check;
            i = 0;
            while (x[i] <= x_check)
            {
                temp = x[i] + h;
                x.push_back(temp);
                temp = y[i] + (h / 2) * (func_value(x[i], y[i]) +
func_value(x[i + 1], y[i] + h * func_value(x[i], y[i])));
                y.push_back(temp);
                i++;
            }
            cout << "y(" << x_check << ")=" << y[i] << endl;
            break;
        case 2:
            goto xx;
            break;
        default:
            printf("Wrong input.\n");
        }
    }
}
int forth_order(void)
{
    double temp;
    double h;
    int choice, i;
    double x_check;
    double k1, k2, k3, k4;
    while (1)
    {
        vector<double> x;
        vector<double> y;
        printf("1. New value\n2. New h\n0. Terminate\n   Enter choice:
");
        cin >> choice;
        switch (choice)
        {
        case 0:
            return 1;
            break;
        case 1:
            printf("Enter initial value of x: ");
            cin >> temp;
            x.push_back(temp);
```

```
            printf("Enter initial value of y: ");
            cin >> temp;
            y.push_back(temp);
        xx:
            printf("Enter value of h: ");
            cin >> h;
            printf("Enter the value of x to find y: ");
            cin >> x_check;
            i = 0;
            while (x[i] <= x_check)
            {
                temp = x[i] + h;
                x.push_back(temp);
                k1 = h * func_value(x[i], y[i]);
                k2 = h * func_value(x[i] + (h / 2), y[i] + (k1 / 2));
                k3 = h * func_value(x[i] + (h / 2), y[i] + (k2 / 2));
                k4 = h * func_value(x[i + 1], y[i] + k3);
                temp = y[i] + ((k1 + 2 * k2 + 2 * k3 + k4) / 6);
                y.push_back(temp);
                i++;
            }
            cout << "y(" << x_check << ")=" << y[i] << endl;
            break;
        case 2:
            goto xx;
            break;
        default:
            printf("Wrong input.\n");
        }
    }
}
int main(void)
{
    int order;
    int temp;
    while (1)
    {
        printf("1. 2nd order\n2. 4th order\n0. Exit program\n   Enter
your choice: ");
        cin >> order;
        switch (order)
        {
        case 0:
            return 0;
        case 1:
            temp = second_order();
            break;
        case 2:
            temp = forth_order();
            break;
        default:
            printf("Wrong input.\n");
        }
    }
}
```

Output:

```
"D:\2nd year odd sem\CSE 2104\Runge_Kutta_Method.exe"
1. 2nd order
2. 4th order
0. Exit program
   Enter your choice: 1
1. New value
2. New h
0. Terminate
   Enter choice: 1
Enter initial value of x: 0
Enter initial value of y: 2
Enter value of h: 0.01
Enter the value of x to find y: 0.1
y(0.1)=2.22628
1. New value
2. New h
0. Terminate
   Enter choice: 0
1. 2nd order
2. 4th order
0. Exit program
   Enter your choice: 2
1. New value
2. New h
0. Terminate
   Enter choice: 1
Enter initial value of x: 0
Enter initial value of y: 2
Enter value of h: 0.01
Enter the value of x to find y: 0.1
y(0.1)=2.22628
1. New value
2. New h
0. Terminate
   Enter choice: 0
1. 2nd order
2. 4th order
0. Exit program
   Enter your choice: 0

Process returned 0 (0x0)   execution time : 63.756 s
Press any key to continue.
```

Discussion:

This method is more efficient because it does not need to small value of h for more accurate result. It also possesses the advantage of requiring only the function values at some selected points on the subinterval.