Rajshahi University of Engineering & Technology

CSE 2104: Sessional Based on CSE 2103

Lab Report 09

Dated: 07.05.18

Submitted to

Shyla Afroge

Assistant Professor

Dept. of Computer Science & Engineering

&

Instructor, CSE 2104

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of Computer Science & Engineering

Problem#01: Numerical Solution of Ordinary Differential Equation by Taylor's Series

Theory: We consider the differential equation

$$y' = f(x, y)$$

with the initial condition

$$y(x_0) = y_0.$$

If y(x) is the exact solution of the differential eq. then the Taylor's series for y(x) around $x = x_o$ is given by

$$y(x) = y_0 + (x - x_0)y_0' + \frac{(x - x_0)^2}{2!} y_0'' + \cdots$$

If the values of y_o', y_o'' ... are known, then above eq. gives a power series for y. Using the formula for total derivatives, we can write

$$y'' = f' = f_x + y' f_y = f_x + f f_y,$$

where the suffixes denote partial derivatives with respect to the variable concerned. Similarly, we obtain,

$$\begin{aligned} y''' = f'' &= f_{xx} + f_{xy} f + f(f_{yx} + f_{yy} f) + f_y(f_x + f_y f) \\ &= f_{xx} + 2 f f_{xy} + f^2 f_{yy} + f_x f_y + f f_y^2 \end{aligned}$$

and other higher derivatives of y. The method can easily be extended to simultaneous and higher-order differential equations.

Here, the differential equation and the initial values considered in this example respectively

$$y' = x - y^2 \quad \text{and} \quad y(0) = 1.$$

and we determined the function value of y where x = 0.1

Solution:

```
#include <iostream>
#include <cmath>

using namespace std;

double _dy (double x, double y) {
    return x - y * y;
}

double _ddy (double x, double y) {
    return 1 - 2 * y * _dy(x, y);
}

double _d3y (double x, double y) {
    return - 2 * y * _ddy(x, y) - 2 * _dy(x, y) * _dy(x, y);
}

double _d4y (double x, double y) {
    return - 2 * y * _d3y(x, y) - 2 * _dy(x, y) * _ddy(x, y);
}

double _d5y (double x, double y) {
    return - 2 * y * _d4y(x, y) - 8 * _dy(x, y) * _d3y(x, y) - 6 * _ddy(x, y) *
    _ddy(x, y);
}

double result_Y (double new_x, double x, double y) {
    return 1 + new_x * _dy(x, y) + pow(new_x, 2) * _ddy(x, y) / 2 + pow(new_x, 3) *
    _d3y(x, y) / 6 + pow(new_x, 4) * _d4y(x, y) / 24 + pow(new_x, 4) * _d5y(x, y) /
    120;
}

int main() {
    double input_x, x, y;
    cout << "Enter input value to find desired output: ";
    cin >> input_x;
    cout << "Enter initial value of x and y respectively: ";
    cin >> x >> y;
    cout << result_Y(input_x, x, y);
}
```

OUTPUT:

```
Enter input value to find desired output: 0.1
Enter initial value of x and y respectively: 0 1
0.913623
```

Discussion: Though numerous functions are used here in the solution, this problem also could be solved using an array. This is also an efficient method than Euler method done later.

Problem#02: Numerical Solution of Ordinary Differential Equation by Euler's Method

Theory: To obtain numerical solution using Euler's method, we use the generalised equation

$$y_{n+1} = y_n + hf(x_n, y_n)$$

$$x_{n+1} = x_n + h$$

here the differential equation is f(x, y) and where n = 0, 1, 2, 3, …

Though, the algorithm is very simple for this method, it is a very efficient and covers less steps.

Solution:

```
#include <iostream>

using namespace std;

double func(double y) {
    return - y;
}
int main(){
    double x, y, h, input_x, temp;
    cout << "Enter input value to find desired output: ";
    cin >> input_x;
    cout << "Enter initial value of x and y respectively: ";
    cin >> x >> y;
    cout << "Enter the value of h: ";
    cin >> h;

    while(x != input_x){
        temp = h * func(y);
        y = y + temp;
        x = x + h;
    }
    cout << y << endl;
}
```

OUTPUT:

```
Enter input value to find desired output: 0.1
Enter initial value of x and y respectively: 0 1
Enter the value of h: 0.05
0 1 0.05 0.1
0.9025
```

Discussion: Hence, the solution is preety similar both cases, this method is somewhat inefficient with respect to the Taylor's Series Method for this function.