# EE1103: Numerical Methods

# Programming Assignment # 6

Nachiket Dighe, EE21B093

March 28, 2022

# Contents

# List of Figures

# List of Tables

# 1 Problem 1

Consider the following differential equation,

$$\frac{dy}{dt} = yt^3 - 1.5y$$

over the interval t = 0 to 2 (2). It is given that y(0) = 1. Solve the following parts:

1. Write the analytical solution for the differential equation.
2. Numerically solve the same by applying the following methods. Use step size $h = 0.1$, $0.25$ and $0.5$. For each value of h, plot the results from across ALL methods along with the true value, on the same graph.
   (a) Euler's Method
   (b) Heun's Predictor-Corrector Method
   (c) Midpoint Method
   (d) Fourth Order Runge-Kutta Method
3. Report the run time for each method (1).
4. For each value of h, tabulate the absolute error of y(2) for the different methods with respect to the true value. Infer the quality of the methods in terms of the error obtained. (Table should have h in the rows and the error from different methods as column entries; you can choose to use more values of h than estimated above, to prove your point).

## 1.1 Approach

In this problem, we use the *For Loops* and *Functions* to estimate the Function y(x) by various Methods.

## 1.2 Algorithm

The Pseudo-Code for Estimating the Function by using Various Methods are given in Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4.

**Algorithm 1:** Function Estimation using Euler Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 0.1$
for $i = 1$ to $2/h$ do
    $y_{i+1} \leftarrow y_i + f(x_i, y_i)h$
    $y_i \leftarrow y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

**Algorithm 2:** Function Estimation using Huen Predictor-Corrector Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 0.1$
for $i = 1$ to $2/h$ do
    $y_d \leftarrow y_i + f(x_i, y_i)h$
    $y_{i+1} \leftarrow y_i + (f(x_{i+1}, y_d) + f(x_i, y_i))\frac{h}{2}$
    $y_i \leftarrow y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

**Algorithm 3:** Function Estimation using Midpoint Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 0.1$
for $i = 1$ to $2/h$ do
    $y_{i+1/2} \leftarrow y_i + f(x_i, y_i)\frac{h}{2}$
    $y_{i+1} \leftarrow y_i + f(x_{i+1/2}, y_{i+1/2})h$
    $y_i \leftarrow y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

**Algorithm 4:** Function Estimation using Runge Kutta Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 0.1$
for $i = 1$ to $2/h$ do
    $k_1 = f(x_i, y_i)$
    $k_2 = (x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1)$
    $k_3 = (x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2)$
    $k_4 = (x_i + h, y_i + hk_3)$
    $y_{i+1} = y_i + (k_1 + 2k_2 + 2k_3 + k_4)\frac{h}{6}$
    $y_i = y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

### 1.3 Results

The analytical solution for the differential equation is:

$$\frac{dy}{dt} = yt^3 - 1.5y$$

$$\frac{dy}{y} = (t^3 - 1.5)dt$$

$$\int \frac{1}{y}\,dy = \int (t^3 - 1.5)\,dt$$

$$\log y = \frac{t^4}{4} - 1.5t + c$$

for x=0,y=1 we get c=0

$$y = e^{\frac{t^4}{4} - 1.5t}$$

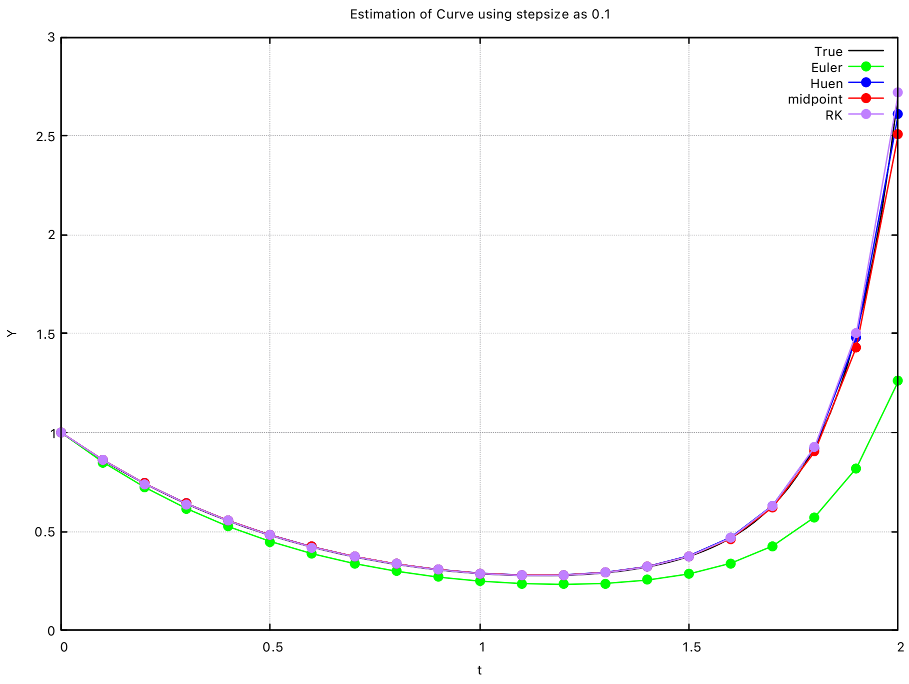

Figure 1: Estimation of Curve for stepsize 0.1.

As seen above, Fig 1 Estimates the Function for stepsize ($h = 0.1$), Fig 2 Estimates the Function for stepsize ($h = 0.25$) and Fig 3 Estimates the Function for stepsize ($h = 0.5$) for all the Four methods. Absolute Error as a Function of Stepsize for all the Four Methods is given in Fig 4.

The Absolute Error at y(2) for different stepsizes is given in Table 1.The Runtime for each Method for different Stepsizes is given in Table 2.

Table 1: Absolute Error for different Stepsizes.

| **h** | **Euler** | **Midpoint** | **Huen** | **Runge Kutta** |
|---|---|---|---|---|
| 1 | 2.968 | 1.139 | 1.827 | 0.848 |
| 0.5 | 2.604 | 1.126 | 0.833 | 0.204 |
| 0.25 | 2.188 | 0.689 | 0.397 | 0.035 |
| 0.1 | 2.022 | 0.542 | 0.299 | 0.017 |
| 0.02 | 1.458 | 0.211 | 0.0055 | 0.000264 |
| 0.01 | 0.237 | 0.0029 | 0.0012 | 0.000258 |
| 0.001 | 0.042 | 0.0170 | 0.01698 | 0.01697 |

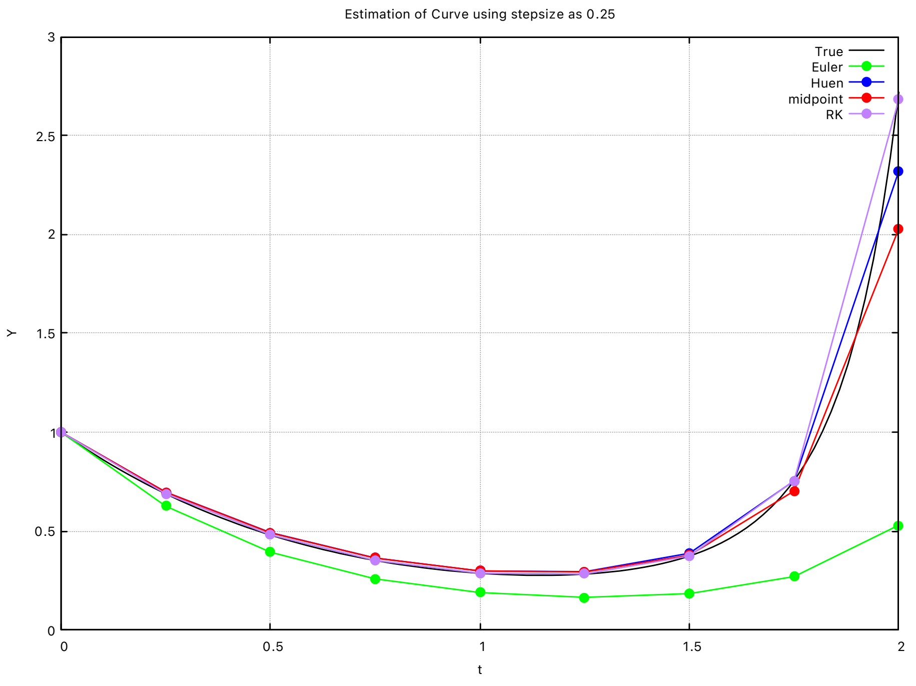

Figure 2: Estimation of Curve for stepsize 0.25.

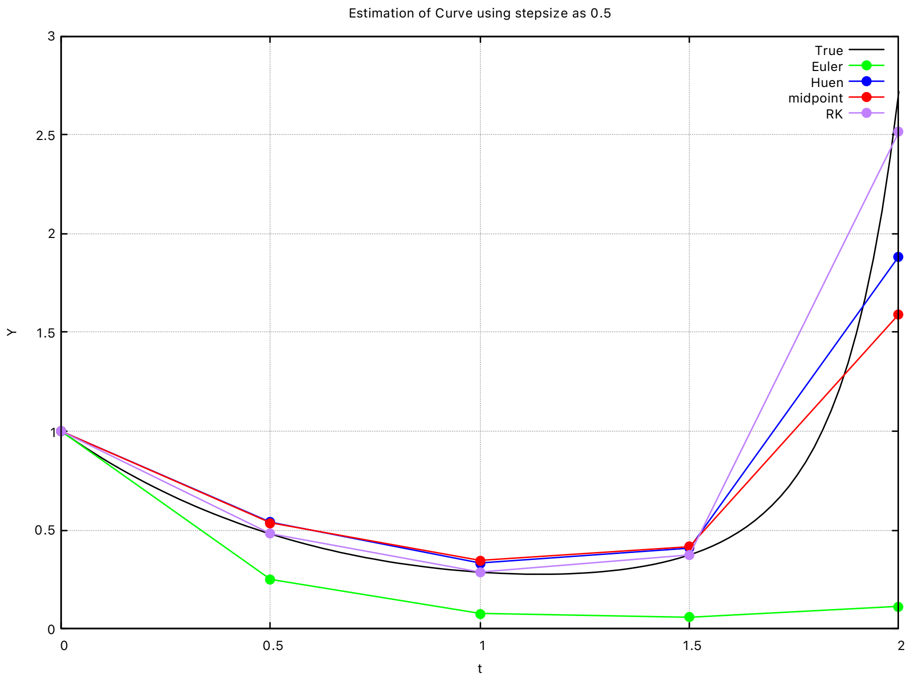

Figure 3: Estimation of Curve for stepsize 0.5.

Table 2: Runtime For all Methods in ($\mu$s)

| **h** | **Euler** | **Midpoint** | **Huen** | **Runge Kutta** |
|---|---|---|---|---|
| 0.1 | 84 | 104 | 120 | 95 |
| 0.25 | 60 | 70 | 84 | 64 |
| 0.5 | 48 | 57 | 63 | 54 |

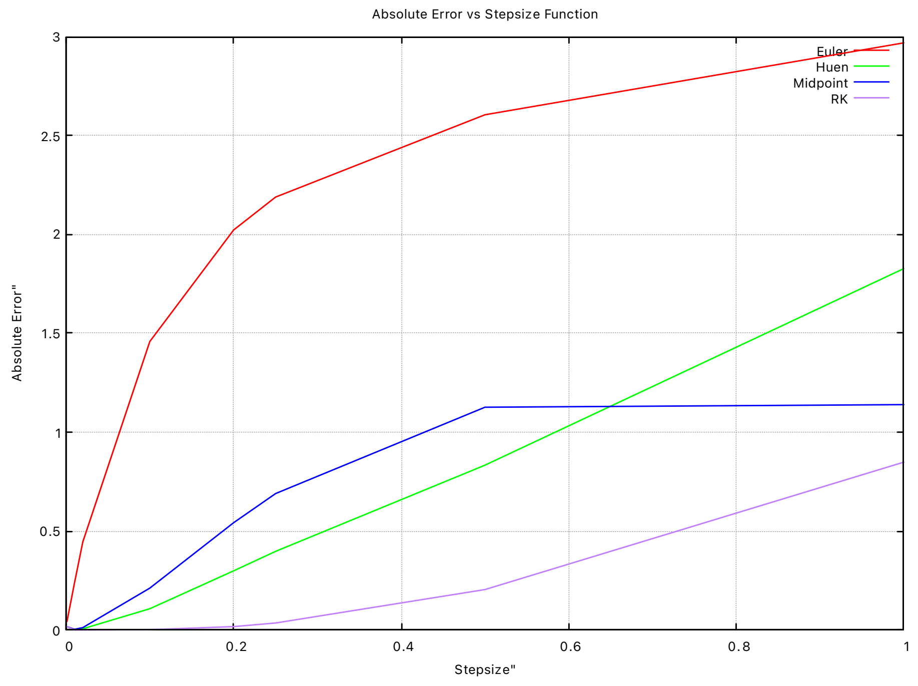

Figure 4: Absolute Error vs Stepsize.

## 1.4 Inferences

We deduce the following inferences in this experiment:

- In Estimation of the curve using Ordinary Differential Equation(ODE), we can see from Table 1 that the Efficiency of the methods is as follows:

$$Euler \leq Midpoint \leq Huen \leq RK$$

- A fundamental source of error in Euler's method is that the derivative at the beginning of the interval is assumed to apply across the entire interval.

- One method to improve the estimate of the slope involves the determination of two derivatives for the interval—`one at the initial point and another at the end point`. The two derivatives are then averaged to obtain an improved estimate of the slope for the entire interval. This approach is called **Huen's** Method.

- The midpoint method is superior to Euler's method because it utilizes a slope estimate at the *midpoint* of the prediction interval. Runge-Kutta (RK) methods achieve the accuracy of a *Taylor series* approach without requiring the calculation of higher derivatives. Thus, Fourth order Runge-Kutta Method is far efficient than the above methods.

- One more thing to conclude is that as we *decrease* the stepsize the Absolute Error also *decreases*, but for very small values of stepsize(h) error starts to increase. This is because the *approximations* that we took at the beginning start to add up.

## 1.5 Code

The code used for the experiments is mentioned in Listing 1

```c
#include<stdio.h>
#include<math.h>
#include<time.h>
#define trueval 2.718 //y(2) found analytically

//Function to return the value of differential f(x,y).
float diffunc(float t,float y)
{
    return y*(pow(t,3)-1.5);
}

//Estimating the Function using Euler method
void euler(float xi,float yi,float h)
{
    float ynew,errab;
    int i;
    for(i=1;i<=2/h;i++)
    {
        printf("%.2f \t\t %f\n",xi,yi);
        ynew=yi+diffunc(xi,yi)*h;
        yi=ynew;
        xi=xi+h;
    }
    errab=fabs(trueval-yi);
    printf("Absolute Error is: %f\n",errab);
}

//Estimating the Function using Huen's Predictor-Corrector method
void huen(float xi,float yi,float h)
{
    float ynew,yd,errab;//yn is dummy ynew
    int i;
    for(i=1;i<=2/h;i++)
    {
        printf("%.2f \t\t %f\n",xi,yi);
        yd=yi+diffunc(xi,yi)*h;
        ynew=yi+(diffunc(xi,yi)+diffunc(xi+h,yd))*h/2;
        yi=ynew;
        xi=xi+h;

    }
    errab=fabs(trueval-yi);
    printf("Absolute Error is: %f\n",errab);

```

```c
}

//Estimating the Function using Midpoint method
void mid(float xi,float yi,float h)
{
    float ynew,ymid,errab;//ymid is the estimation at the midpoint of
     ↪  xi and xi+1
    int i;
    for(i=1;i<=2/h;i++)
    {
        printf("%.2f \t\t %f\n",xi,yi);
        ymid=yi+diffunc(xi,yi)*h/2;
        ynew=yi+diffunc(xi+h/2,ymid)*h;
        yi=ynew;
        xi=xi+h;


    }
    errab=fabs(trueval-yi);
    printf("Absolute Error is: %f\n",errab);
}

//Estimating the Function using Runge Kutta method
void RK(float xi,float yi,float h)
{
    float ynew,k1,k2,k3,k4,errab;
    int i;
    for(i=1;i<=2/h;i++)
    {
        k1=diffunc(xi,yi);
        k2=diffunc(xi+h/2,yi+h*k1/2);
        k3=diffunc(xi+h/2,yi+h*k2/2);
        k4=diffunc(xi+h,yi+h*k3);
        printf("%.2f \t\t %f\n",xi,yi);
        ynew=yi+(k1+2*k2+2*k3+k4)*h/6;
        yi=ynew;
        xi=xi+h;

    }
     errab=fabs(trueval-yi);
     printf("Absolute Error is: %f\n",errab);
}


int main()
{
    float yold=1,xold=0,ynew,h=0.25;
    clock_t tStart = clock();

    printf("The Estimation of the function using Euler method
     ↪  is:\n");
    printf("X-value \t Y-values\n");
    euler(xold,yold,h);

    printf("\n\n");
    printf("The Estimation of the function using huen method is:\n");
    printf("X-value \t Y-values\n");
    huen(xold,yold,h);

    printf("\n\n");
    printf("The Estimation of the function using midpoint method
     ↪  is:\n");
    printf("X-value \t Y-values\n");
    mid(xold,yold,h);

    printf("\n\n");
    printf("The Estimation of the function using RK method is:\n");
    printf("X-value \t Y-values\n");
    RK(xold,yold,h);

    printf("Time taken: %.8fs\n", (double)(clock() -
     ↪  tStart)/CLOCKS_PER_SEC);
}
```

Listing 1: Estimation of a Curve using Runge Kutta Methods.

# 2 Problem 2

Solve

$$\frac{dy}{dt} = -100,000y + 99,999e^{-t}$$

over the interval from t = 0 to 2 using the following methods. Note that $y(0) = 0$.

1. Explicit Euler method, after estimating the step size required to maintain stability.
2. Implicit Euler method with a step size of 0.1.

## 2.1 Approach

In this problem, we use the *For Loops*,to calculate the function and print them to estimate the Curve for both methods.

## 2.2 Algorithm

The Pseudo-Code for Estimating the Function by using *Explicit* and *Implicit-Euler* methods are given in Algorithm 5 and Algorithm 6.

**Algorithm 5:** Function Estimation using Explicit Euler Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 10^{-5}$
for $i = 1$ to $2/h$ do
    $y_{i+1} \leftarrow y_i + f(x_i, y_i)h$
    $y_i \leftarrow y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

**Algorithm 6:** Function Estimation using Implicit Euler Method.

```
Inputs: $x_i, y_i, f(x_i, y_i)$
$h \leftarrow 0.1$
for $i = 1$ to $2/h$ do
    $y_{i+1} \leftarrow y_i + f(x_{i+1}, y_{i+1})h$
    $y_i \leftarrow y_{i+1}$
    $x_i \leftarrow x_i + h$
end
$Err_{abs} = |y_{true} - y_i|$
print $x_i, y_i$
```

## 2.3 Results

The Estimation of the Curve using Both Explicit and Implicit Euler Methods are shown in Fig. 5 and Fig. 6. The True Curve obtained from analytical method is:

$$y = e^{-t} - e^{-10^5 t}$$

In this Problem , I have taken $h = 10^{-5}$ for Explicit Euler method(Reason is mentioned in Inference). The Absolute Error(%) for both methods is given below:

$$Explicit = 6.07 \times 10^{-4}\%$$

$$Implicit = 4.67 \times 10^{-4}\%$$

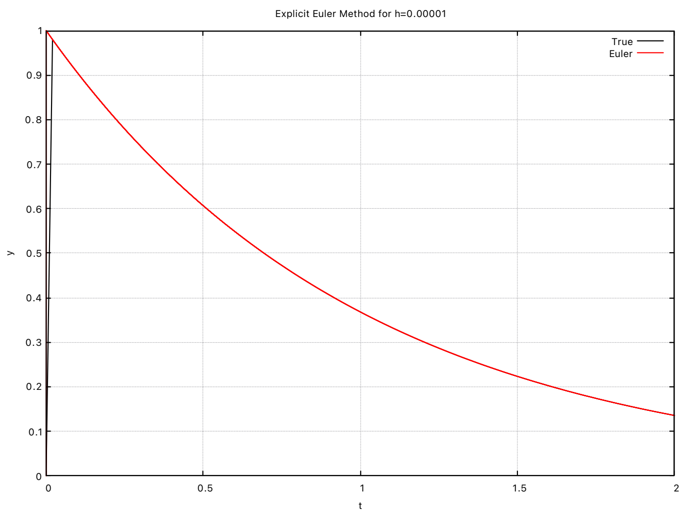

Figure 5: Estimation of Curve for Explicit Euler Method.

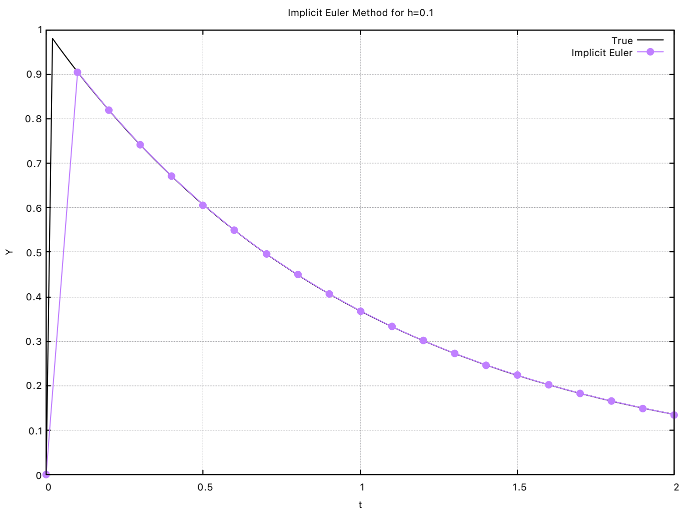

Figure 6: Estimation of Curve for Implicit Euler method.

## 2.4 Inference

We deduce the following inferences in this experiment:

- For differential Equations like $\boldsymbol{\frac{dy}{dt} = -ay}$ the factor which decides the stability of the curve is **a** and stepsize(**h**) and the condition for stability is $\boldsymbol{h < \frac{2}{a}}$.
- By taking $\boldsymbol{h < \frac{2}{a}}$, we can estimate the Function using Explicit Euler Method.
- Implicit methods offer an alternative remedy. Such representations are called implicit because the unknown appears on both sides of the equation. An implicit form of Euler's method can be developed by evaluating the derivative at the future time.
- In this Problem, we can see that the dominant part of the Differential Equation is $\boldsymbol{-10^5 y}$. Thus, for Value of $h < 2 \times 10^{-5}$ the curve will be stable, But for other values of h,the Curve obtained will fluctuate between $-\infty$ to $\infty$.

## 2.5 Code

The code used for the experiments are mentioned in Listing 2 and Listing 3.

```c
#include<stdio.h>
#include<math.h>
#include<time.h>
#define trueval 0.13534 //y(2) found analytically
//Function to return the value of differential f(x,y).
double diffunc(double t,double y)
{
    return -pow(10,5)*y+99999*exp(-t);
}

//Estimating the Function using Euler method
void euler(double xi,double yi,double h)
{
    double ynew,errab;
    int i;
    for(i=0;i<=2/h+1;i++)
    {
        printf("%.5lf \t\t %lf\n",xi,yi);
        ynew=yi+diffunc(xi,yi)*h;
        yi=ynew;
        xi +=h;


    }
    errab=fabs(trueval-yi);
    printf("Absolute Error is:%.8f\n",errab);
}
```

```c

int main()
{
    double yold=0,xold=0,ynew,h=pow(10,-5);
    clock_t begin = clock();


    printf("The Estimation of the function using Euler method
     ↪  is:\n");
    printf("X-value \t Y-values\n");
    euler(xold,yold,h);

    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("Time taken: %.8fs\n",time_spent);

}
```

Listing 2: Estimation of Curve using Explicit Euler Method.

```c
#include<stdio.h>
#include<math.h>
#include<time.h>
#define trueval 0.13534 //y(2) found analytically
//Function to return the value of differential f(x,y).
float impfunc(float t,float y,float h)
{
    return (y+99999*exp(-t-h)*h)/(1+pow(10,5)*h);
}

//Estimating the Function using Euler method
void imp_euler(float xi,float yi,float h)
{
    float ynew,errab;
    for(int i=0;i<=2/h;i++)
    {
        printf("%.2f \t\t %f\n",xi,yi);
        if(i==2/h)
            break;
        ynew=impfunc(xi,yi,h);
        yi=ynew;
        xi=xi+h;


    }
    errab=fabs(trueval-yi)*100;
    printf("Absolute Error is:%f\n",errab);
}

int main()
{
     float yold=0,xold=0,h=0.1;
    clock_t begin = clock();

    printf("The Estimation of the function using Implicit Euler
     ↪  method is:\n");
    printf("X-value \t Y-values\n");
    imp_euler(xold,yold,h);

    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("Time taken: %.8fs\n",time_spent);

}
```

Listing 3: Estimation of Curve using Implicit Euler Method.