Numerical Solutions of Ordinary Differential Equations: Euler's Method

Course: MPHYCC-05 Modeling and Simulation, MPHYCC-09 Lab-II
(M.Sc. Sem-II)

By

Dr. Sanjay Kumar

Assistant Professor
Department of Physics

Patna University

Contact Details: Email- sainisanjay35@gmail.com;
Contact No.- 9413674416

Unit-3 of CC-05

<u>Solving Ordinary Differential Equations Numerically</u>

Most of the differential equations, representing physical systems, are non-linear and, hence, not analytically solvable. Although there are numerous techniques for finding the analytic solution of first order differential eqautions, we are unable to easily obtain closed form (i.e., a representation of the solution in terms of a finite number of simple functions; not like power series solutions having infinite terms) analytic solutions for many equations. Then numerical methods become nessessity. First we will discuss the underlying concepts which are common to most of numerical methods for solving ODEs.

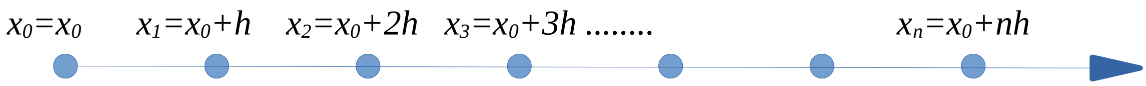
For the purpose, we consider a general first order ODE

$$\frac{dy}{dx}=f(x,y) \quad \text{with the initial condition} \quad y(x=x_0)=y_0 \quad \textit{find } y(x) \textit{ for } x_0<x<L \quad \text{----}(1)$$

Now, our aim is to solve the ODE (i.e. calculate the values of unknown variable y at known x values) with the given initial condition. One of the simple numerical method to solve such an equation is the **Euler's method.**

<u>EULER'S METHOD</u>

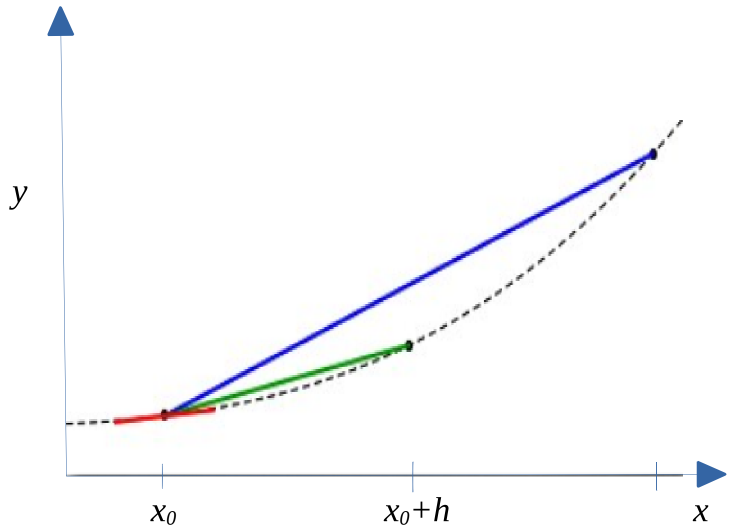
To discuss the method, we note that, in reality, the variables x and y are continuous. Therefore, in principle, we have to calculate the values of y at infinite values (points) of x, which is not possible by computers. As a result, in all the numerical methods, we only calculate the values of y at the finite values of x. For example, suppose we want to find the values of y at $x_i=x_0+ih$ *(where $i=1,2,3....n$).* This process of replacing infinite values of x with finite values of x is called **descretization of the domain.**
To further illustrate it, in the following, we show a figure.



The figure shows the descretization of the domain [x_0,L] into $n+1$ evenly spaced points x_i *where* $x_i=x_0+ih$ *(where $i=1,2,3...n$).* The value of y at the first point x_0 is known to us by the initial condition. Also note that the last point is $x_n=x_0+nh=L$ *and* the spacing between two adjacent points (also known as "grid-spacing")

$$h=\frac{(x_n-x_0)}{n}=\frac{(L-x_0)}{n} \qquad \text{------ (2)}$$

Clearly, the grid-spacing h decreases as we increase n (which also known as "grid-resolution").



Let's assume that the dashed curve in the above figure represents the actual solution $y(x)$ between $x_0 < x < x_0+h$. Now we draw a secant line between $x=x_0$ and $x=x_0+h$ on the curve. For different h values, we get different secant lines shown in color blue, green and red in the above figure. It is evident that for a smaller value of h, the secant line approximately represents the tangent of the curve at $x=x_0$ point. Therefore, for a small value of h,

$$\frac{dy}{dx}(x=x_0) \approx \frac{y(x=x_0+h)-y(x=x_0)}{h} \qquad \text{-------------- (3)}$$

From equation (1), the slope (dy/dx) at $x=x_0$ is $f(x_0, y_0)$. Put this in equation (3) we get,

$$y(x=x_0+h)=y(x=x_0)+h\,f(x_0,y_0)$$
$$y(x=x_1)=y_0+h\,f(x_0,y_0)$$
$$y_1=y_0+h\,f(x_0,y_0) \qquad \text{---------------- (4)}$$

If we do the similar procedure for the interval $x_0+h < x < x_0+2h$ (i.e. $x_1 < x < x_2$), we can obtain

$$y_2=y_1+h\,f(x_1,y_1)$$

The formula can be generalized for the interval $x_0+ih < x < x_0+(i+1)h$ (i.e. $x_i < x < x_{i+1}$) *as*:

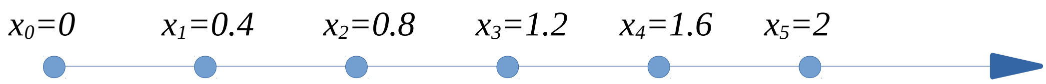
$$y(i+1)=y(i)+h\,f(x(i),y(i)) \qquad \text{---------------- (5)}$$

<u>Example:</u> *Solve the differential equation*

$$\frac{dy}{dx}=-y \qquad \text{---------------- (6)}$$

find y for x ε [0, 2] with the initial condition $y(x=0)=y_0=1$.

For solving the equation, first we descretize the domain $x \in [0, 2]$.
If we take $n=5$ then $h=(2-0)/5=0.4$



From the comparison of equations (1) and (6), we know

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

Using equation (5) with $h=0.4$,

$$y(x_1=0.4)=y_1=y_0+h\,f(x_0,y_0)$$
$$y_1=y_0+h\,(-y_0)$$
$$y_1=1+0.4\,(-1)=0.6$$

$$y(x_2=0.8)=y_2=y_1+h\,f(x_1,y_1)$$
$$y_2=y_1+h\,(-y_1)$$
$$y_2=0.6+0.4\,(-0.6)=0.36$$

$$y(x_3=1.2)=y_3=y_2+h\,f(x_2,y_2)$$
$$y_3=y_2+h\,(-y_2)$$
$$y_3=0.36+0.4\,(-0.36)=0.216$$

$$y(x_4=1.6)=y_4=y_3+h\,f(x_3,y_3)$$
$$y_4=y_3+h\,(-y_3)$$
$$y_4=0.216+0.4\,(-0.216)=0.1296$$

$$y(x_5=2.0)=y_5=y_4+h\,f(x_4,y_4)$$
$$y_5=y_4+h\,(-y_4)$$
$$y_5=0.1296+0.4\,(-0.1296)=0.07776$$

Note that the exact solution of the differential equation (6) is $y=e^{-x}$. Now calculate the values of y at $x=0.4, 0.8, 1.2, 1.6, 2$.

$$y(0.4)=0.6703$$
$$y(0.8)=0.4493$$
$$y(1.2)=0.3011$$
$$y(1.6)=0.2019$$
$$y(2)=0.1353$$

If we compare the exact values of y with the numerically calculated values (y_1, y_2, y_3, y_4, y_5) using the Euler's method, we found that there is difference between the exact values and the numerical values. The reason for the difference is the errors arise in the numerical calculations due to the descretization of the domain and rounding-off the numbers. From the comparison, we can see that the contributation of the errors leads to a large deviation of solutions obtained from the Euler's method from the exact ones and, therefore, for practical purposes, the method is not used. Instead, another more accurate method, known as "Runge-Kutta Methods", is generally used.

Algorithm to Write a Program of the Euler's method

Problem: $\frac{dy}{dx}=f(x,y)$ *with the initial condition* $y(x=x_0)=y_0$ *find y(x) for x_0 <x<L*

1. *Input x_0, L, y_0, n*

2. *h=(x_n-x_0)/n*

3. *Do iteraction (i=1,n)*
   {x_1=x_0+h;
   s=f(x_0, y_0)
   *y_1=y_0+h*s*
   write x_1, y_1
   y_0=y_1
   x_0=x_0 }

4. *end*

C- Program of the Euler's method

Problem: $\frac{dy}{dx}=-y$ *with the initial condition* $y(x=0)=1$ *find y(x) for 0 <x<2*

```
#include <stdio.h>
#include <math.h>
int main()
{float f, x0, l, y0, h, x1, y1;
int n,i;
printf("enter the value of n \n");
scanf("%d",&n);
printf("enter the initial point x0, last point L and initial condition y0:\n");
scanf("%f %f %f",&x0,&l,&y0);
h=(l-x0)/n;

for(i=1;i<=n;i++)
{x1=x0+h;
f=-y0;
y1=y0+h*f;
printf("x[%d] and y[%d]:%f\t\t%f \n",i,i,x1,y1);
x0=x1;
y0=y1;}

return 0;}
```

Output of the program:

```
enter the value of n
5
enter the initial point x0, last point L and initial condition y0:
0 2 1
x[1] and y[1]:0.400000          0.600000
x[2] and y[2]:0.800000          0.360000
x[3] and y[3]:1.200000          0.216000
x[4] and y[4]:1.600000          0.129600
x[5] and y[5]:2.000000          0.077760
```