Physics through Computational Thinking

Euler's Method

Auditya Sharma and Ambar Jain
Dept. of Physics, IISER Bhopal

Outline

In this video you will

learn about Euler's Method of solving an ordinary differential method and learn to implement it on the computer.

Euler’s Method

- It’s not possible to solve all problems involving differential equations analytically. So we will learn to solve the initial value problems numerically through ***Euler’s method***, one of the first and simplest methods of this variety.
- Let’s say your differential equation is given by

$$\dot{x}(t) = f(x, t) \tag{1}$$

- Differentially, you can write this equation as (which is the inspiration of the Euler’s method)

$$dx = f(x, t)\, dt \tag{2}$$

- We are interested in finding the solution of this equation between times t_i and t_f.
- In order to solve it numerically we will discretize the time into a grid N intervals and $N+1$ time instants. $t_i = t_0$ being initial time and $t_f = t_N$ being the final time.

$$t_0, t_1, t_2, \ldots, t_N \tag{3}$$

- Step size h is defined as,

$$h \equiv \Delta t = t_n - t_{n-1} = \left(t_f - t_i\right)/N \tag{4}$$

- According to Euler’s method

$$\begin{gathered} x_0 = x_i \\ x_{n+1} = x_n + h\, f(x_n, t_n) \end{gathered} \tag{5}$$

- This successively determines all the x_n.

For Loop

- We will use the **For** loop to implement Euler's method in Mathematica. **For** loop has the following structure

$$\text{For}[\textit{intialization},\ \textit{condition},\ \textit{increment},\ \textit{body}] \tag{6}$$

- This means execute the *body* of the **For** loop starting from *initialization* until the *condition* holds to be true, executing the *increment* after executing the *body*.
- Here is an example of **For** loop to produce a list of squares.

```
array1 = {};
For[n = 1, n ≤ 10, n++, AppendTo[array1, n^2]]
array1
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

- As you are familiar, same can also be accomplished with **Table**

```
array2 = Table[n^2, {n, 1, 10}]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

- Typically **Table** is faster and more efficient than **For**. We should use **For** only as a last resort. In implementation of Euler method, Table won't quite work as we need to depend upon previous iteration for the current one.