

Group 5: Euler's Method

March 24, 2023

Ashraf Mohammed Hassan Anil: *SCT211-0255/2021*

Yunus Mohamed Abdi: *SCT211-0720/2021*

Yusin Ali Adan: *SCT211-0655/2021*

Description of Euler's Method

Euler's Method is a numerical method used in numerical analysis for solving ordinary differential equations (ODEs) with a given initial value. It is named after the mathematician Leonhard Euler who developed the method in the 18th century.

The basic idea of Euler's method is to approximate the solution to the ODE by using the tangent line at the initial point to make a local linear approximation of the solution. Then, the solution at a small time increment from the initial point is estimated by following the tangent line for that increment.

The points are computed iteratively as follows:

$$y_{n+1} = y_n + h * f(x_n, y_n)$$

where h is the time step size and $x_{n+1} = x_n + h$.

1. Given an initial value problem of the form $y' = f(x, y)$, $y(x_0) = y_0$, choose a time step size h .
2. Set $x = x_0$ and $y = y_0$ as the initial values.
3. Use Euler's method to compute the values y_{n+1} for $n = 0, 1, \dots, n - 1$:

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

$$y_{n+1} = y_n + h * f'(x_n, y_n)$$

4. Repeat step 3 until you reach the desired endpoint or number of time steps.
5. Plot the graph to visualize the solution

Solved example

Question: Given that $y = x^2$ and the initial points are $p(1, 1)$ with step-size $h = 0.2$. Obtain $y(2.0)$

Solution:

$n = (y_n - y_0) \text{ divide by } h$

$n = (2.0 - 1)/0.2$ therefore: $n = 5$. The values of x_n and y_n will be:

n	x_n	y_n	Accurate
0	1	1	1
1	1.2	1.48	1.44
2	1.4	2.04	1.96
3	1.6	2.68	2.56
4	1.8	3.40	3.24
5	2.0	4.16	4.00

Table: Values of x_n and y_n

Here's an example implementation of Euler's method in Python:

```
import matplotlib.pyplot as plt
```

```
def f(x):  
    return x ** 2
```

```
def df(x):  
    return 2 * x
```

```
def euler_method(f_prime, x0, y0, h, num_steps):
```

```
    x = [x0]  
    y = [y0]
```

```
for i in range(num_steps):  
    x_i = x[i]    # Get the current time  
    y_i = y[i]    # Get the current solution value  
    y_next = y_i + h * f_prime(x_i)  
  
    x_next = x_i + h  
  
    x.append(x_next)  
    y.append(y_next)  
  
return x, y
```

```
x_initial = 1
y_initial = 1
size_step = 0.2
n = 5
```

```
x_values, y_values = euler_method(df, x_initial, y_initial,
size_step, n)
```

```
# Getting the error
y_euler = y_values[-1]
y_accurate = f(x_values[-1])
```

```
print("x values:", x_values)
print("y values:", y_values)
print(f"Error = {y_accurate-y_euler}")
```

```
x_real = [1, 1.1, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9]  
y_real = [num ** 2 for num in x_values]
```

```
plt.plot(x_values, y_values, color='red',  
label="Approximation with Euler's method")
```

```
plt.plot(x_values, y_real, color='blue',  
label='Accurate curve')
```

```
plt.legend()  
plt.xlabel('x-axis')  
plt.ylabel('y-axis')  
plt.grid()  
plt.show()
```


Explanation of the output of the code

When the above code is run, the output is as follows:

x values: [1, 1.2, 1.4, 1.5999999999999999,
1.7999999999999998, 1.9999999999999998]

y values: [1, 1.4, 1.88, 2.44, 3.08, 3.8]

Error = 0.19999999999999993

Same code output but with h increased to 0.4

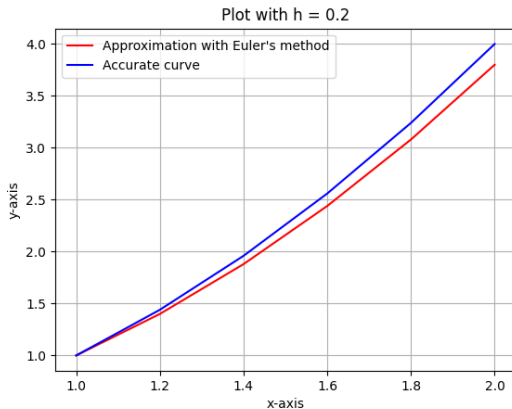
x values: [1, 1.4, 1.7999999999999998,
2.1999999999999997, 2.5999999999999996,
2.9999999999999996]

y values: [1, 1.8, 2.92, 4.359999999999999,
6.119999999999999, 8.2]

Error = 0.7999999999999998

Conclusion: Euler's method gives more accuracy if the size step h is lower and number of steps n is lower

Euler's Plot with $h = 0.2$



Euler's Plot with $h = 0.4$

