ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

MÉTODOS NUMÉRICOS

Tarea 12:
ODE Método de Euler

NOMBRE:

Marck Hernández

PROFESOR: Ing. Jonathan A. Zea

FECHA DE ENTREGA: 15-08-2024

Tarea 12 - ODE Método de Euler

```
%load_ext autoreload
import numpy as np
import math
from src import ODE_euler, graphics, ODE_euler_nth
```

Conjunto de Ejercicios

1. Use el método de Euler para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

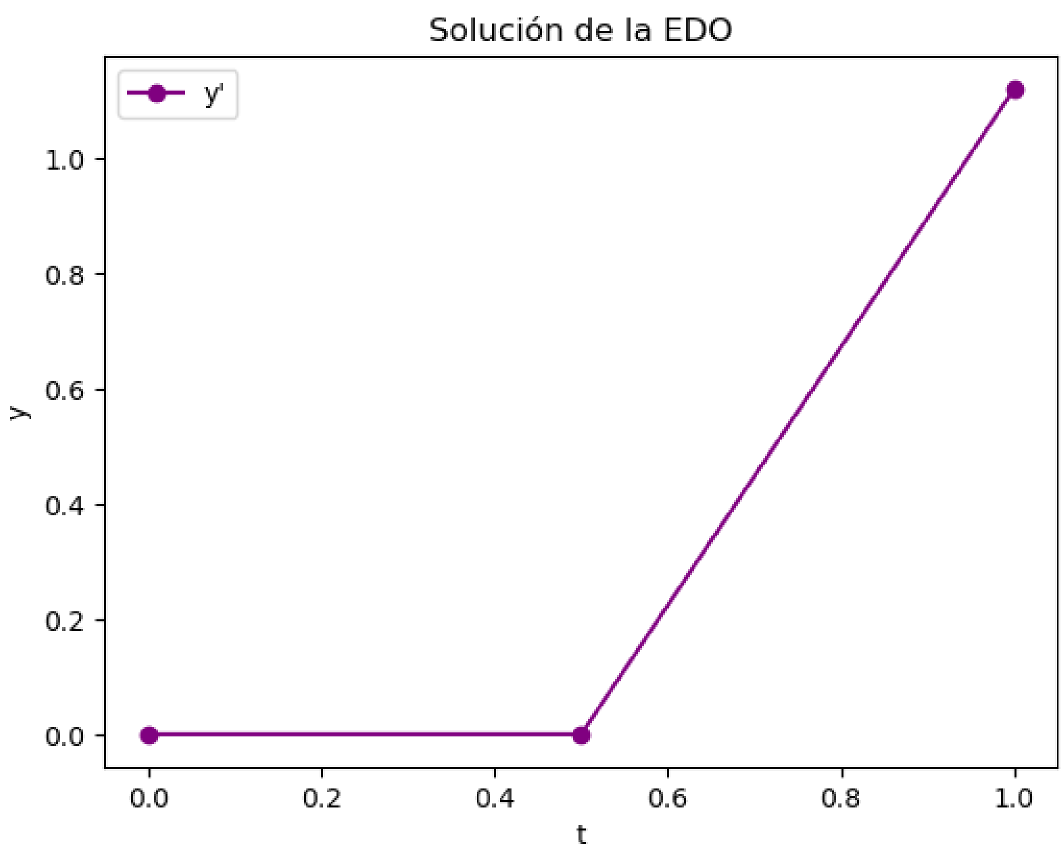
a) $y' = te^{3t} - 2y$, $0 \leq t \leq 1$, $y(0) = 0$, con $h = 0.5$

```
%autoreload 2
y_der = lambda t, y: t*math.exp(3*t) - 2*y
y_init = 0

ys_a, ts_a, h = ODE_euler(a = 0, b = 1, f = y_der, y_t0 = y_init, N = 2)

print(f"El valor de h es: {h}")
graphics(ts_a, ys_a, "PURPLE")
```

```
El valor de h es: 0.5
```



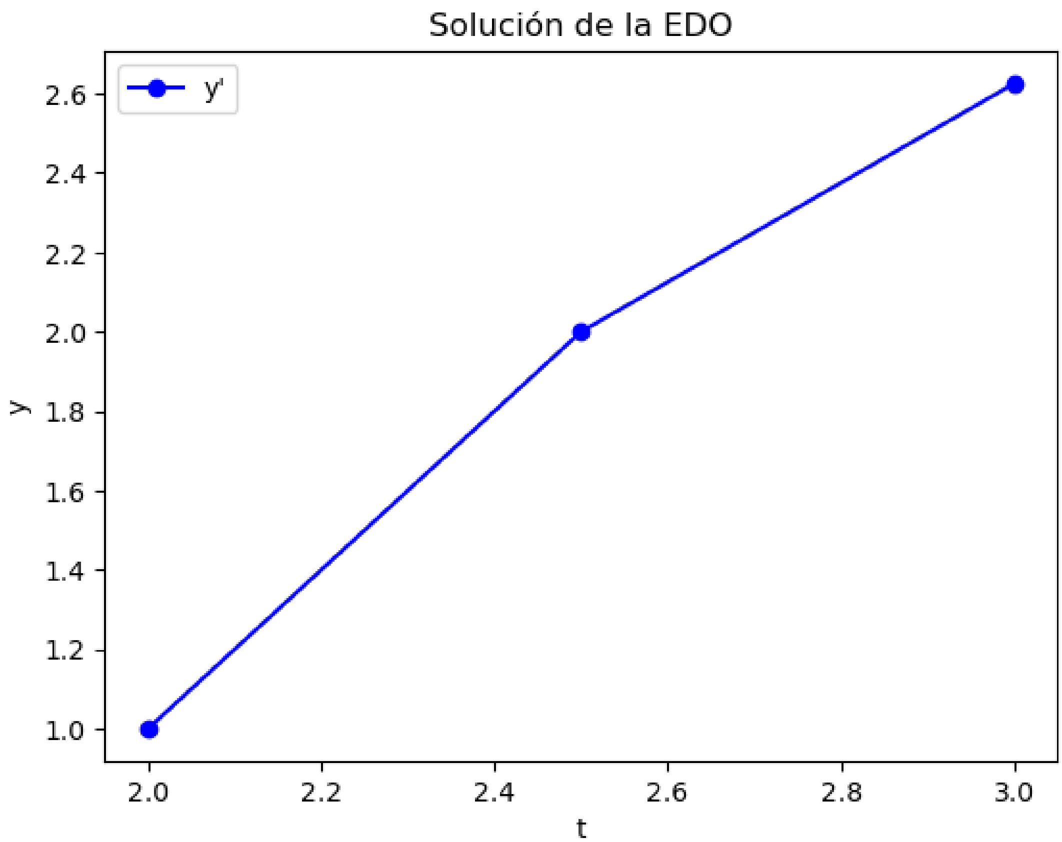
b) $y' = 1 + (t - y)^2$, $2 \leq t \leq 3$, $y(2) = 1$, con $h = 0.5$

```python
%autoreload 2
y_der = lambda t, y: 1 + (t - y)**2
y_init = 1

ys_b, ts_b, h = ODE_euler(a = 2, b = 3, f = y_der, y_t0 = y_init, N = 2)

print(f"El valor de h es: {h}")
graphics(ts_b, ys_b,"BLUE")
```

```
El valor de h es: 0.5
```



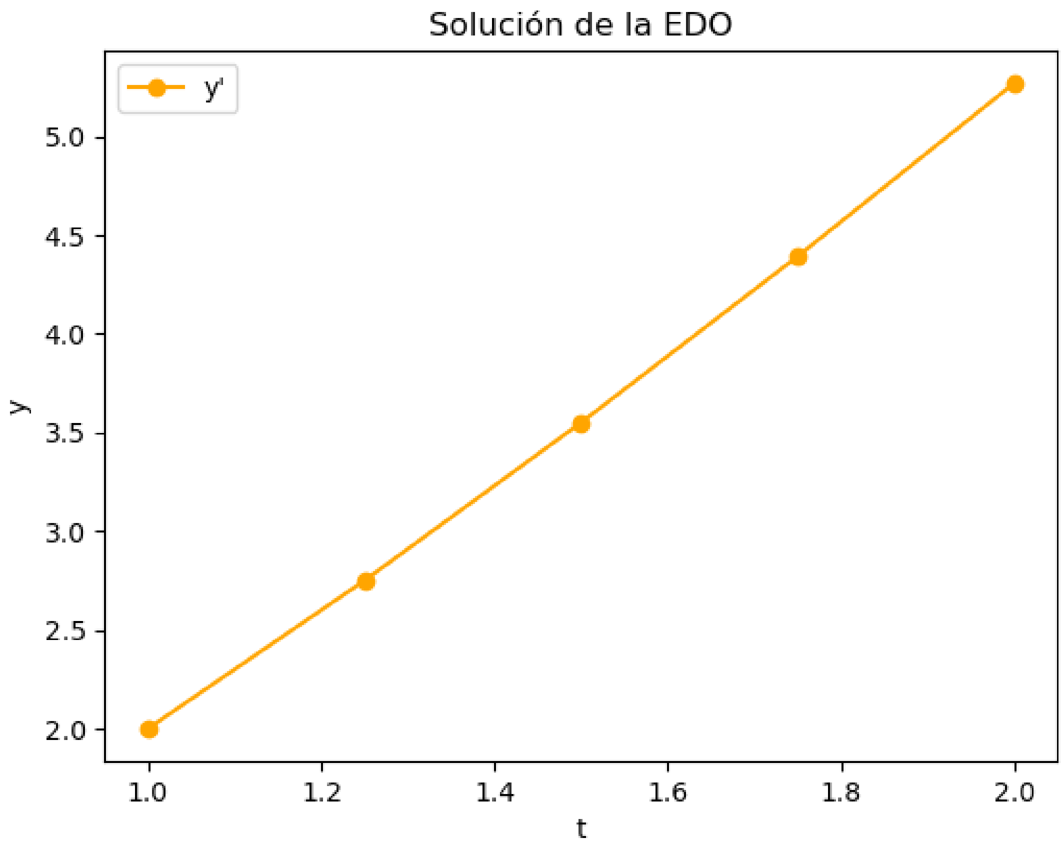
c) $y' = 1 + \frac{y}{t}$, $1 \leq t \leq 2$, $y(1) = 2$, con $h = 0.25$

```python
%autoreload 2
y_der = lambda t, y: 1 + y/t
y_init = 2

ys_c, ts_c, h = ODE_euler(a = 1, b = 2, f = y_der, y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_c, ys_c,"ORANGE")
```

```
El valor de h es: 0.25
```



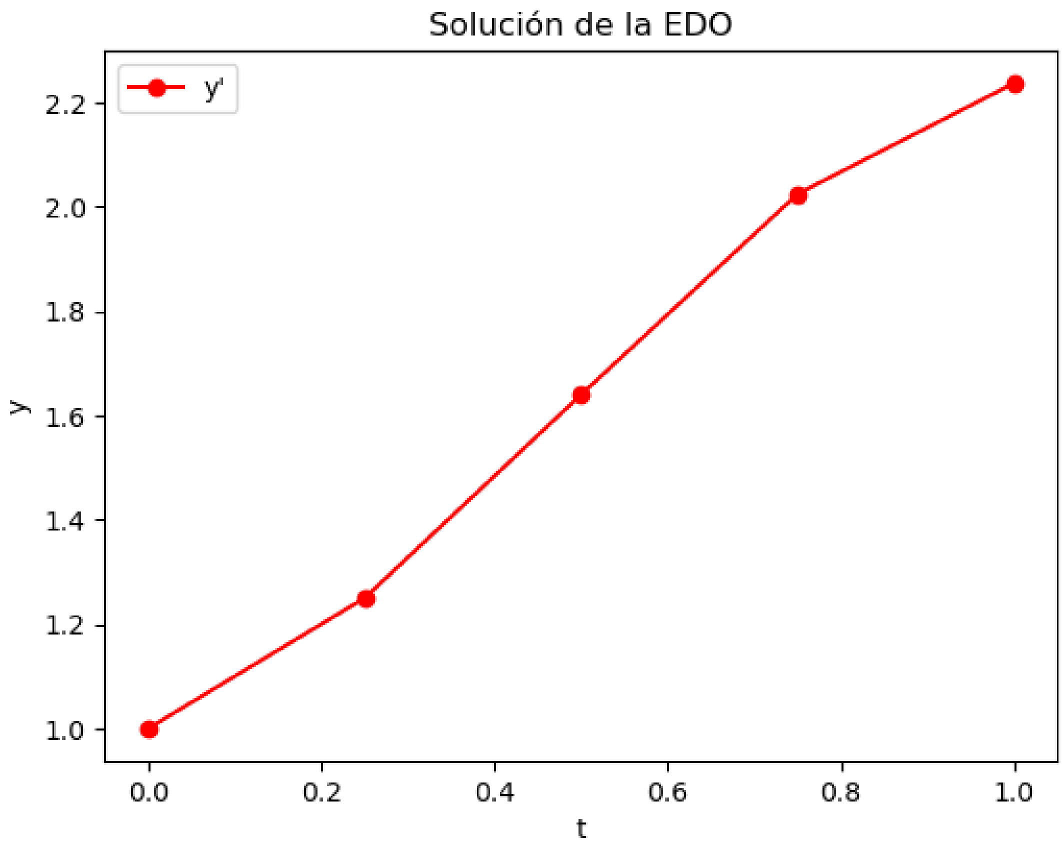
d) $y' = \cos 2t + \sin 3t$, $0 \leq t \leq 1$, $y(0) = 1$, con $h = 0.25$

```
%autoreload 2
y_der = lambda t, y: math.cos(2*t) + math.sin(3*t)
y_init = 1

ys_d, ts_d, h = ODE_euler(a = 0, b = 1, f = y_der, y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_d, ys_d,"RED")
```

```
El valor de h es: 0.25
```



2. Las soluciones reales para los problemas de valor inicial en el ejercicio 1 se proporcionan aquí. Compare el error real en cada paso.

a) $y(t) = \frac{1}{5}te^{3t} - \frac{1}{25}te^{3t} + \frac{1}{25}te^{-2t}$

```
%autoreload 2
def y(t):
    return 1/5*t*math.exp(3*t) - 1/25*t*math.exp(3*t) + 1/25*t*math.exp(-2*t)

# Calculamos y(t) para todos los valores en ts_a y luego evaluamos el error
try:
    # Si y(t) no es cero, calculamos el error real
    y_values = [y(t) for t in ts_a]
    if all(val != 0 for val in y_values):
        errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_a, ts_a)])
        print(f"El error real es: {errorReal}")
    else:
        print("NO ES POSIBLE REALIZAR UNA DIVISIÓN POR CERO")
except Exception as e:
    print(f"Se produjo un error: {e}")
```

```
NO ES POSIBLE REALIZAR UNA DIVISIÓN POR CERO
```

b) $y(t) = t + \frac{1}{1-t}$

```
%autoreload 2
def y(t):
    return t + 1/(1 - t)

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_b, ts_b)])
print(f"El error real es: {errorReal}")
```

```
```

```
El error real es: 0.04696969696969694
```

c) $y(t) = t \ln t + 2t$

```
%autoreload 2
def y(t):
    return t * math.log(t) + 2*t

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_c, ts_c)])
print(f"El error real es: {errorReal}")
```

```
El error real es: 0.013575458924045315
```

d) $y(t) = \frac{1}{2}\sin 2t - \frac{1}{3}\cos 3t + \frac{4}{3}$

```
%autoreload 2
def y(t):
    return 1/2*math.sin(2*t) - 1/3*math.cos(3*t) + 4/3

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_d, ts_d)])
print(f"El error real es: {errorReal}")
```

```
El error real es: 0.035265188624637164
```

3. Utilice el método de Euler para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

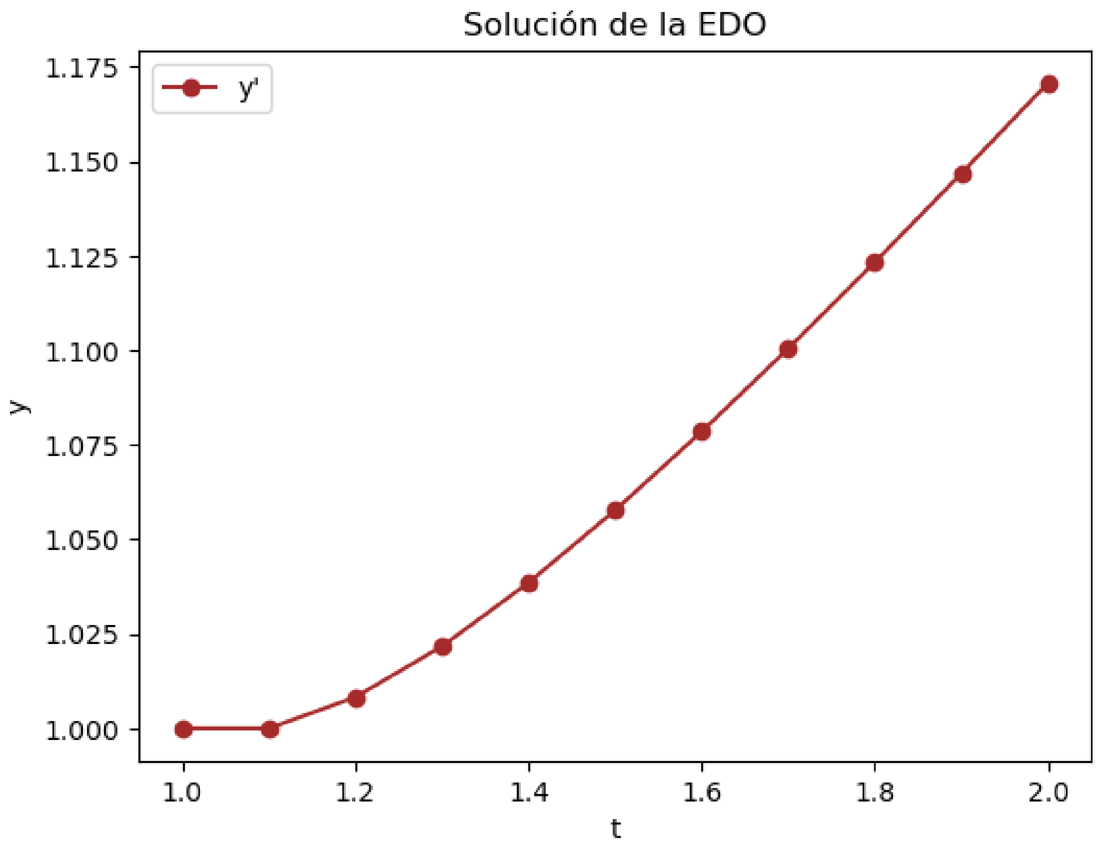
a) $y' = \frac{y}{t} - (\frac{y}{t})^2$, $1 \leq t \leq 2$, $y(1) = 1$, con $h = 0.1$.

```
%autoreload 2
y_der = lambda t, y: y/t - (y/t)**2
y_init = 1

ys_a2, ts_a2, h = ODE_euler(a = 1, b = 2, f = y_der, y_t0 = y_init, N = 10)

print(f"El valor de h es: {h}")
graphics(ts_a2, ys_a2,"BROWN")
```

```
El valor de h es: 0.1
```



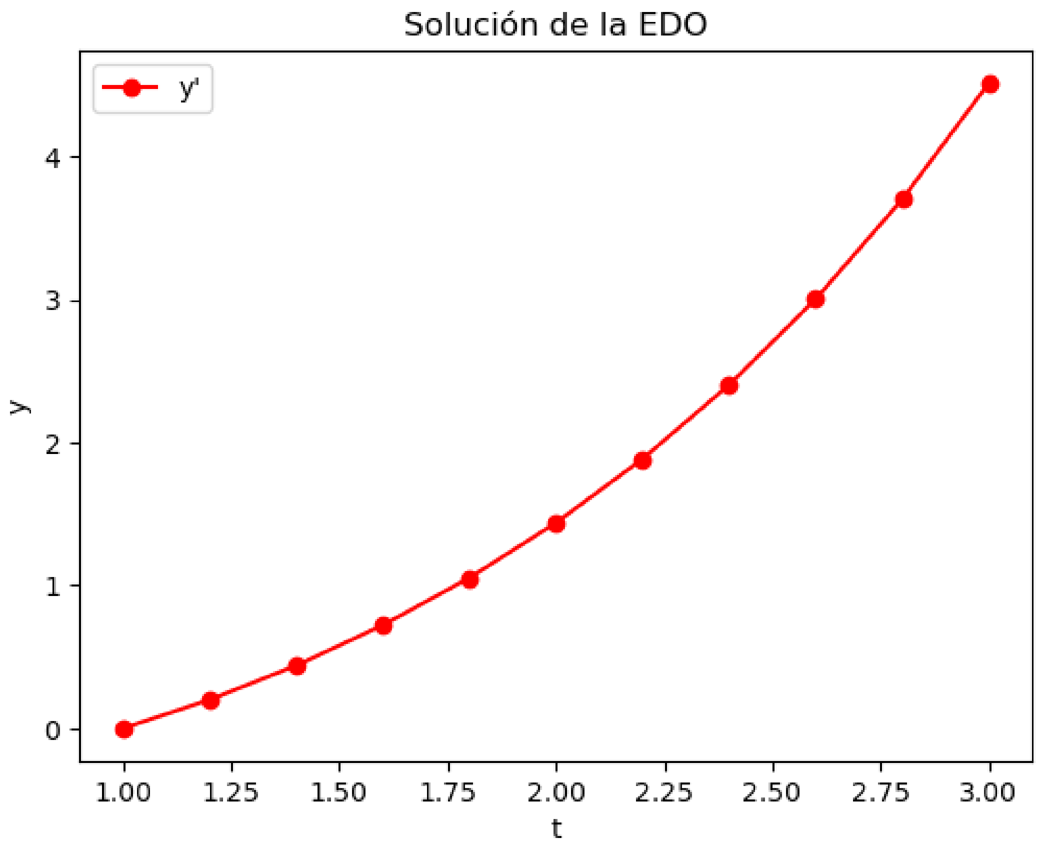
b) $y' = 1 + \frac{y}{t} + (\frac{y}{t})^2$, $1 \le t \le 3$, $y(1) = 0$, con $h = 0.2$.

```
%autoreload 2
y_der = lambda t, y: 1 + y/t + (y/t)**2
y_init = 0

ys_b2, ts_b2, h = ODE_euler(a = 1, b = 3, f = y_der, y_t0 = y_init, N = 10)

print(f"El valor de h es: {h}")
graphics(ts_b2, ys_b2,"RED")
```

```
El valor de h es: 0.2
```



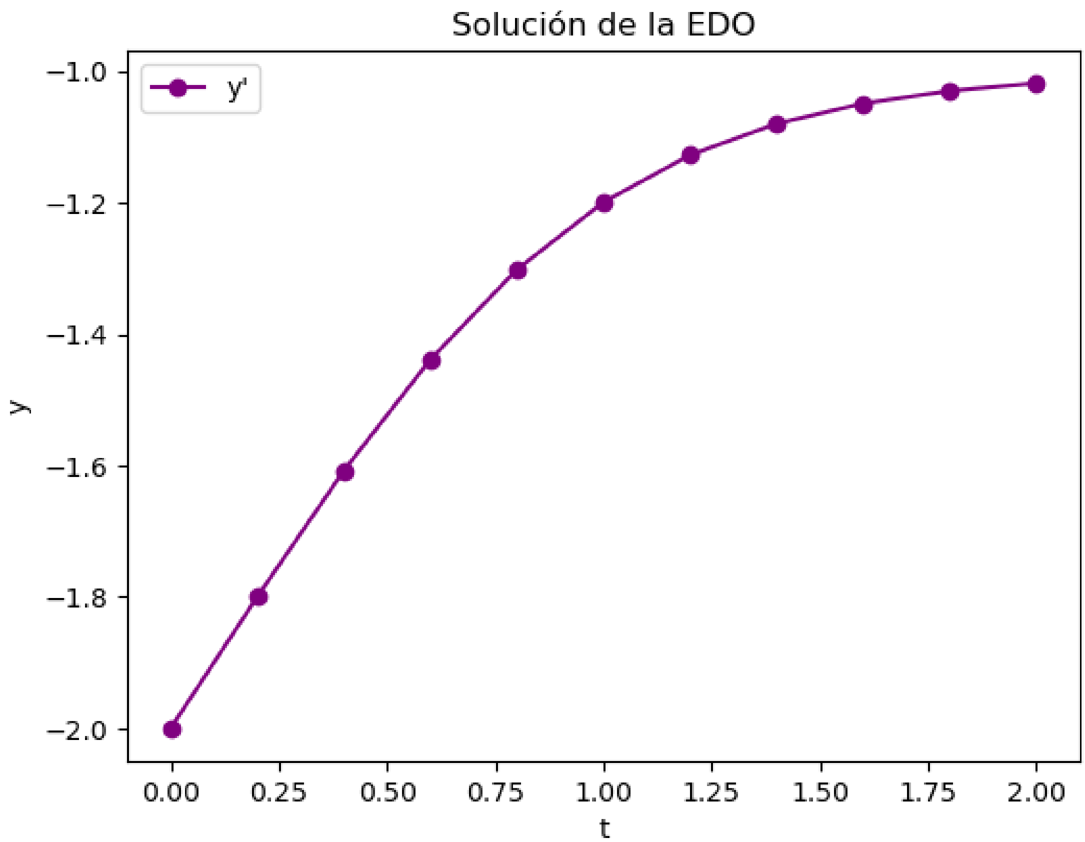
c) $y' = -(y+1)(y+3)$, $0 \leq t \leq 2$, $y(0) = -2$, con $h = 0.2$.

```
%autoreload 2
y_der = lambda t, y: -(y + 1)*(y + 3)
y_init = -2

ys_c2, ts_c2, h = ODE_euler(a = 0, b = 2, f = y_der, y_t0 = y_init, N = 10)

print(f"El valor de h es: {h}")
graphics(ts_c2, ys_c2,"PURPLE")
```

```
El valor de h es: 0.2
```



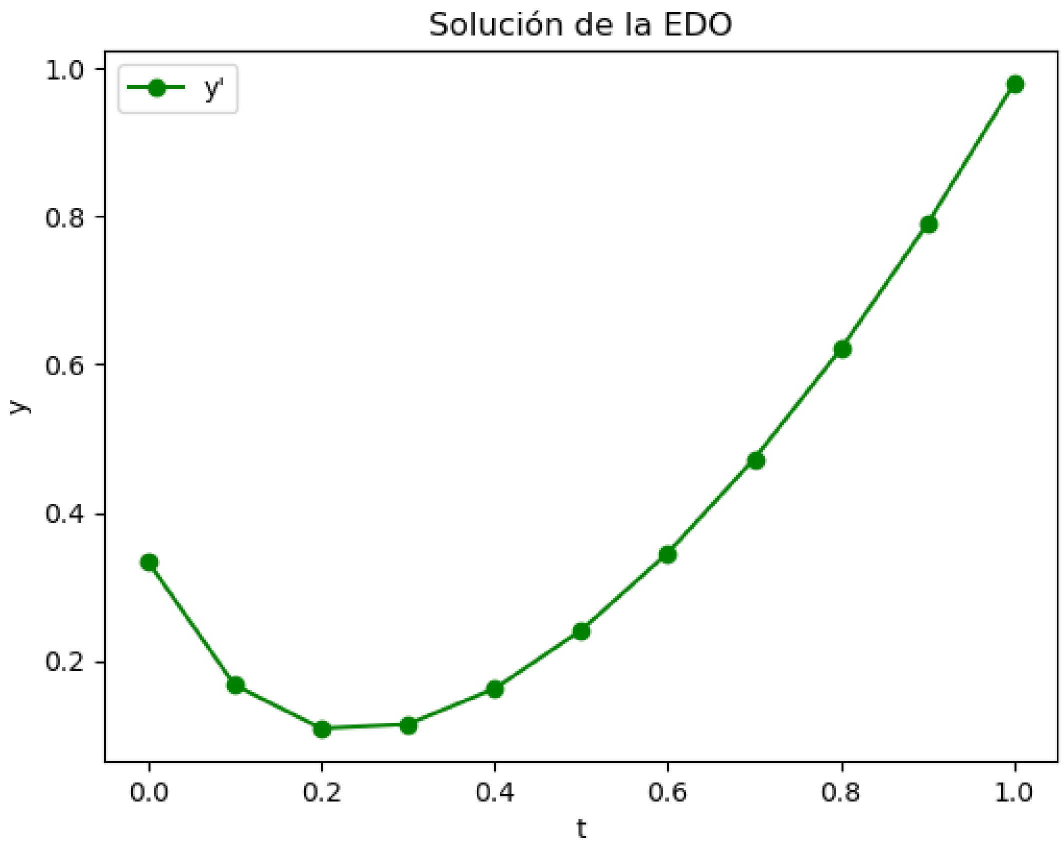
d) $y' = -5y + 5t^2 + 2t$, $0 \leq t \leq 1$, $y(0) = \frac{1}{3}$, con $h = 0.1$.

```
%autoreload 2
y_der = lambda t, y: -5*y + 5*t**2 + 2*t
y_init = 1/3

ys_d2, ts_d2, h = ODE_euler(a = 0, b = 1, f = y_der, y_t0 = y_init, N = 10)

print(f"El valor de h es: {h}")
graphics(ts_d2, ys_d2,"GREEN")
```

```
El valor de h es: 0.1
```



4. Aquí se dan las soluciones reales para los problemas de valor inicial en el ejercicio 3. Calcule el error real en las aproximaciones del ejercicio 3.

a) $y(t) = \frac{t}{1+\ln t}$

```
%autoreload 2
def y1(t):
    return t/(1 + math.log(t))

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_a2, ts_a2)])
print(f"El error real es: {errorReal}")
```

```
El error real es: 0.4026114748989524
```

b) $y(t) = t \tan \ln t$

```
%autoreload 2
def y2(t):
    return t*math.tan(math.log(t))

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_b2, ts_b2)])
print(f"El error real es: {errorReal}")
```

```
El error real es: 1.4857714189452615
```

c) $y(t) = -3 + \frac{2}{1+e^{-2t}}$

```
%autoreload 2
def y3(t):
    return - 3 + 2/(1 + math.exp(-2*t))

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_c2, ts_c2)])
```

```
print(f"El error real es: {errorReal}")
```

```
El error real es: 2.0191941754493365
```

d) $y(t) = t^2 + \frac{1}{3}e^{-5t}$

```
%autoreload 2
def y4(t):
    return t**2 + (1/3)*math.exp(-5*t)

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys_d2, ts_d2)])
print(f"El error real es: {errorReal}")
```

```
El error real es: 0.7773952281750381
```

5. Utilice los resultados del ejercicio 3 y la interpolación lineal para aproximar los siguientes valores de ( ). Compare las aproximaciones asignadas para los valores reales obtenidos mediante las funciones determinadas en el ejercicio 4.

a) $y(0.25)$ y $y(0.93)$.

```
res_1 = y1(0.25)
print(res_1)

res_2 = y1(0.93)
print(res_2)
```

```
-0.6471748623905226
1.0027718477462106
```

b) $y(1.25)$ y $y(1.93)$.

```
res_1 = y2(1.25)
print(res_1)

res_2 = y2(1.93)
print(res_2)
```

```
0.2836531261952289
1.4902277738186658
```

c) $y(2.10)$ y $y(2.75)$.

```
res_1 = y3(2.1)
print(res_1)

res_2 = y3(2.75)
print(res_2)
```

```
-1.0295480633865461
-1.008140275431792
```

d) $y(0.54)$ y $y(0.94)$.

```
res_1 = y4(0.54)
print(res_1)

res_2 = y4(0.94)
print(res_2)
```

```
0.3140018375799166
0.8866317590338986
```

6. Use el método de Taylor de orden 2 para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

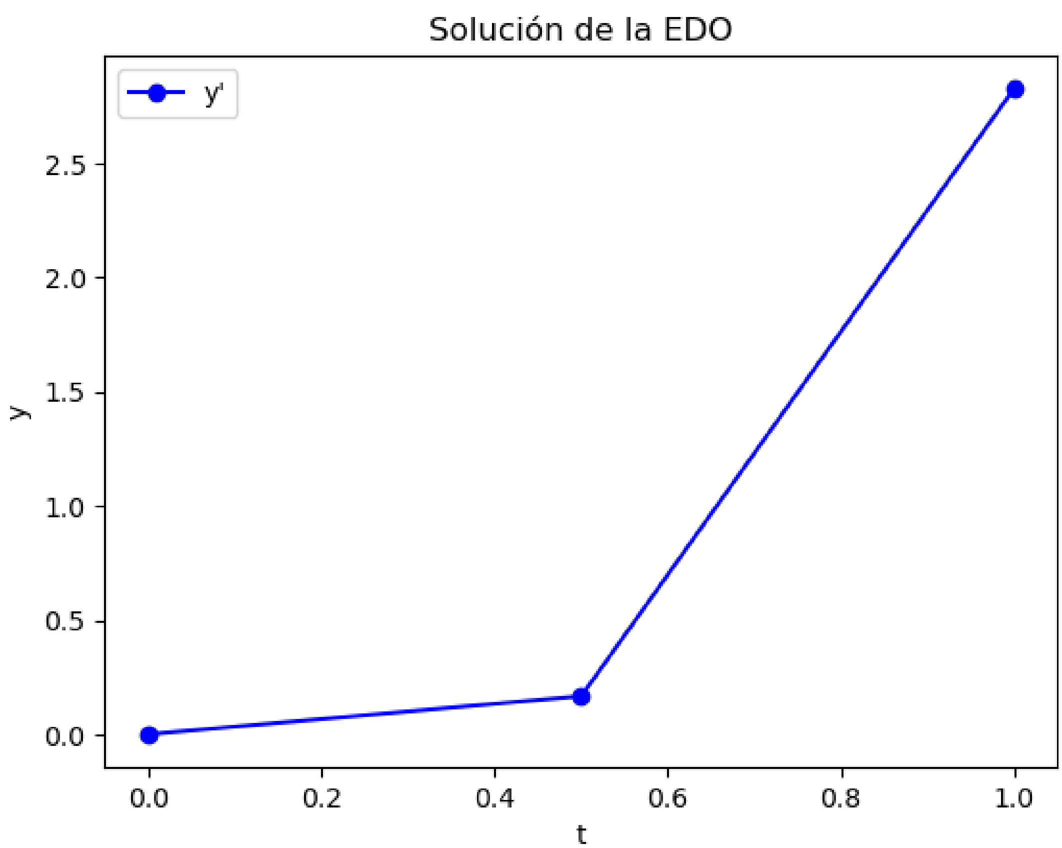
a) $y' = te^{3t} - 2y$, $0 \leq t \leq 1$, $y(0) = 0$, con $h = 0.5$

```
%autoreload 2
y_der = lambda t, y: t*math.exp(3*t) - 2*y
y_der_2 = lambda t, y: -2*y_der(t, y) + math.exp(3*t) + 3*t*math.exp(3*t)
y_der_3 = lambda t, y: -2*y_der_2(t, y) + 3 * math.exp(3*t) + 9*t*math.exp(3*t)
y_init = 0

ys_a6, ts_a6, h = ODE_euler_nth(a = 0, b = 1, f = y_der,
                                f_derivatives = [y_der_2, y_der_3],
                                y_t0 = y_init, N = 2)

print(f"El valor de h es: {h}")
graphics(ts_a6, ys_a6,"BLUE")
```

```
El valor de h es: 0.5
```



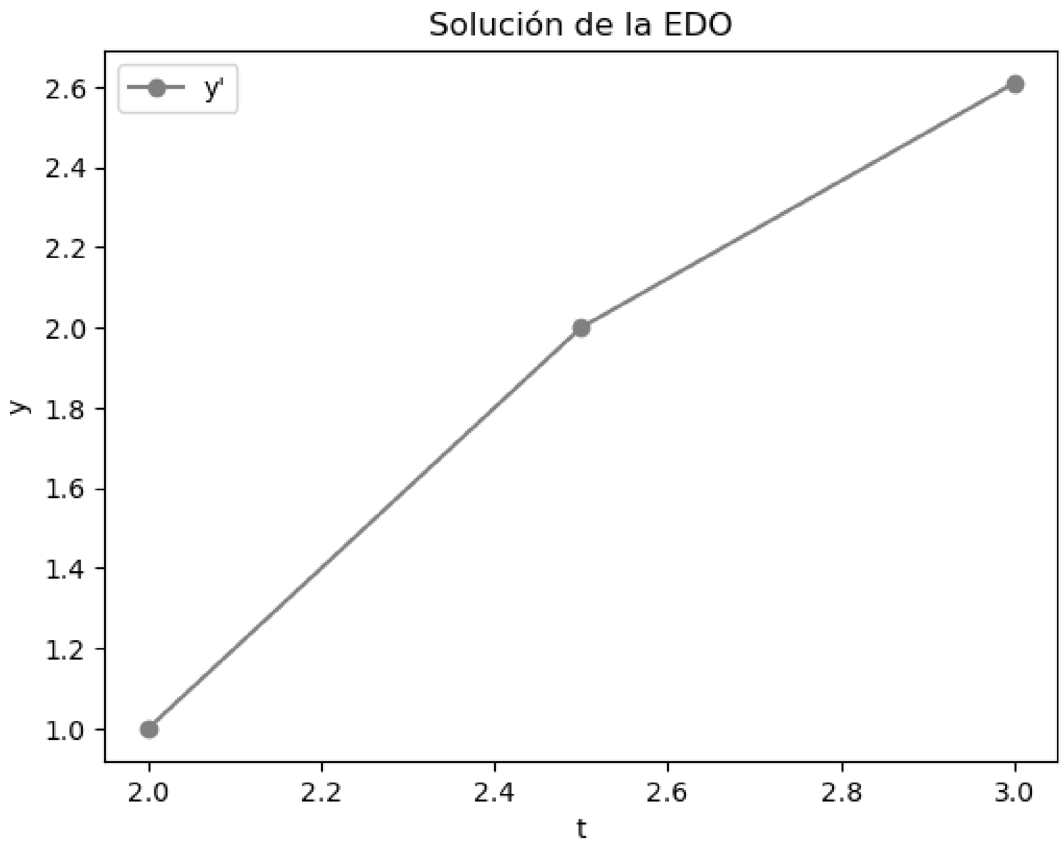
b) $y' = 1 + (t - y)^2$, $2 \leq t \leq 3$, $y(2) = 1$, con $h = 0.5$

```
%autoreload 2
y_der = lambda t, y: 1 + (t - y)**2
y_der_2 = lambda t, y: 2*(t - y)*(1 - y_der(t,y))
y_der_3 = lambda t, y: 2*(1 - y_der(t, y))**2 - 2*(t - y)*y_der_2(t,y)
y_init = 1

ys_b6, ts_b6, h = ODE_euler_nth(a = 2, b = 3, f = y_der,
                                f_derivatives = [y_der_2, y_der_3],
                                y_t0 = y_init, N = 2)
```

```
print(f"El valor de h es: {h}")
graphics(ts_b6, ys_b6,"GREY")
```

```
El valor de h es: 0.5
```



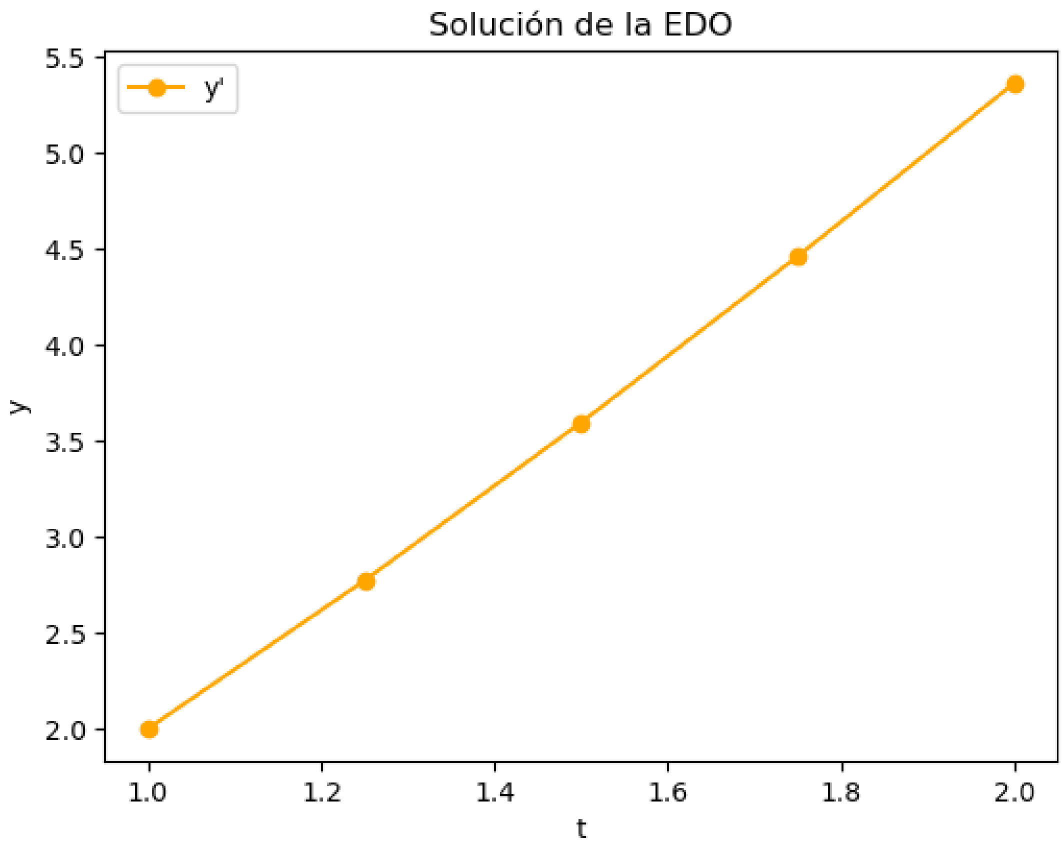
c) $y' = 1 + \frac{y}{t}$, $1 \leq t \leq 2$, $y(1) = 2$, con $h = 0.25$

```
%autoreload 2
y_der = lambda t, y: 1 + y/t
y_der_2 = lambda t, y: (t * y_der(t, y) - y)/t**2
y_der_3 = lambda t, y: -1/t**2
y_init = 2

ys_c6, ts_c6, h = ODE_euler_nth(a = 1, b = 2, f = y_der,
                          f_derivatives = [y_der_2, y_der_3],
                          y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_c6, ys_c6,"ORANGE")
```

```
El valor de h es: 0.25
```



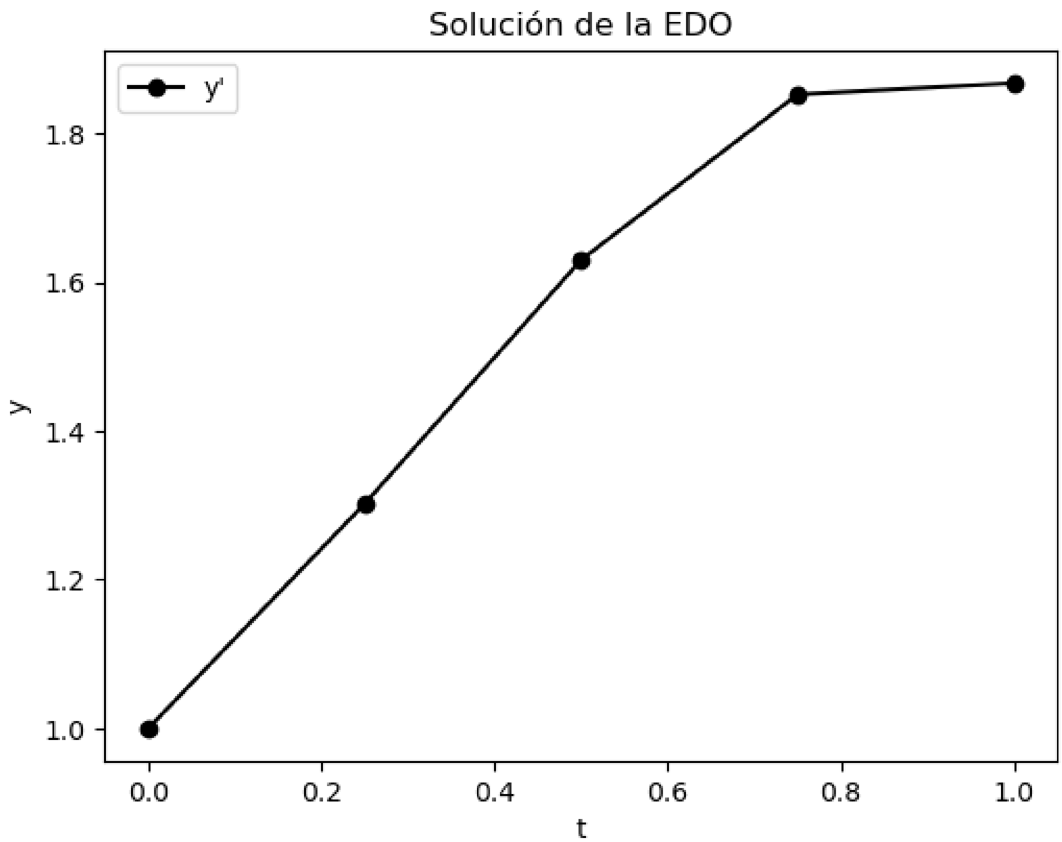
d) $y' = \cos 2t + \sin 3t$, $0 \leq t \leq 1$, $y(0) = 1$, con $h = 0.25$

```
%autoreload 2
y_der = lambda t, y: math.cos(2*t) + math.sin(3*t)
y_der_2 = lambda t, y: -2*math.sin(2*t) + 3*math.cos(3*t)
y_der_3 = lambda t, y: -4*math.cos(2*t) - 9*math.sin(3*t)
y_init = 1

ys_d6, ts_d6, h = ODE_euler_nth(a = 0, b = 1, f = y_der,
                                f_derivatives = [y_der_2, y_der_3],
                                y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_d6, ys_d6,"BLACK")
```

```
El valor de h es: 0.25
```



7. Repita el ejercicio 6 con el método de Taylor de orden 4.

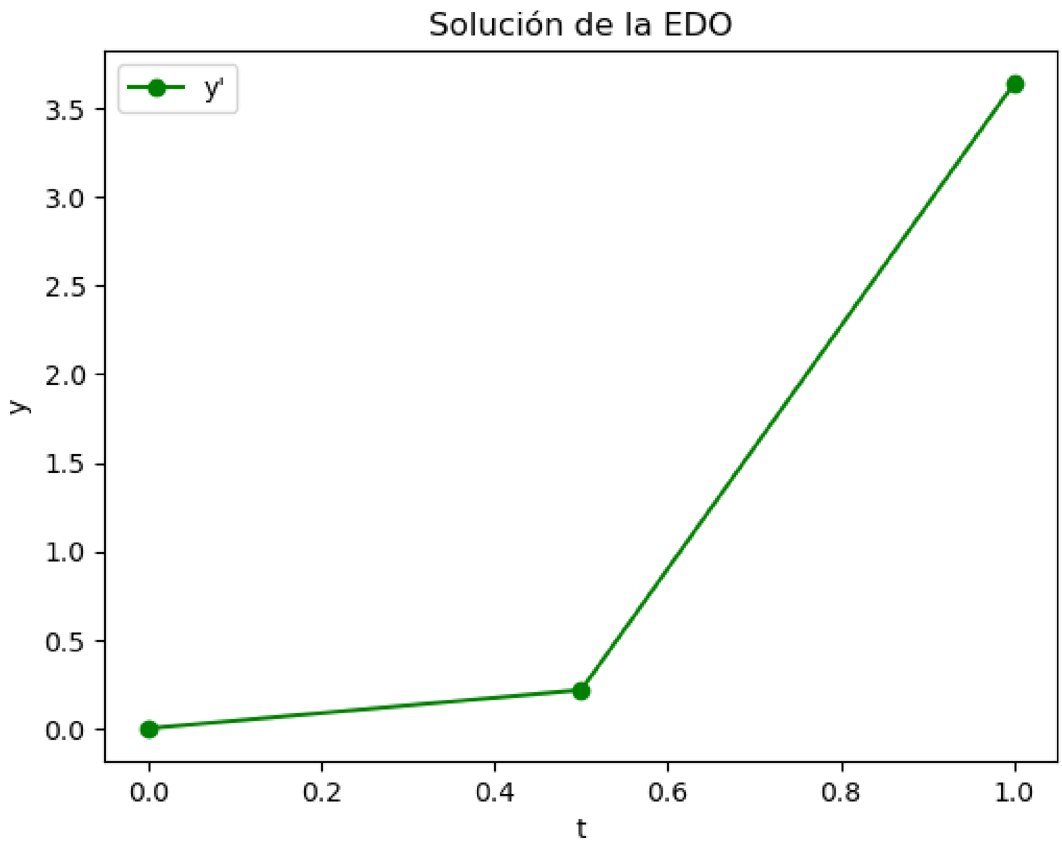
a) $y' = te^{3t} - 2y$, $0 \leq t \leq 1$, $y(0) = 0$, con $h = 0.5$

```
%autoreload 2
y_der = lambda t, y: t*math.exp(3*t) - 2*y
y_der_2 = lambda t, y: -2*y_der(t, y) + math.exp(3*t) + 3*t*math.exp(3*t)
y_der_3 = lambda t, y: -2*y_der_2(t, y) + 3*math.exp(3*t) + 9*t*math.exp(3*t)
y_der_4 = lambda t, y: -2*y_der_3(t, y) + 6*math.exp(3 * t) + 27*t*math.exp(3*t)
y_der_5 = lambda t, y: -2*y_der_4(t, y) + 12*math.exp(3 * t) + 81*t*math.exp(3*t)
y_init = 0

ys_a7, ts_a7, h = ODE_euler_nth(a = 0, b = 1, f = y_der,
                          f_derivatives = [y_der_2, y_der_3, y_der_4, y_der_5],
                          y_t0 = y_init, N = 2)

print(f"El valor de h es: {h}")
graphics(ts_a7, ys_a7,"GREEN")
```

```
El valor de h es: 0.5
```



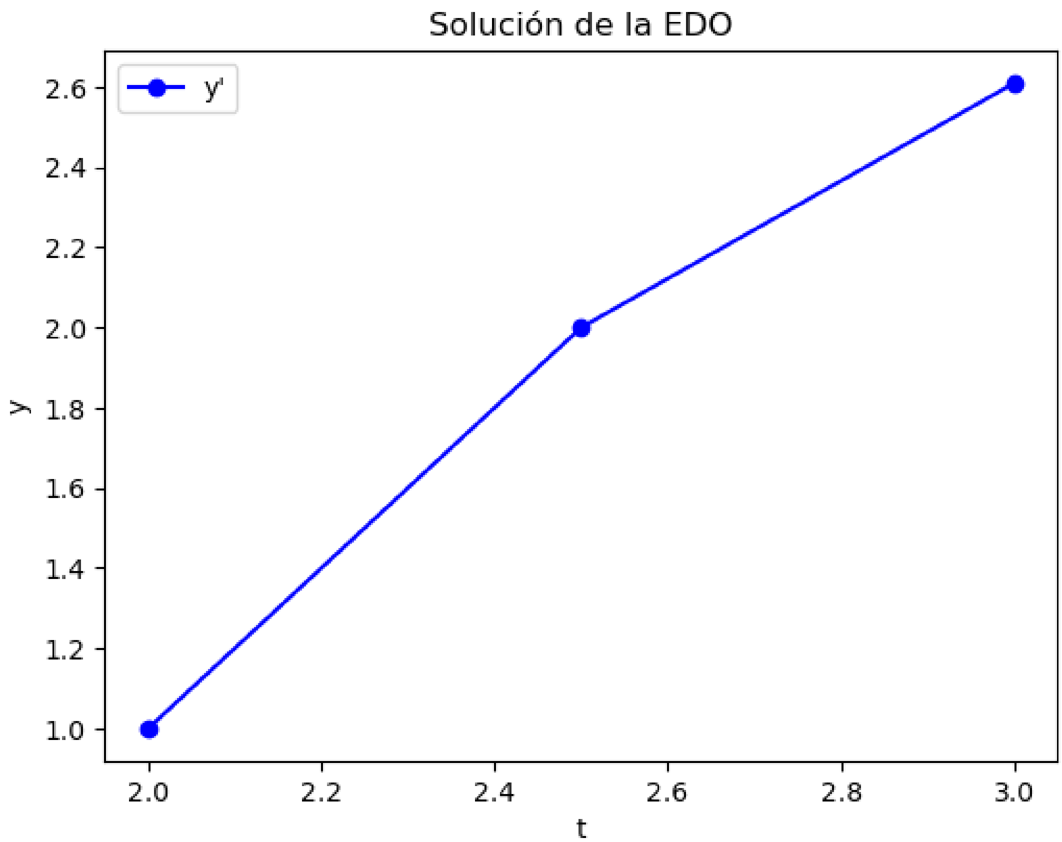
b) $y' = 1 + (t - y)^2$, $2 \leq t \leq 3$, $y(2) = 1$, con $h = 0.5$

```
%autoreload 2
y_der = lambda t, y: 1 + (t - y)**2
y_der_2 = lambda t, y: 2*(t - y)*(1 - y_der(t,y))
y_der_3 = lambda t, y: 2*(1 - y_der(t, y))**2 - 2*(t - y)*y_der_2(t,y)
y_init = 1

ys_b7, ts_b7, h = ODE_euler_nth(a = 2, b = 3, f = y_der,
                         f_derivatives = [y_der_2, y_der_3],
                         y_t0 = y_init, N = 2)

print(f"El valor de h es: {h}")
graphics(ts_b7, ys_b7,"BLUE")
```

```
El valor de h es: 0.5
```



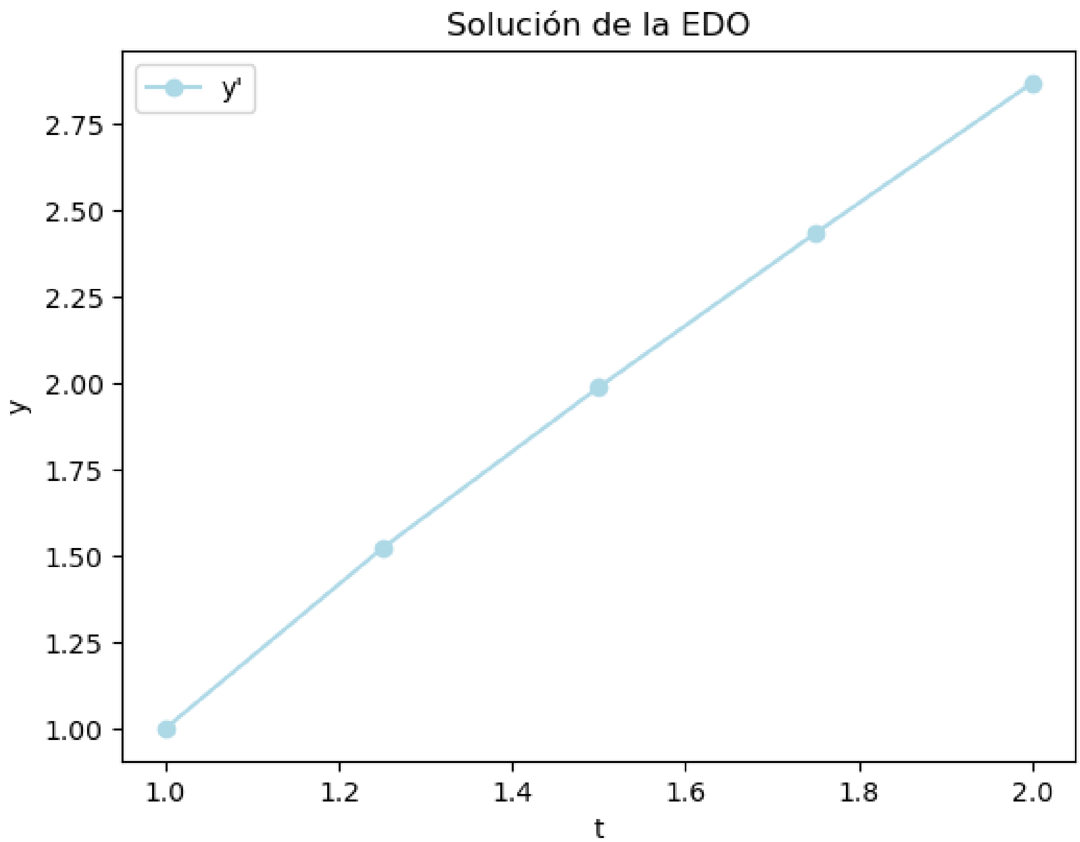
c) $y' = 1 + \frac{y}{t}$, $1 \leq t \leq 2$, $y(1) = 2$, con $h = 0.25$

```
%autoreload 2
y_der = lambda t, y: 1 + (t / y)
y_der_2 = lambda t, y: (t * y_der(t, y) - y) / t**2
y_der_3 = lambda t, y: ((t**2) * y_der_2(t, y) - (2 * t * (y_der(t, y) - (y / t)))) / t**3
y_init = 1

ys_c7, ts_c7, h = ODE_euler_nth(a = 1, b = 2, f = y_der,
                         f_derivatives = [y_der_2, y_der_3],
                         y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_c7, ys_c7,"LIGHTBLUE")
```

```
El valor de h es: 0.25
```



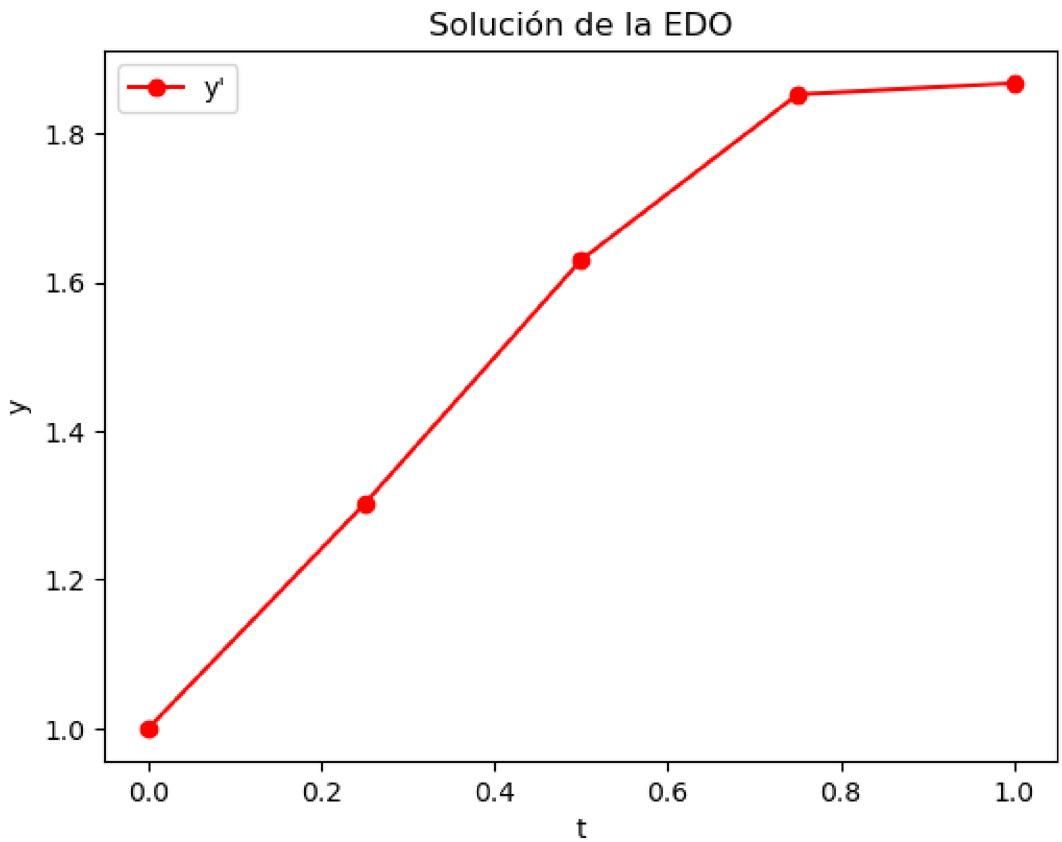
d) $y' = \cos 2t + \sin 3t$, $0 \le t \le 1$, $y(0) = 1$, con $h = 0.25$

```
%autoreload 2
y_der = lambda t, y: math.cos(2*t) + math.sin(3*t)
y_der_2 = lambda t, y: -2 * math.sin(2*t) + 3 * math.cos(3*t)
y_der_3 = lambda t, y: -4 * math.cos(2*t) - 9 * math.sin(3*t)
y_init = 1

ys_d7, ts_d7, h = ODE_euler_nth(a = 0, b = 1, f = y_der,
                         f_derivatives = [y_der_2, y_der_3],
                         y_t0 = y_init, N = 4)

print(f"El valor de h es: {h}")
graphics(ts_d7, ys_d7,"RED")
```

```
El valor de h es: 0.25
```



Link del repositorio:

https://github.com/MarckHA/Tarea_12-ODE-Metodo-Euler.git