ESCUELA POLITÉCNICA NACIONAL
MÉTODOS NUMÉRICOS

Leandro Bravo

GR1CC

Tarea 12 - ODE Método de Euler

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

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Conjunto de ejercicios

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

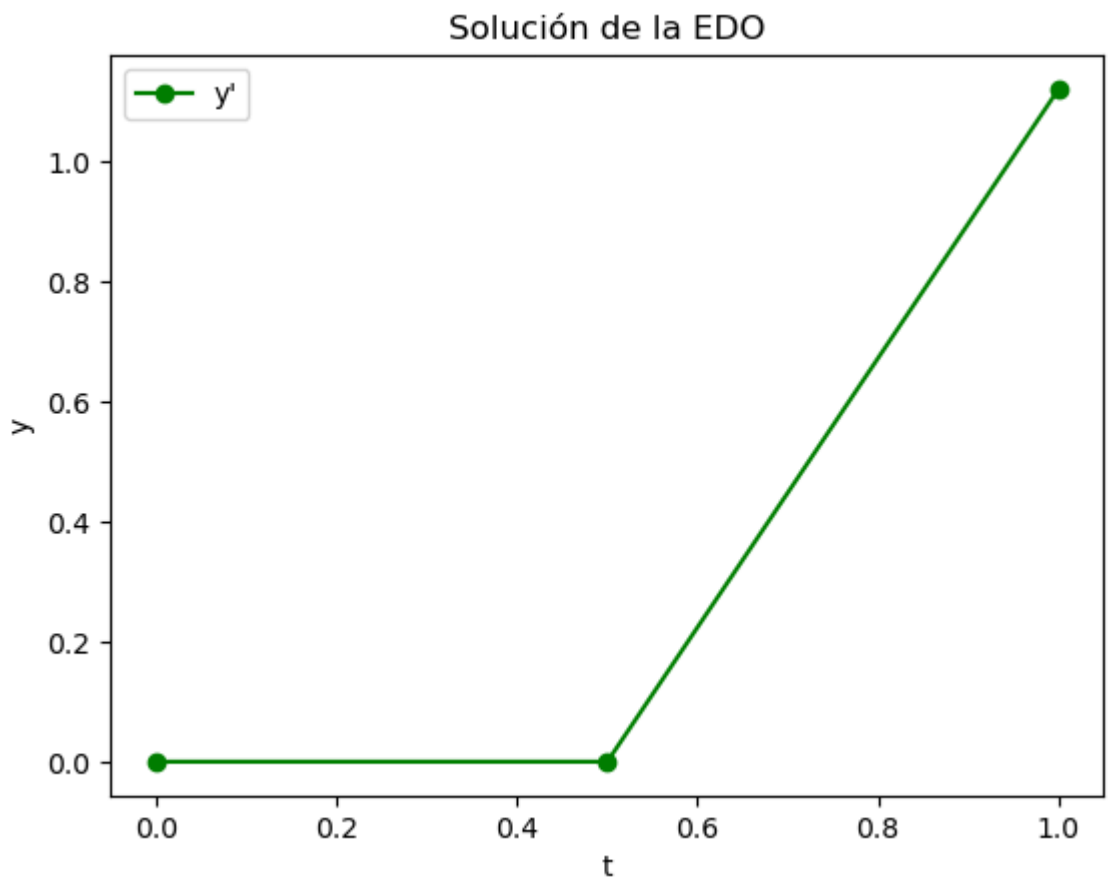
a. $y' = te^{3t} - 2y, 0 \leq t \leq 1, y(0) = 0$, con $h = 0.5$

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

ys1a, ts1a, 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(ts1a, ys1a)
```

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



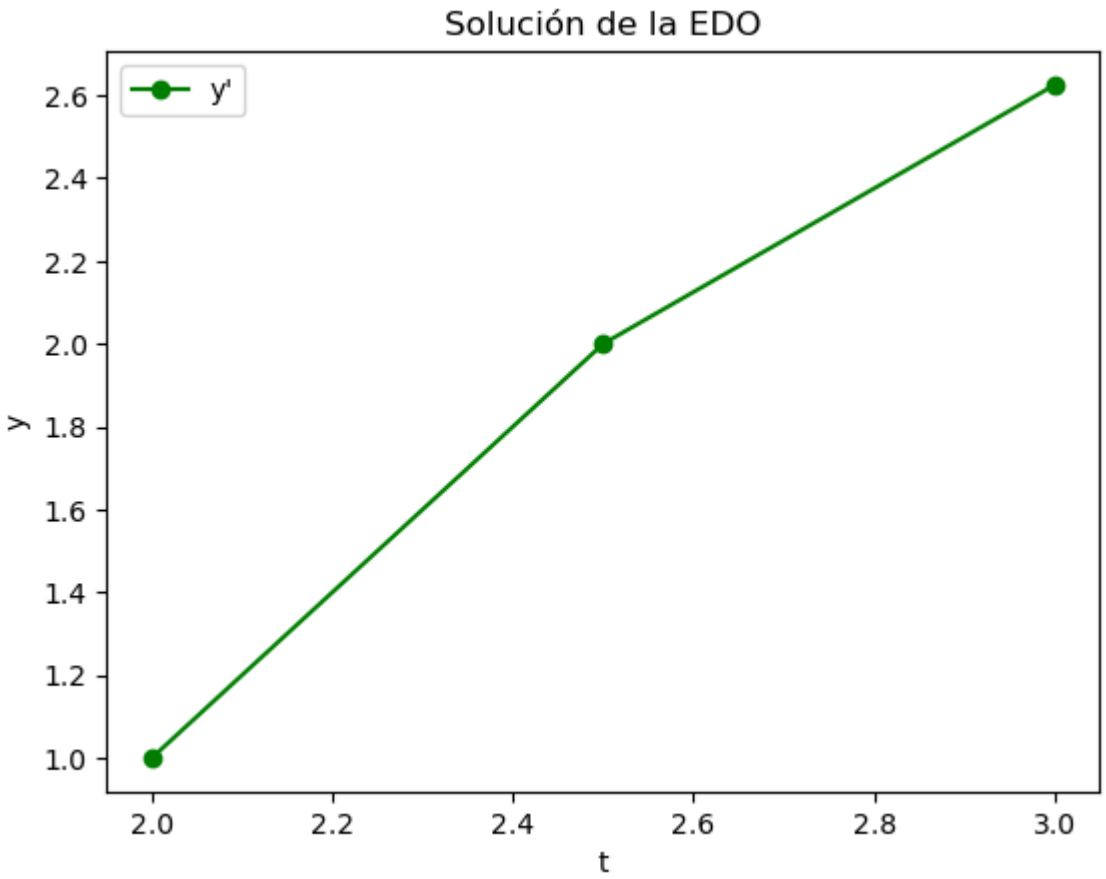
b. $y' = 1 + (t - y)^2, 2 \leq t \leq 3, y(2) = 1$, con $h = 0.5$

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

ys1b, ts1b, 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(ts1b, ys1b)
```

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



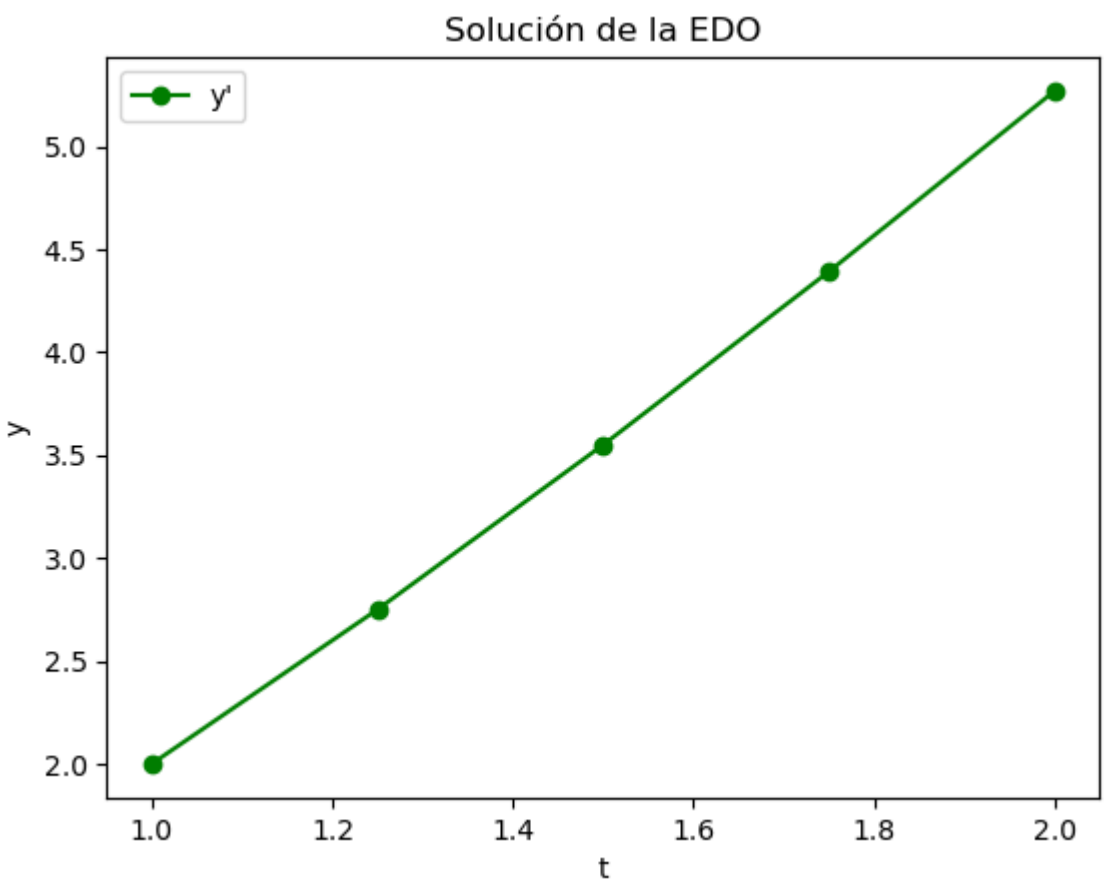
c. $y' = 1 + \frac{y}{t}, 1 \leq t \leq 2, y(1) = 2$, con $h = 0.25$

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

ys1c, ts1c, 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(ts1c, ys1c)
```

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



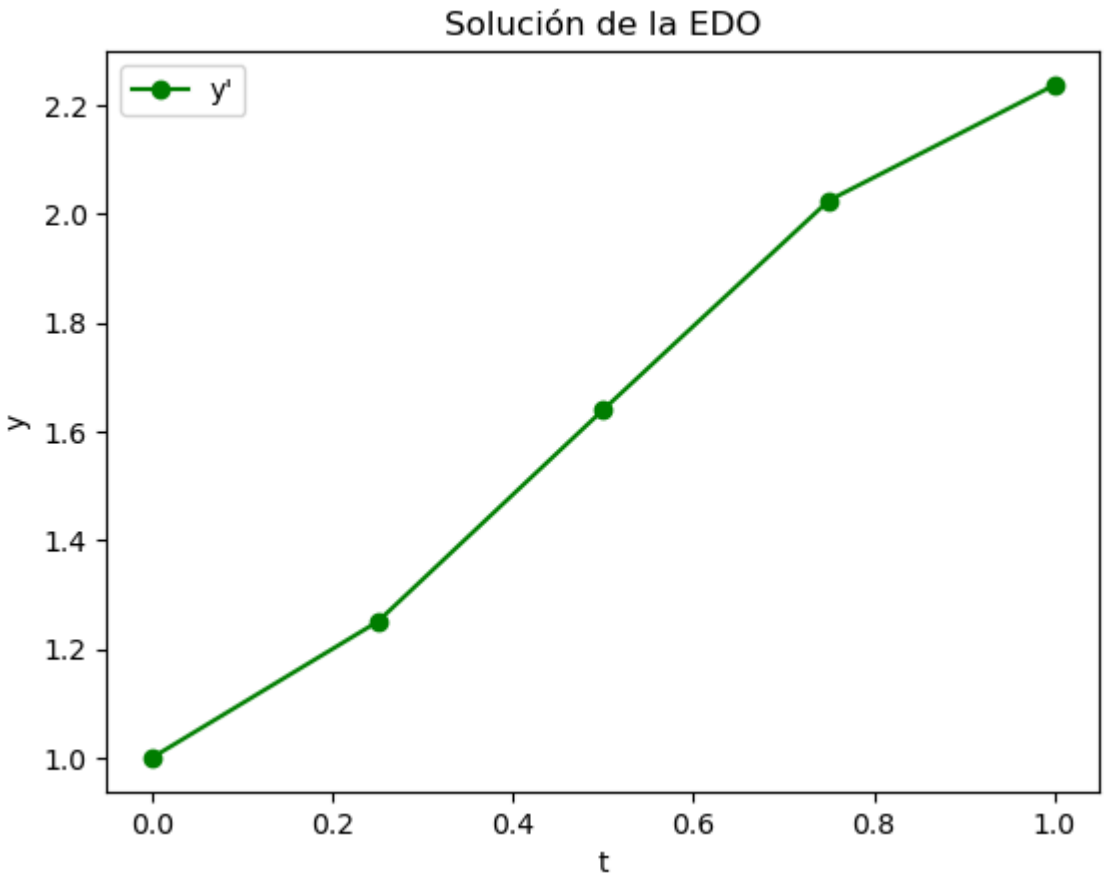
d. $y' = \cos 2t + \sin 3t, 0 \leq t \leq 1, y(0) = 1$, con $h = 0.25$

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

ys1d, ts1d, 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(ts1d, ys1d)
```

```
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)

errorReal = np.mean([abs(y(t) - y_aprox) / abs(y(t)) for y_aprox, t in zip(ys1a, ts1a)])
print(f"El error real es: {errorReal}")
```

```
ZeroDivisionError: float division by zero
```

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(ys1b, ts1b)])
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(ys1c, ts1c)])
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(ys1d, ts1d)])
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.

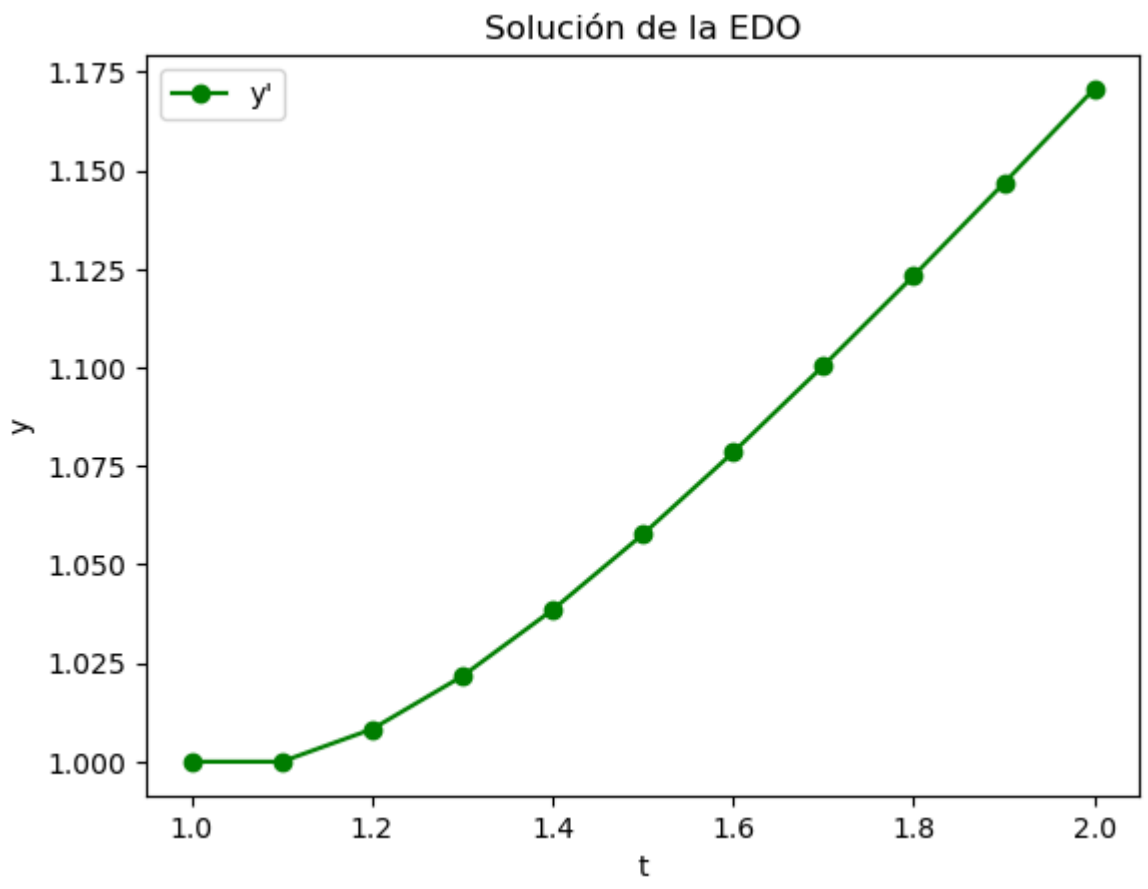
a. $y' = \frac{y}{t} - (\frac{y}{t})^2, 1 \leq t \leq 2, y(1) = 1$, con $h = 0.1$.

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

ys2a, ts2a, 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(ts2a, ys2a)
```

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



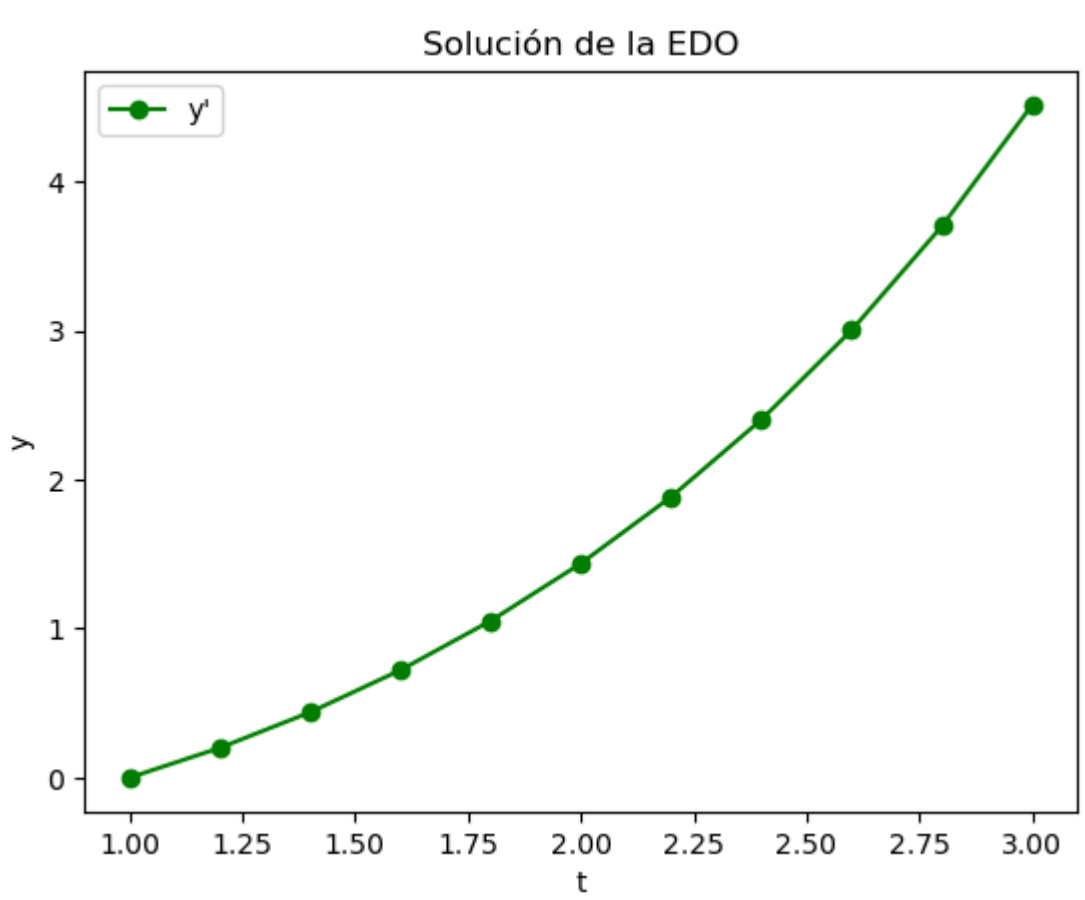
b. $y' = 1 + \frac{y}{t} + (\frac{y}{t})^2, 1 \leq t \leq 3, y(1) = 0,$ con $h = 0.2.$

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

ys2b, ts2b, 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(ts2b, ys2b)
```

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



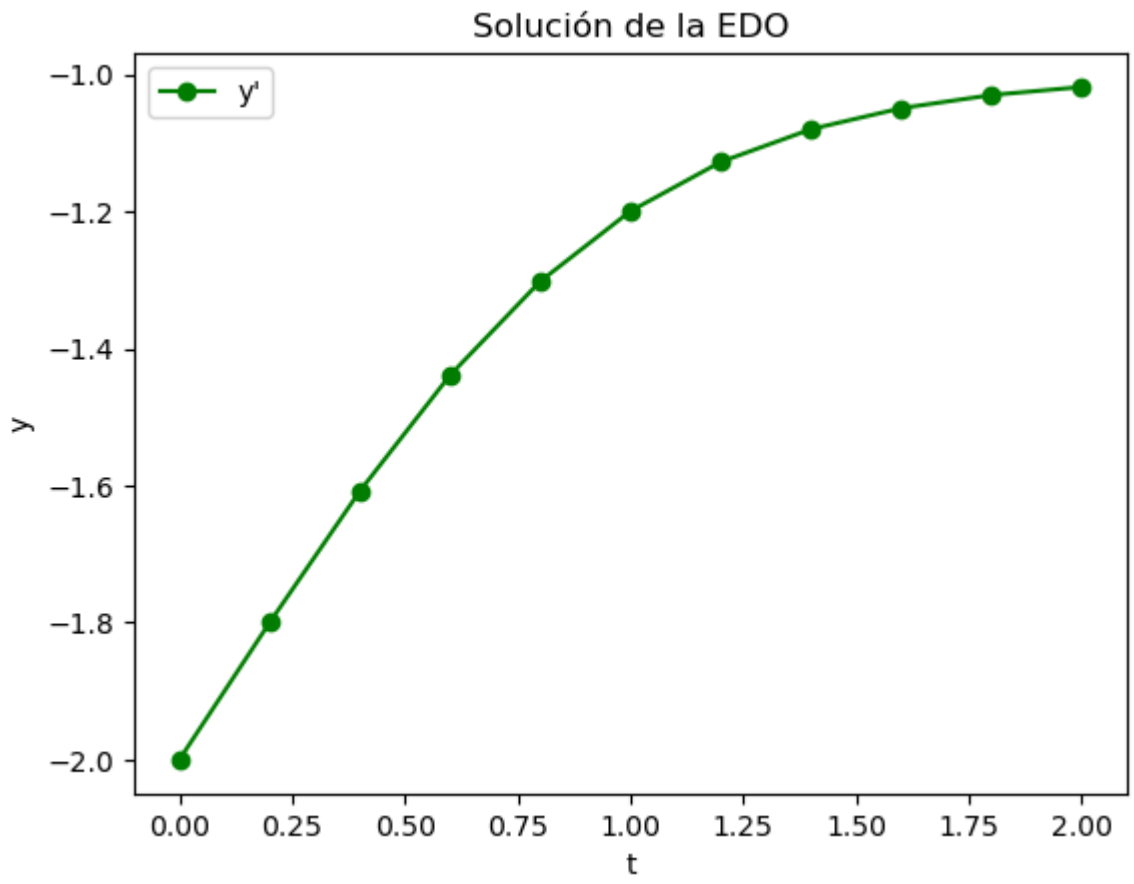
c. $y' = -(y + 1)(y + 3), 0 \leq t \leq 2, y(0) = -2,$ con $h = 0.2.$

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

ys2c, ts2c, 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(ts2c, ys2c)
```

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



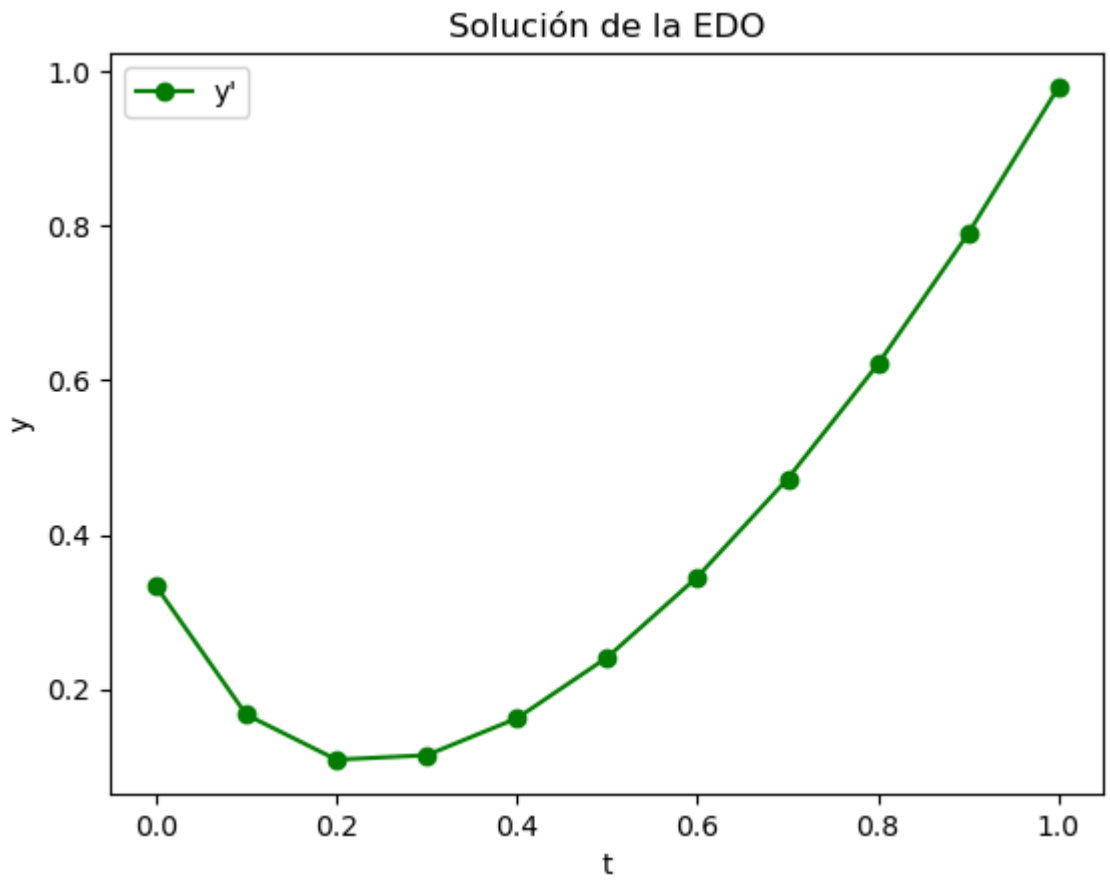
d. $y' = -5y + 5t^2 + 2t, 0 \leq t \leq 1, y(0) = \frac{1}{3}$, con $h = 0.1$.

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

ys2d, ts2d, 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(ts2d, ys2d)
```

```
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(ys2a, ts2a)])
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(ys2b, ts2b)])
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(ys2c, ts2c)])
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(ys2d, ts2d)])
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 $y(t)$. 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 = y1(0.25)
print(res)

res = y1(0.93)
print(res)
```

```
-0.6471748623905226
1.0027718477462106
```

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

```
res = y2(1.25)
print(res)

res = y2(1.93)
print(res)
```

```
0.2836531261952289
1.4902277738186658
```

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

```
res = y3(2.1)
print(res)
```

```
res = y3(2.75)
print(res)
```

```
-1.0295480633865461
-1.008140275431792
```

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

```
res = y4(0.54)
print(res)

res = y4(0.94)
print(res)
```

```
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.

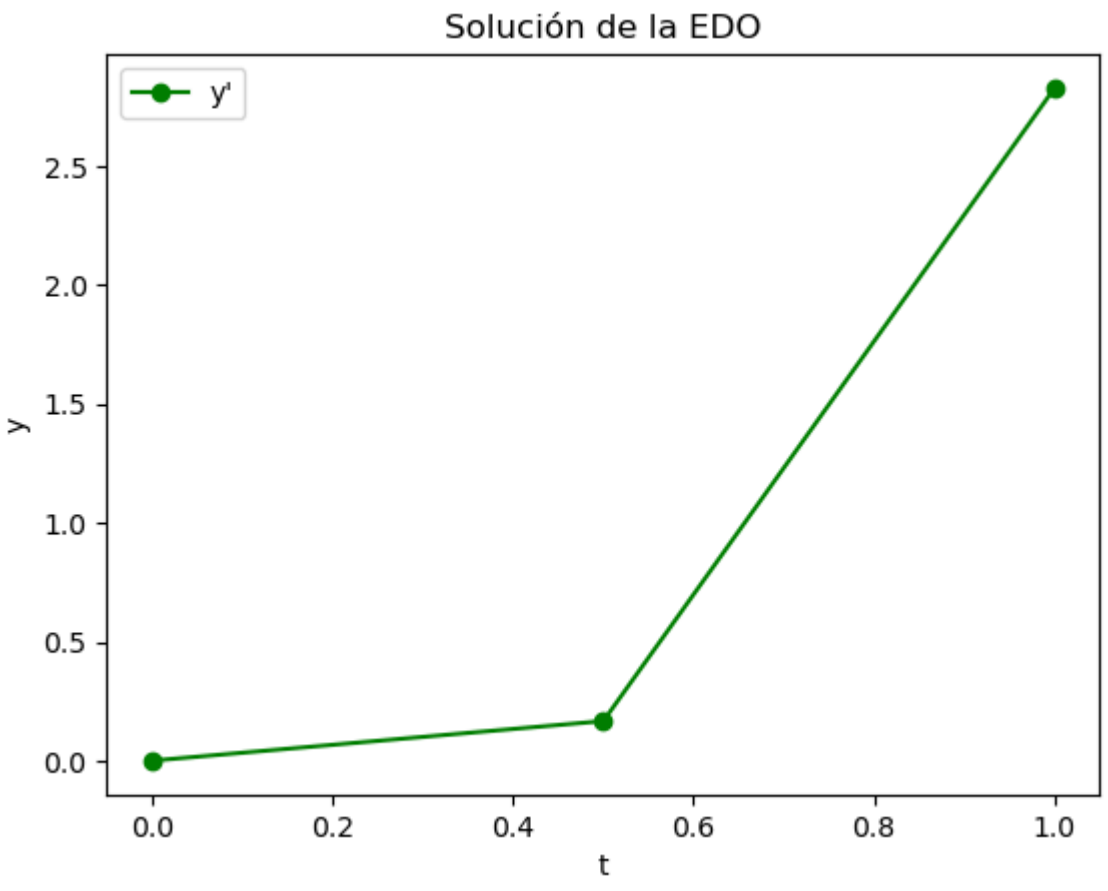
a. $y' = te^{3t} - 2y$, $0 \leq t \leq 1$, $y(0) = 0$, con $h = 0.5$

```
%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

ys3a, ts3a, 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(ts3a, ys3a)
```

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



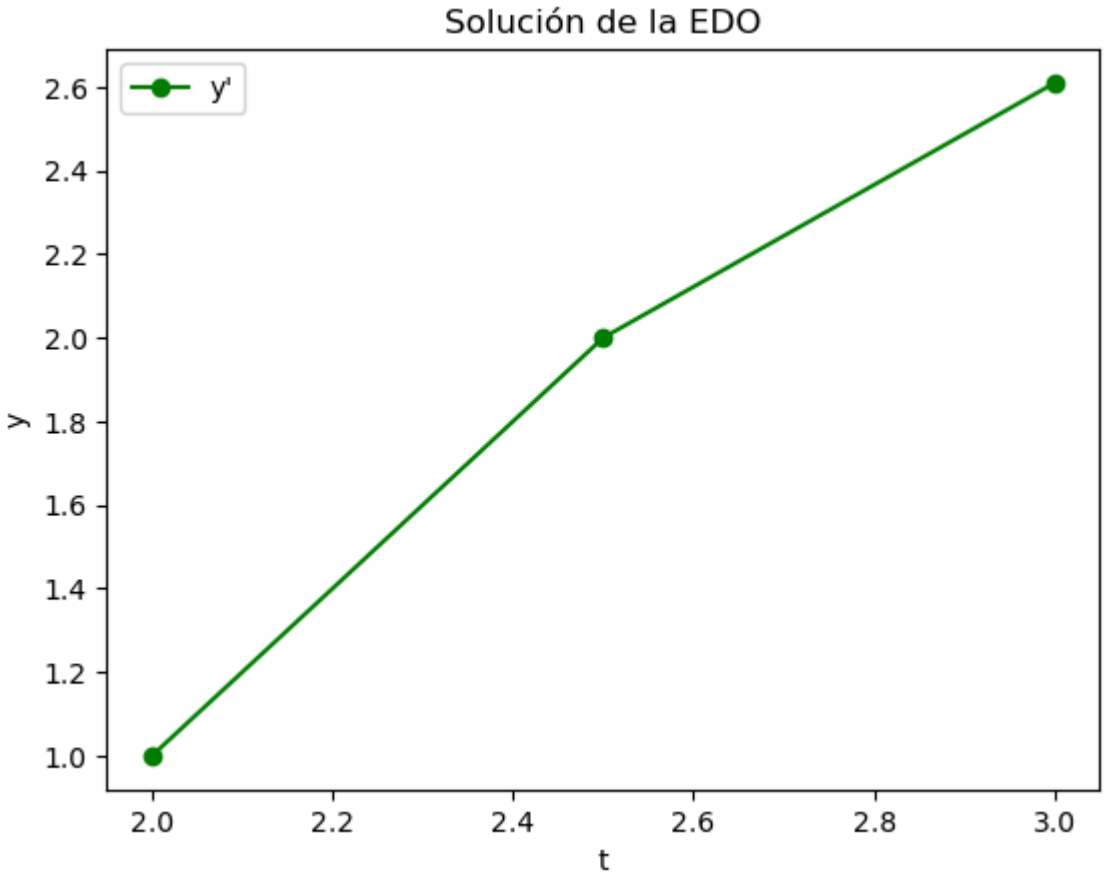
b. $y' = 1 + (t - y)^2$, $2 \leq t \leq 3$, $y(2) = 1$, con $h = 0.5$

```
%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

ys3b, ts3b, 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(ts3b, ys3b)
```

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



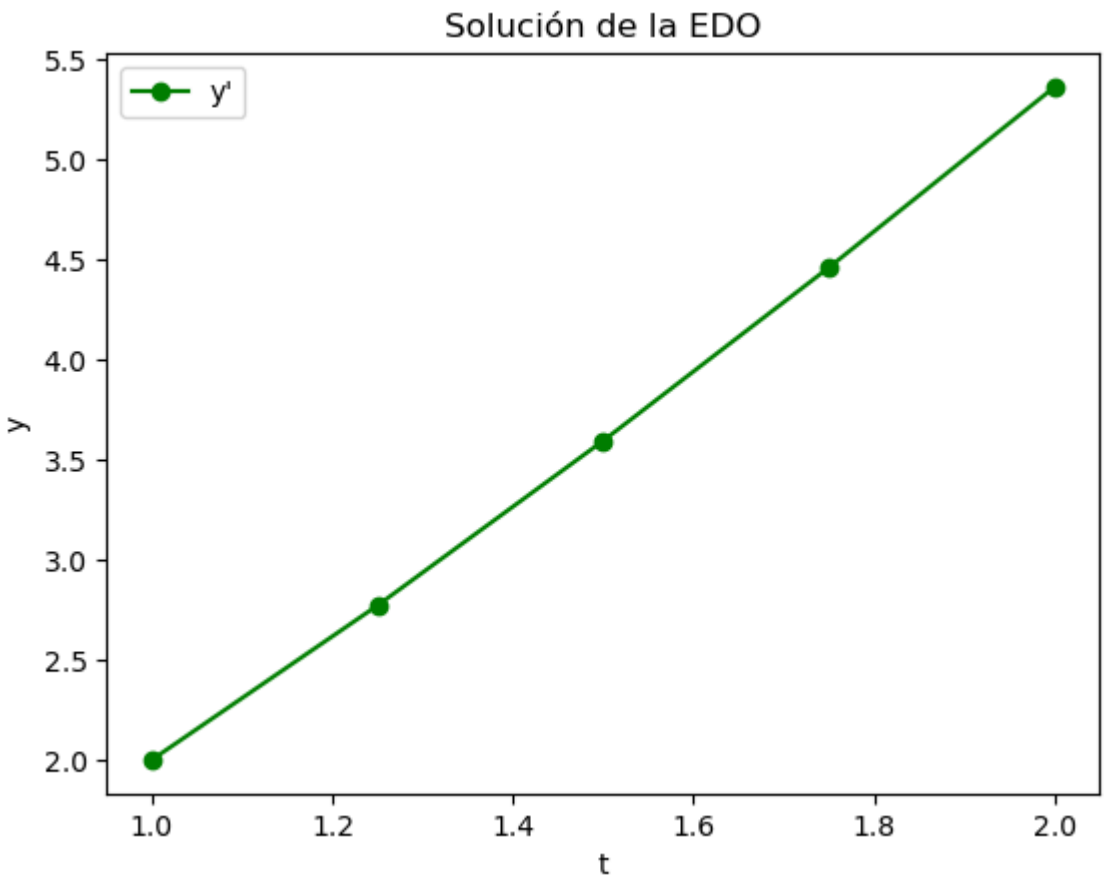
c. $y' = 1 + \frac{y}{t}, 1 \leq t \leq 2, y(1) = 2$, con $h = 0.25$

```
%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

ys3c, ts3c, 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(ts3c, ys3c)
```

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



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

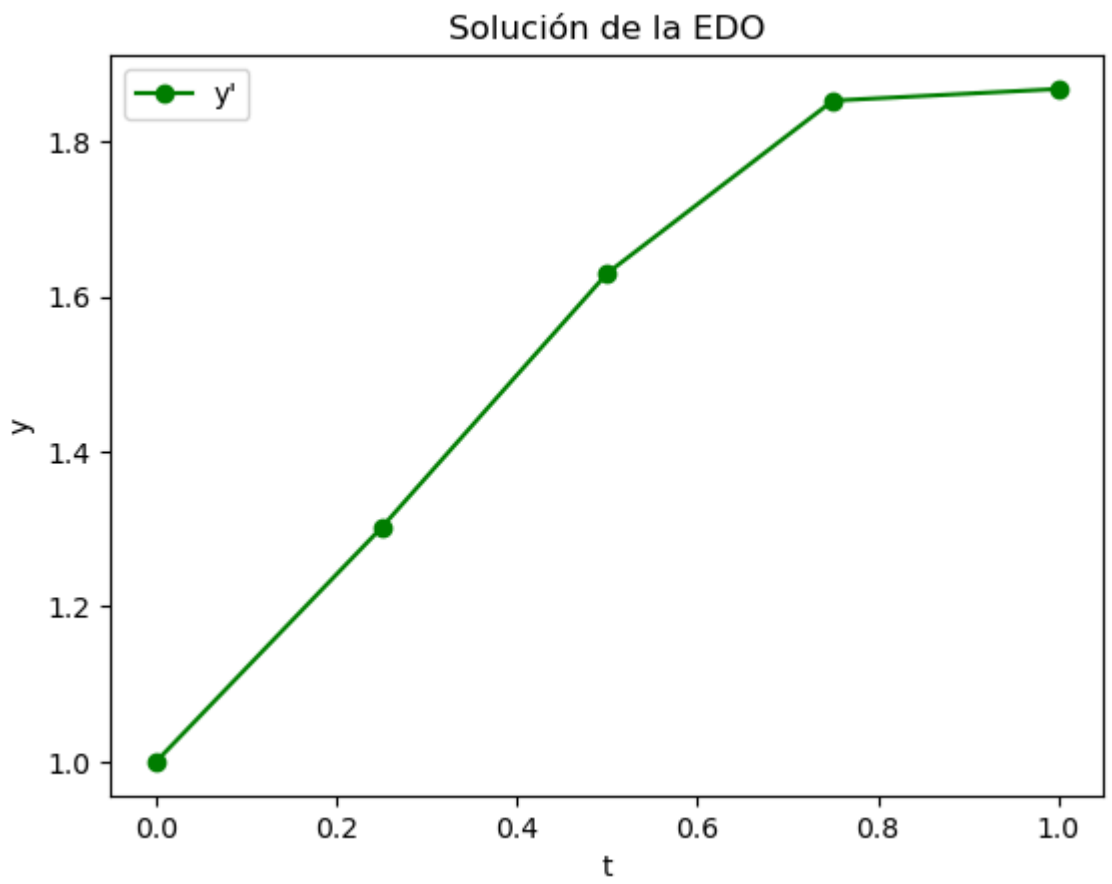
```
%autoreload 2
y_der = lambda t, y: math.cos(2*t) + math.sin(3*t)
y_der_2 = lambda t, y: -2*math.sin(2*t) + 3*math.cos(3*t)
y_der_3 = lambda t, y: -4*math.cos(2*t) - 9*math.sin(3*t)
y_init = 1

ys3d, ts3d, 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(ts3d, ys3d)
```

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



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

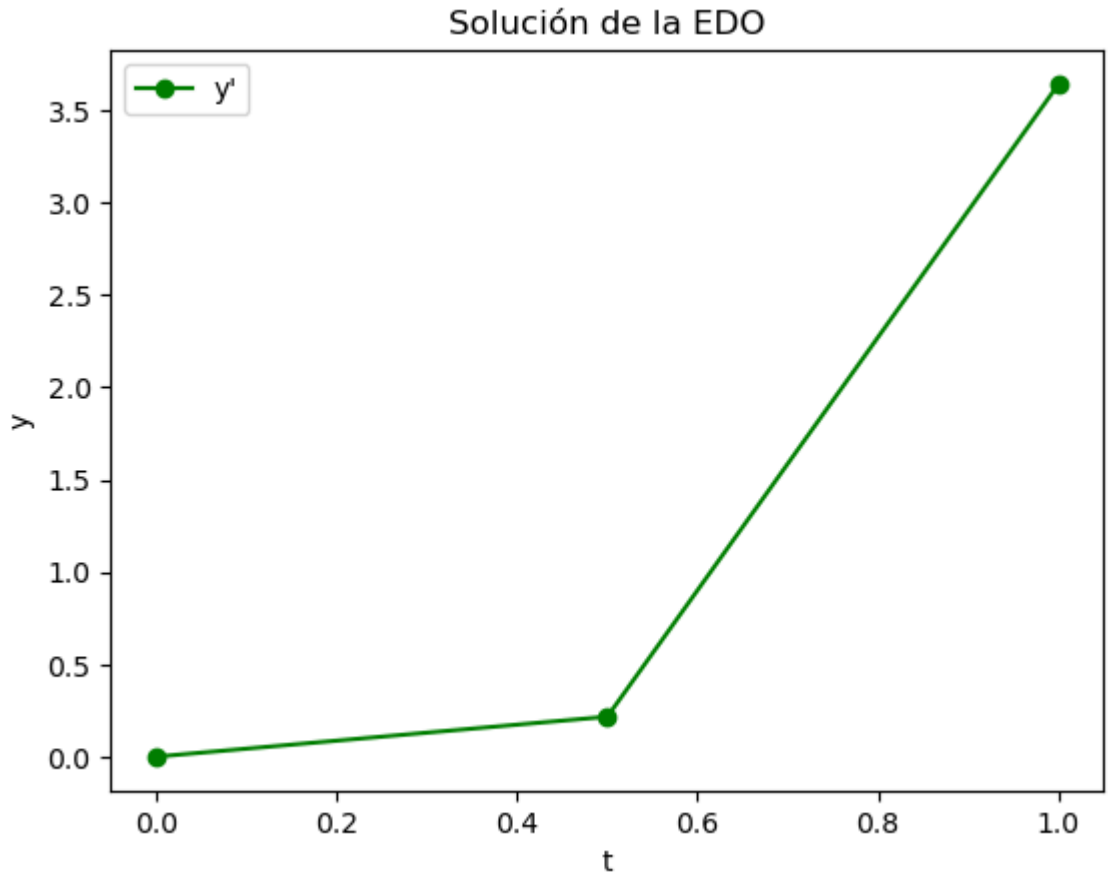
a. $y' = te^{3t} - 2y, 0 \leq t \leq 1, y(0) = 0$, con $h = 0.5$

```
        %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

        ys4a, ts4a, 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(ts4a, ys4a)
```

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



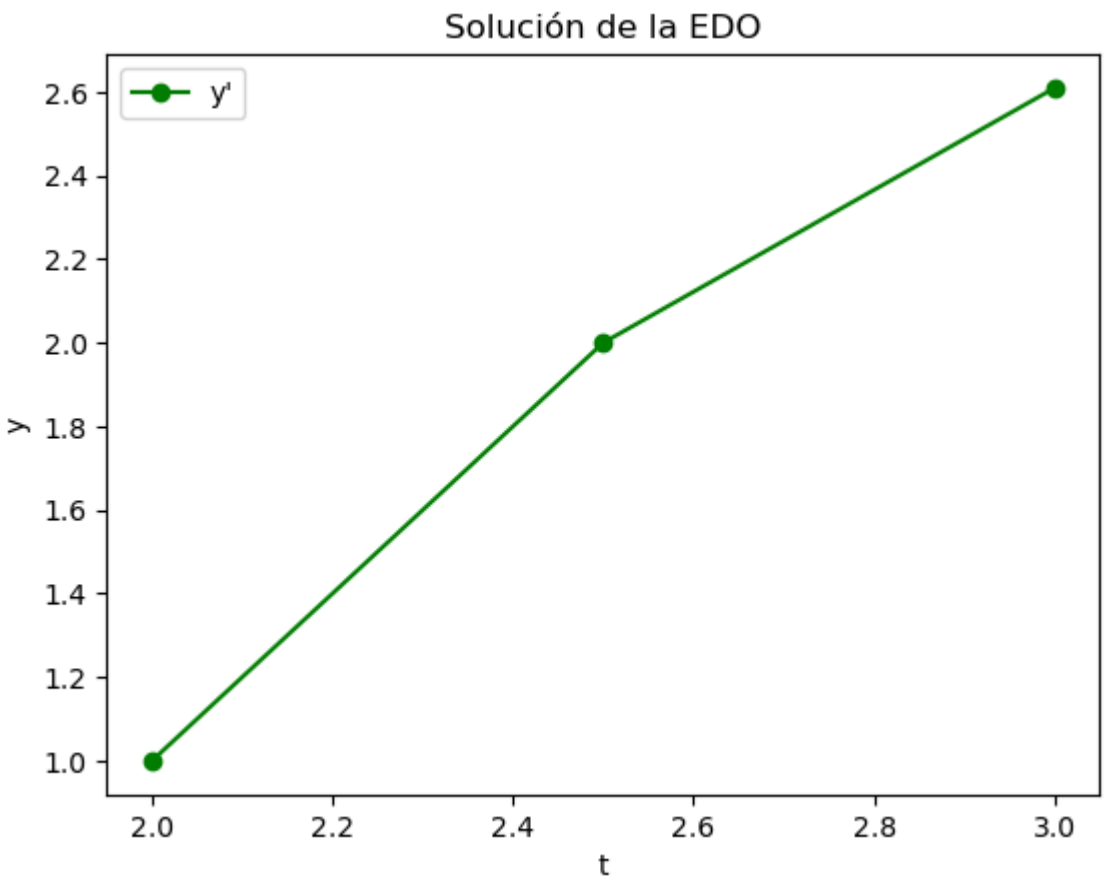
b. $y' = 1 + (t - y)^2$, $2 \leq t \leq 3$, $y(2) = 1$, con $h = 0.5$

```
%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

ys4b, ts4b, 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(ts4b, ys4b)
```

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



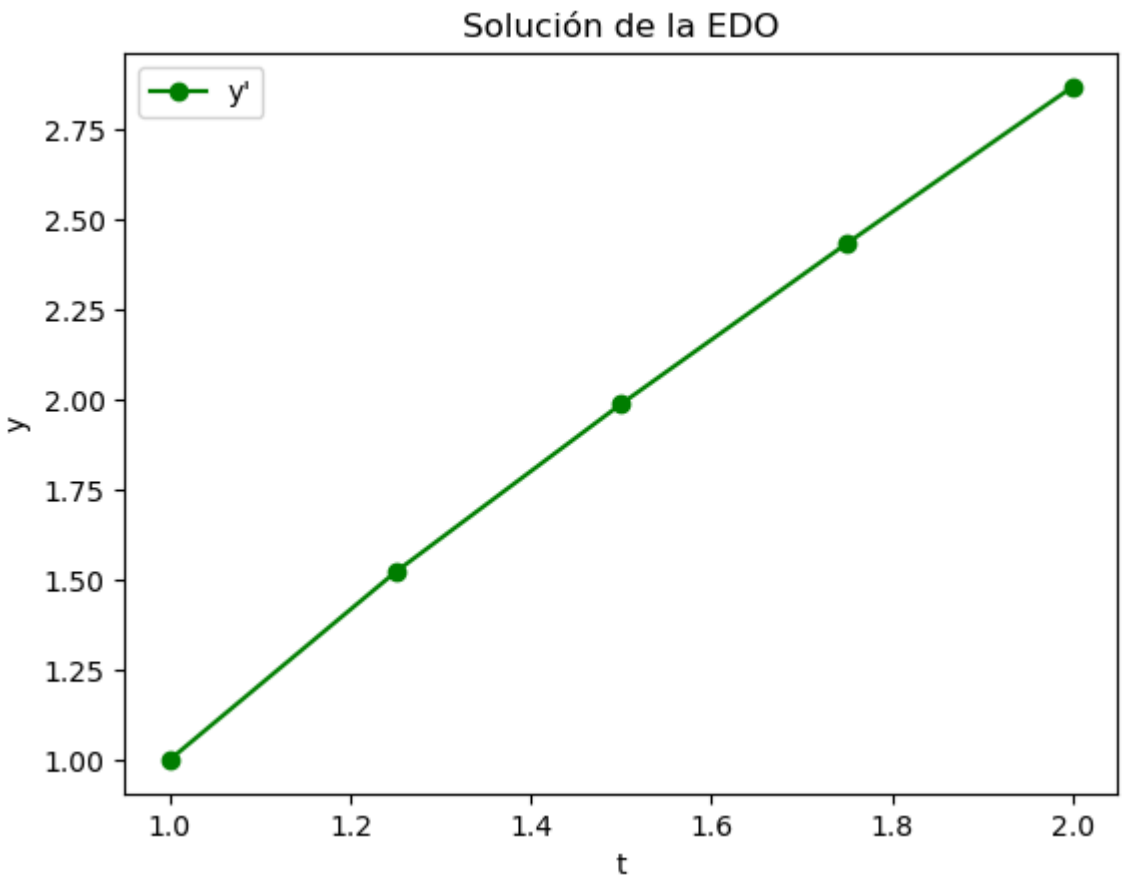
c. $y' = 1 + \frac{y}{t}$, $1 \leq t \leq 2$, $y(1) = 2$, con $h = 0.25$

```
%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

ys4c, ts4c, 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(ts4c, ys4c)
```

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



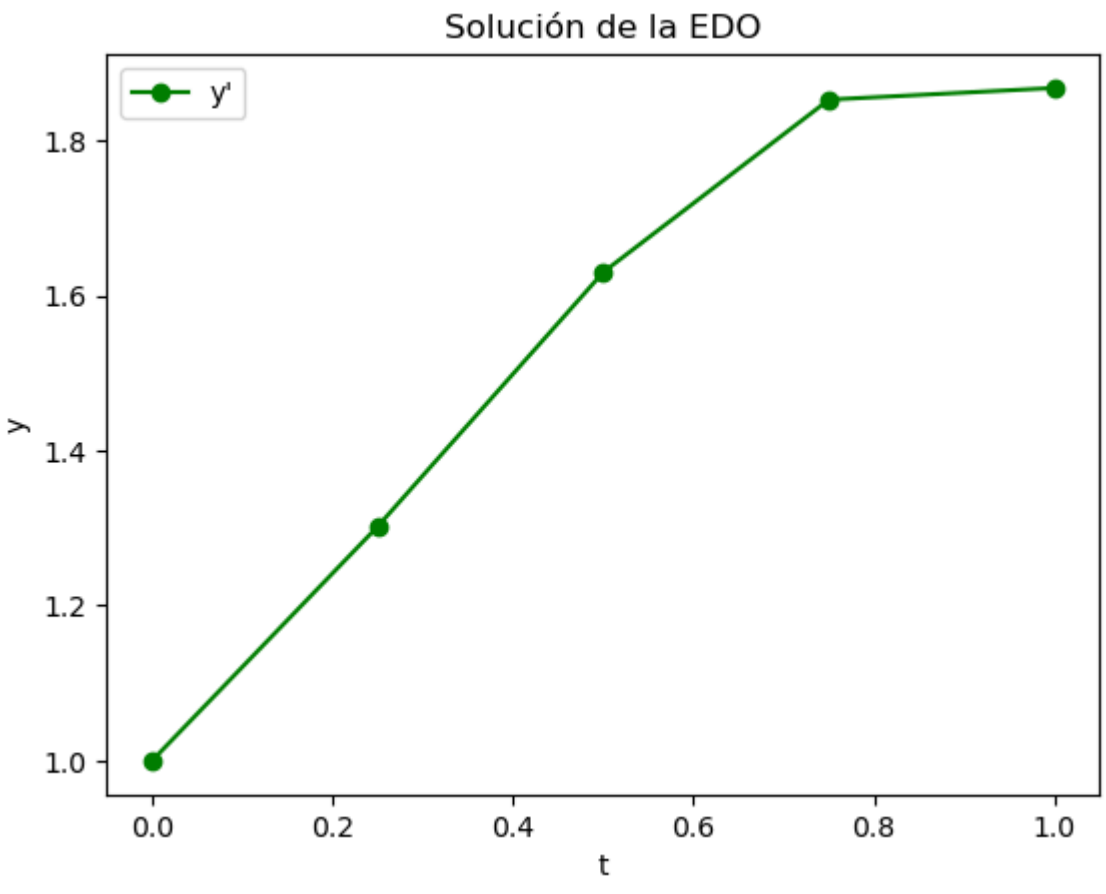
d. $y' = \cos 2t + \sin 3t$, $0 \leq t \leq 1$, $y(0) = 1$, con $h = 0.25$

```
%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

ys4d, ts4d, 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(ts4d, ys4d)
```

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



Link del repositorio:

https://github.com/ElAlfa3007/M-todos_Num-ricos.git