Método de Euler para EDO's

```python
%load_ext autoreload
%autoreload 2
from src import ODE_euler
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
[02-05 08:00:17][INFO] 2025-02-05 08:00:17.629950
[02-05 08:00:17][INFO] 2025-02-05 08:00:17.743846
[02-05 08:00:17][INFO] 2025-02-05 08:00:17.750344
```

```python
import matplotlib.pyplot as plt
import numpy as np
```

Ejercicio 1

$$y' = y - t^2 + 1$$

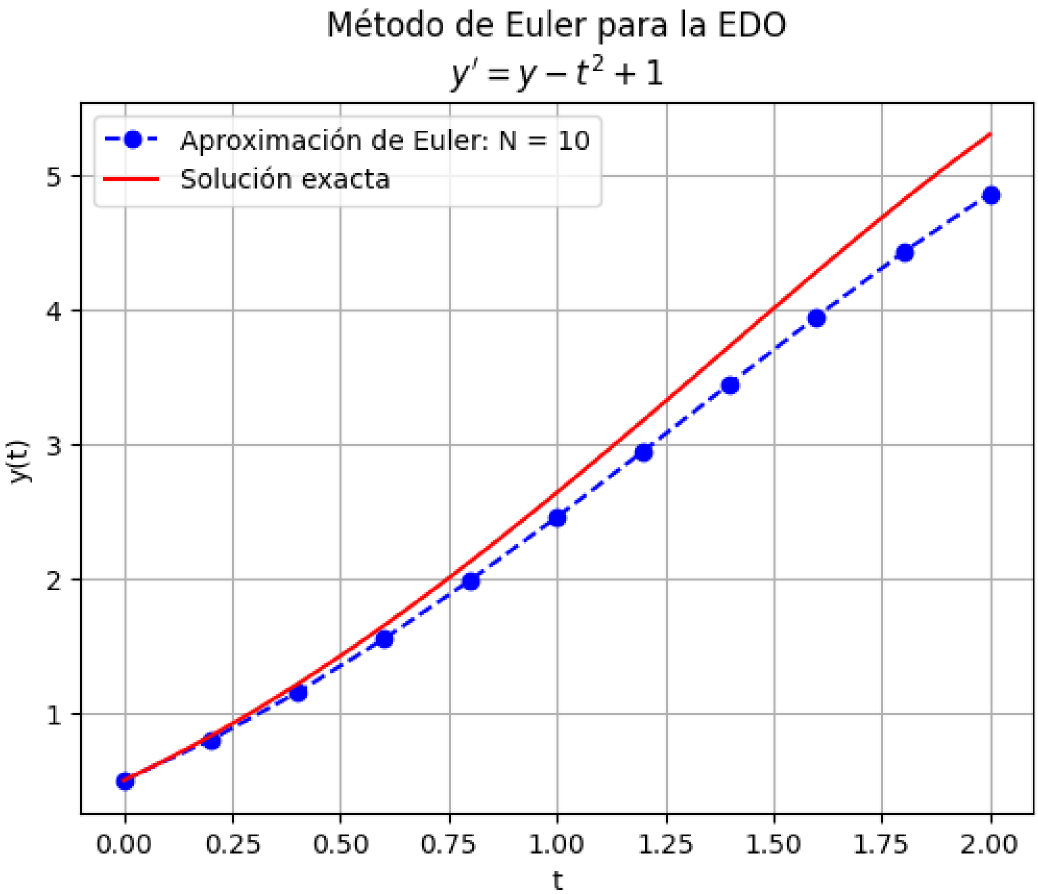
$$0 \leq t \leq 2$$

$$y(t_0) = 0.5$$

```python
# Definimos la ecuación diferencial dy/dt = y - t^2 + 1
def f(t: float, y: float) -> float:
    return y - t**2 + 1


# Solución exacta de la ecuación diferencial
def exact_solution(t: np.ndarray) -> np.ndarray:
    return (t + 1) ** 2 - 0.5 * np.exp(t)
```

Primer Gráfico

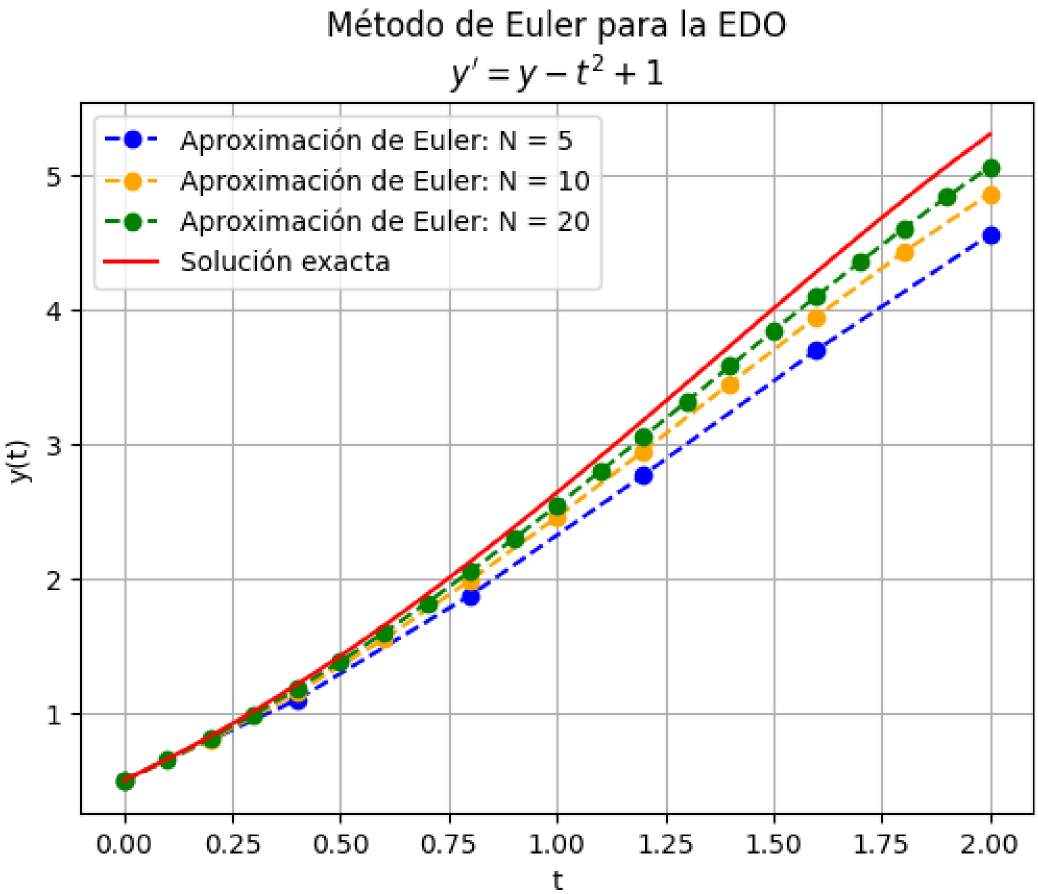
```
# Parámetros iniciales
a, b = 0, 2  # Intervalo de tiempo [a, b]
y_t0 = 0.5  # Condición inicial y(0) = 0.5
N = 10  # Número de puntos

# Ejecutamos el método de Euler
ys, ts, h = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N)

t_exact = np.linspace(a, b, 100)
y_exact = exact_solution(t_exact)

# Graficamos los resultados
plt.plot(
    ts, ys, label="Aproximación de Euler: N = 10", marker="o", linestyle="--", color="b"
)
plt.plot(t_exact, y_exact, label="Solución exacta", linestyle="-", color="red")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Método de Euler para la EDO\n$y' = y - t^2 + 1$")
plt.legend()
plt.grid()
plt.show()
```



Segundo Gráfico

```
# Parámetros iniciales
a, b = 0, 2  # Intervalo de tiempo [a, b]
y_t0 = 0.5  # Condición inicial y(0) = 0.5
N1 = 5  # Número de puntos

# Ejecutamos el método de Euler
ys1, ts1, h1 = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N1)

N2 = 10
ys2, ts2, h2 = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N2)

N3 = 20
```

```python
ys3, ts3, h3 = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N3)

# Solución exacta
t_exact = np.linspace(a, b, 100)
y_exact = exact_solution(t_exact)

# Graficamos los resultados
plt.plot(
    ts1,
    ys1,
    label="Aproximación de Euler: N = 5",
    marker="o",
    linestyle="--",
    color="b",
)
plt.plot(
    ts2,
    ys2,
    label="Aproximación de Euler: N = 10",
    marker="o",
    linestyle="--",
    color="orange",
)
plt.plot(
    ts3,
    ys3,
    label="Aproximación de Euler: N = 20",
    marker="o",
    linestyle="--",
    color="green",
)
plt.plot(t_exact, y_exact, label="Solución exacta", linestyle="-", color="red")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Método de Euler para la EDO\n$y' = y - t^2 + 1$")
plt.legend()
plt.grid()
plt.show()
```



Ejercicio 2

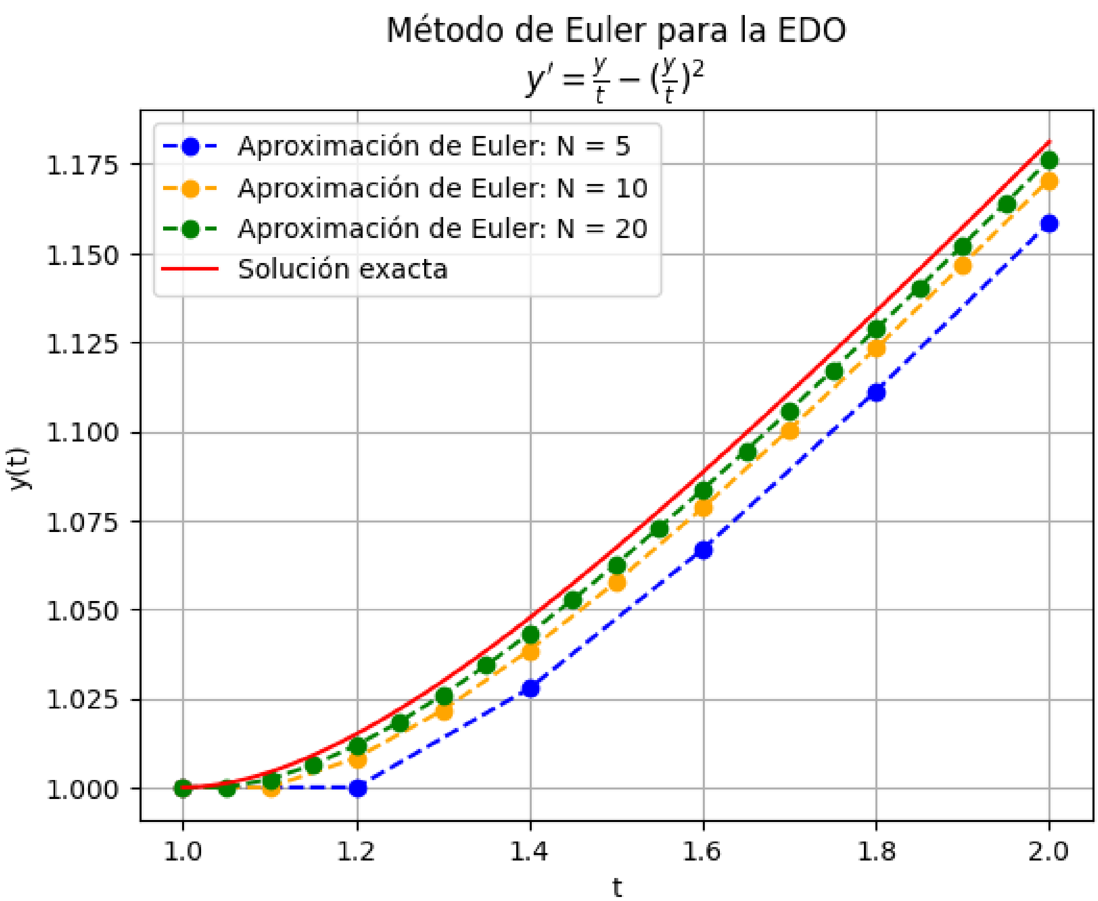
$$y' = \frac{y}{t} - (\frac{y}{t})^2$$
$$1 \leq t \leq 2$$
$$y(t_0) = 1$$

```python
# Definimos la ecuación diferencial
def f2(t: float, y: float) -> float:
    return y / t - (y / t) ** 2


# Solución exacta de la ecuación diferencial
def exact_solution2(t: np.ndarray) -> np.ndarray:
    return t / (1 + np.log(t))
```

Gráfico

```
# Parámetros iniciales
a, b = 1, 2  # Intervalo de tiempo [a, b]
y_t0 = 1  # Condición inicial y(1) = 1
N1 = 5  # Número de puntos

# Ejecutamos el método de Euler
ys1, ts1, h1 = ODE_euler(a=a, b=b, f=f2, y_t0=y_t0, N=N1)

N2 = 10
ys2, ts2, h2 = ODE_euler(a=a, b=b, f=f2, y_t0=y_t0, N=N2)

N3 = 20
ys3, ts3, h3 = ODE_euler(a=a, b=b, f=f2, y_t0=y_t0, N=N3)

# Solución exacta
t_exact = np.linspace(a, b, 100)
y_exact = exact_solution2(t_exact)

# Graficamos los resultados
plt.plot(
    ts1,
    ys1,
    label="Aproximación de Euler: N = 5",
    marker="o",
    linestyle="--",
    color="b",
)
plt.plot(
    ts2,
    ys2,
    label="Aproximación de Euler: N = 10",
    marker="o",
    linestyle="--",
    color="orange",
)
plt.plot(
    ts3,
    ys3,
    label="Aproximación de Euler: N = 20",
    marker="o",
```

```
    linestyle="--",
    color="green",
)
plt.plot(t_exact, y_exact, label="Solución exacta", linestyle="-", color="red")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Método de Euler para la EDO\n$y' = \\frac{y}{t} - (\\frac{y}{t})^2$")
plt.legend()
plt.grid()
plt.show()
```



Conclusión

Mientras el N aumenta, se mejora la precisión. Pero es importante encontrar un balance entre exactitud y eficiencia computacional.