Tarea 12: ODE Método de Euler

Pasquel Johann

Tabla de Contenidos


```
%load_ext autoreload
```

GITHUB

https://github.com/Vladimirjon/MetodosNumericos_PasquelJohann/tree/main/Tarea12

CONJUNTO DE EJERCICIOS

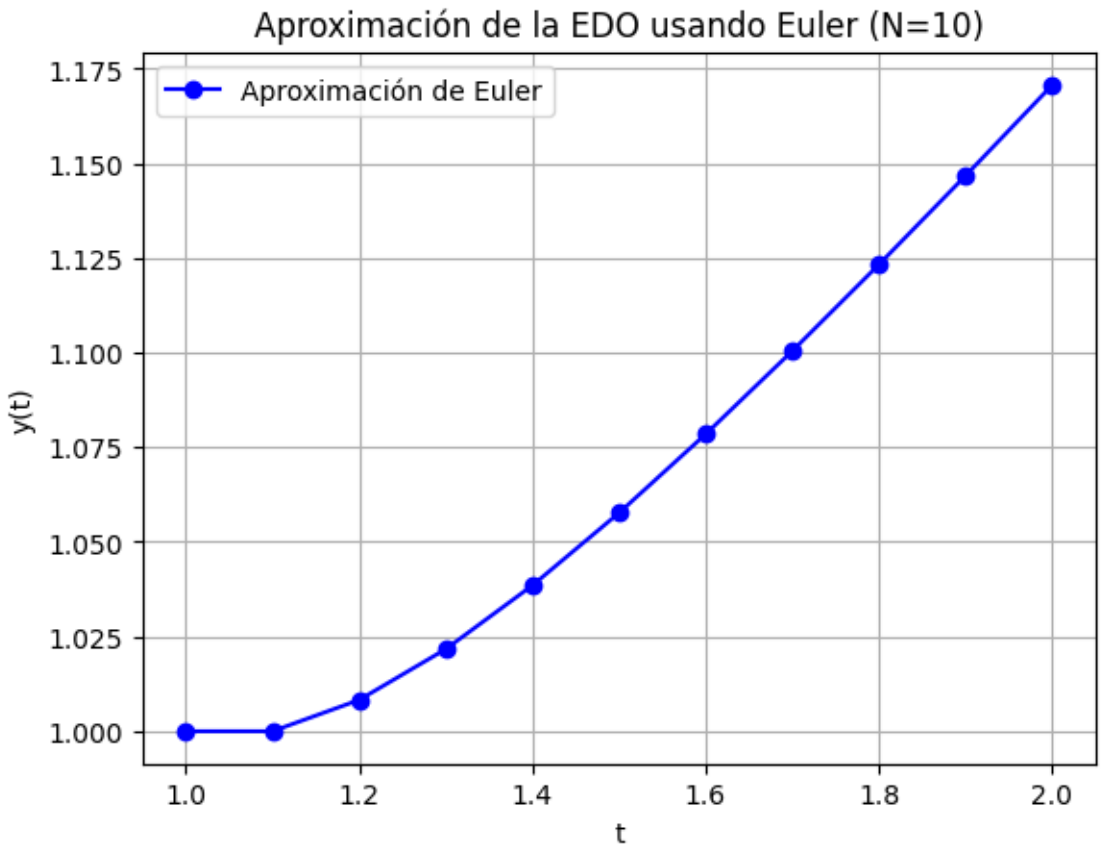
3. Utilice el método de Euler para aproximar las soluciones para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

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

```
%autoreload 2
import pandas as pd
import matplotlib.pyplot as plt
from src.ODE import ODE_euler  # Importar el método de Euler
```

```
def f(t, y):
    return (y / t) - (y / t) ** 2  # Definir la ecuación diferencial

# Parámetros del problema
a = 1        # Inicio del intervalo
b = 2        # Fin del intervalo
y_t0 = 1     # Condición inicial
N = 10       # Número de pasos

# Resolver la EDO usando el método de Euler
ys, ts, h = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N)

# Crear DataFrame con número de iteración, t_i y y_aprox
df = pd.DataFrame({"i": range(N+1), "t_i": ts, "y_aprox": ys})

# Mostrar la tabla
display(df)

# Graficar la solución aproximada
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación de Euler")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Aproximación de la EDO usando Euler (N=10)")
plt.legend()
plt.grid()
plt.show()
```

	i	t_i	y_aprox
0	0	1.0	1.000000
1	1	1.1	1.000000
2	2	1.2	1.008264
3	3	1.3	1.021689
4	4	1.4	1.038515
5	5	1.5	1.057668
6	6	1.6	1.078461
7	7	1.7	1.100432
8	8	1.8	1.123262
9	9	1.9	1.146724
10	10	2.0	1.170652

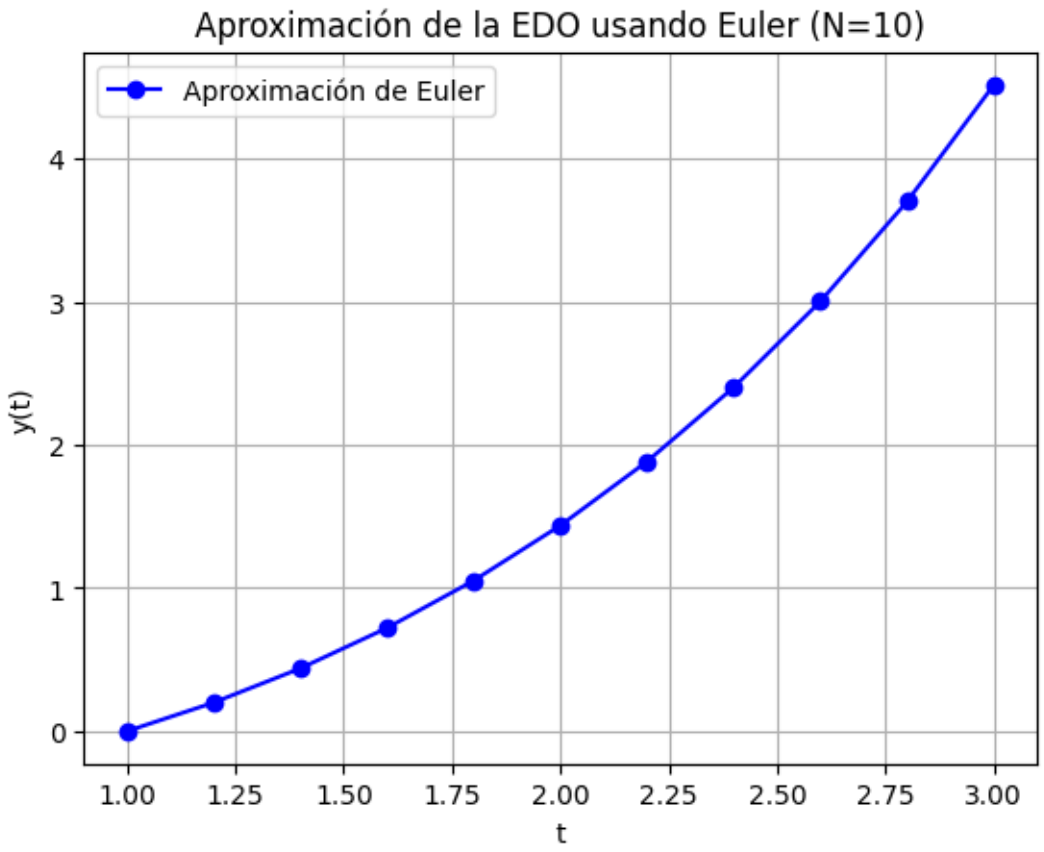


b. $y' = 1 + \frac{y}{t} + (\frac{y}{t})^2$, $1 \leq t \leq 3$,$y(1) = 0$, con $h = 0.2$

```
%autoreload 2
import pandas as pd
import matplotlib.pyplot as plt
from src.ODE import ODE_euler  # Importar el método de Euler

def f(t, y):
    return 1 + (y / t) + (y / t) ** 2  # Definir la ecuación diferencial

# Parámetros del problema
a = 1        # Inicio del intervalo
b = 3        # Fin del intervalo
y_t0 = 0     # Condición inicial
N = 10       # Número de pasos

# Resolver la EDO usando el método de Euler
ys, ts, h = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N)
```

```
# Crear DataFrame con número de iteración, t_i y y_aprox
df = pd.DataFrame({"i": range(N+1), "t_i": ts, "y_aprox": ys})

# Mostrar la tabla
display(df)

# Graficar la solución aproximada
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación de Euler")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Aproximación de la EDO usando Euler (N=10)")
plt.legend()
plt.grid()
plt.show()
```

	i	t_i	y_aprox
0	0	1.0	0.000000
1	1	1.2	0.200000
2	2	1.4	0.438889
3	3	1.6	0.721243
4	4	1.8	1.052038
5	5	2.0	1.437251
6	6	2.2	1.884261
7	7	2.4	2.402270
8	8	2.6	3.002837
9	9	2.8	3.700601
10	10	3.0	4.514277

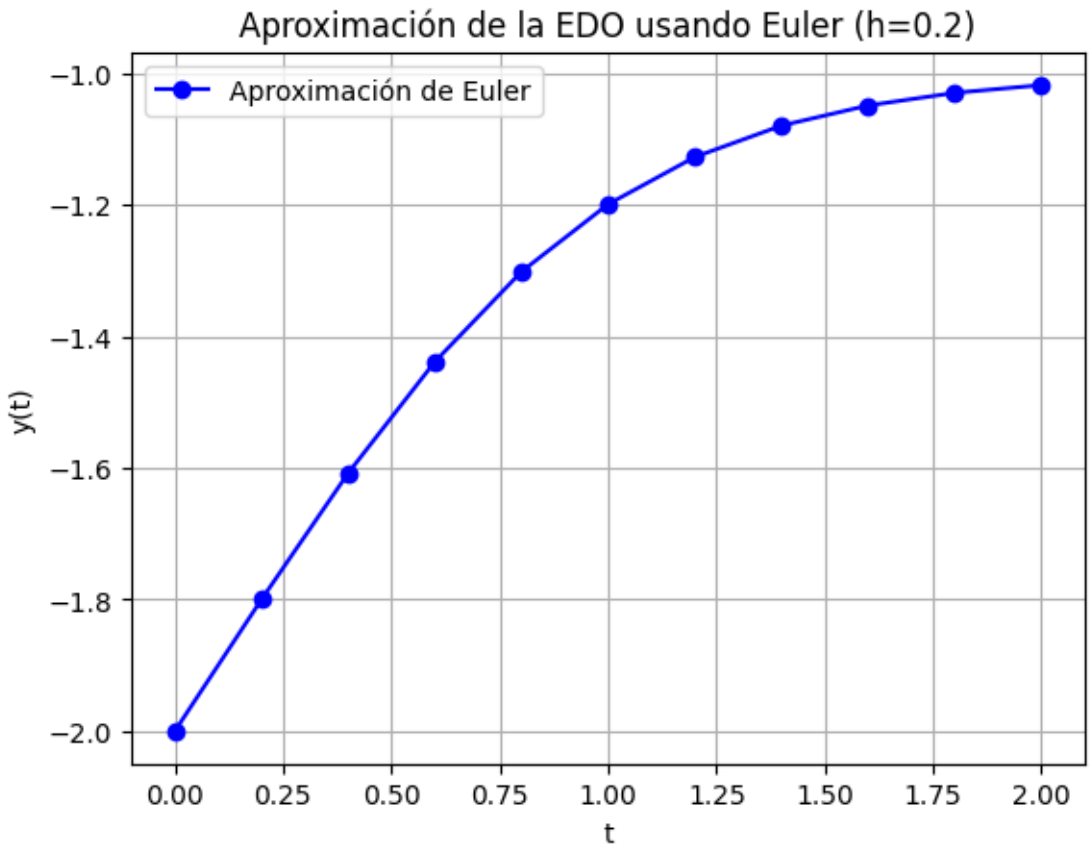


c. $y' = -(y+1)(y+3)$, $0 \leq t \leq 2$,$y(0) = -2$, con $h = 0.2$

```
%autoreload 2
import pandas as pd
import matplotlib.pyplot as plt
from src.ODE import ODE_euler  # Importar el método de Euler

# Definir la ecuación diferencial dada
def f(t, y):
    return - (y + 1) * (y + 3)  # Uso correcto de multiplicación

# Parámetros del problema
a = 0        # Inicio del intervalo
b = 2        # Fin del intervalo
y_t0 = -2    # Condición inicial
h = 0.2      # Tamaño de paso
N = int((b - a) / h)  # Calcular el número de pasos según h
```

```
# Resolver la EDO usando el método de Euler
ys, ts, h = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N)

# Crear DataFrame con número de iteración, t_i y y_aprox
df = pd.DataFrame({"i": range(N+1), "t_i": ts, "y_aprox": ys})

# Mostrar la tabla
display(df)

# Graficar la solución aproximada
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación de Euler")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Aproximación de la EDO usando Euler (h=0.2)")
plt.legend()
plt.grid()
plt.show()
```

	i	t_i	y_aprox
0	0	0.0	-2.000000
1	1	0.2	-1.800000
2	2	0.4	-1.608000
3	3	0.6	-1.438733
4	4	0.8	-1.301737
5	5	1.0	-1.199251
6	6	1.2	-1.127491
7	7	1.4	-1.079745
8	8	1.6	-1.049119
9	9	1.8	-1.029954
10	10	2.0	-1.018152

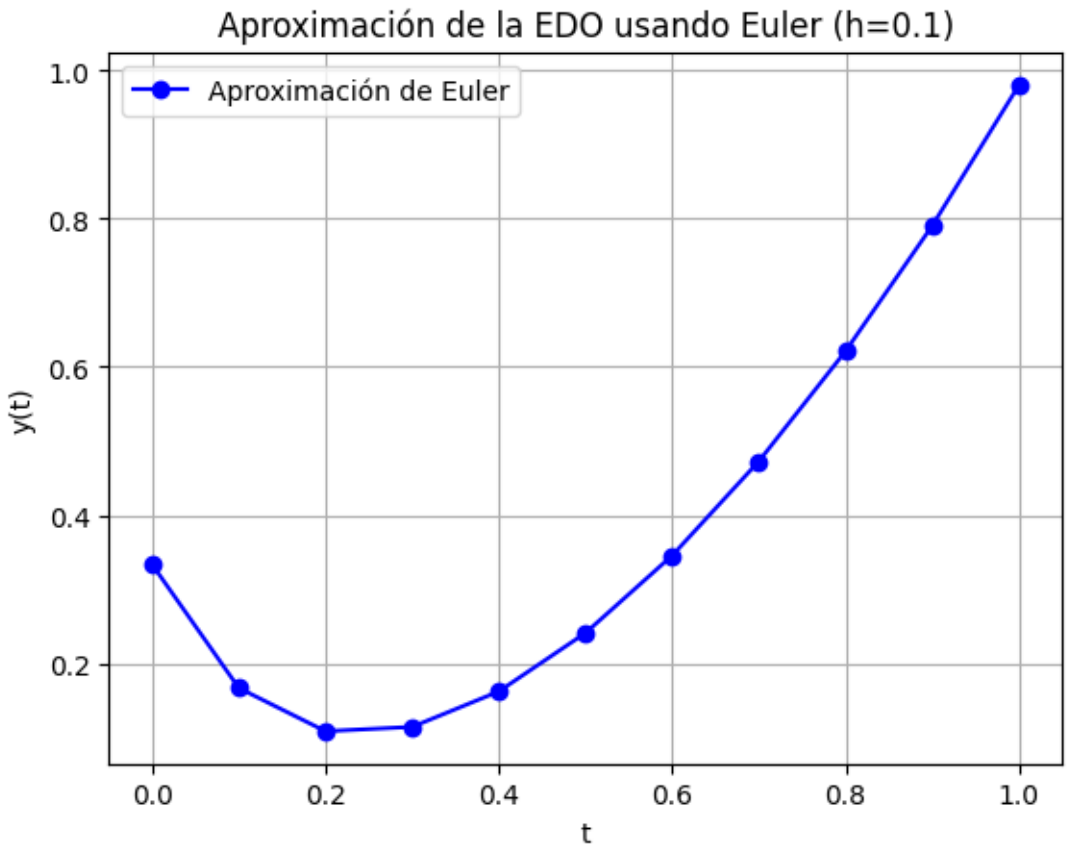


d. $y' = -5y + 5t^2 + 2t,\ 0 \leq t \leq 1, y(0) = \frac{1}{3}$, con $h = 0.1$

```
%autoreload 2
import pandas as pd
import matplotlib.pyplot as plt
from src.ODE import ODE_euler  # Importar el método de Euler

# Definir la ecuación diferencial dada
def f(t, y):
    return -5*y + 5*t**2 + 2*t  # Definir correctamente la ecuación

# Parámetros del problema
a = 0        # Inicio del intervalo
b = 1        # Fin del intervalo
y_t0 = 1/3   # Condición inicial y(0) = 1/3
h = 0.1      # Tamaño de paso
N = int((b - a) / h)  # Calcular el número de pasos basado en h
```

```
# Resolver la EDO usando el método de Euler
ys, ts, h = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, N=N)

# Crear DataFrame con número de iteración, t_i y y_aprox
df = pd.DataFrame({"i": range(N+1), "t_i": ts, "y_aprox": ys})

# Mostrar la tabla
display(df)

# Graficar la solución aproximada
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación de Euler")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Aproximación de la EDO usando Euler (h=0.1)")
plt.legend()
plt.grid()
plt.show()
```

	i	t_i	y_aprox
0	0	0.0	0.333333
1	1	0.1	0.166667
2	2	0.2	0.108333
3	3	0.3	0.114167
4	4	0.4	0.162083
5	5	0.5	0.241042
6	6	0.6	0.345521
7	7	0.7	0.472760
8	8	0.8	0.621380
9	9	0.9	0.790690
10	10	1.0	0.980345



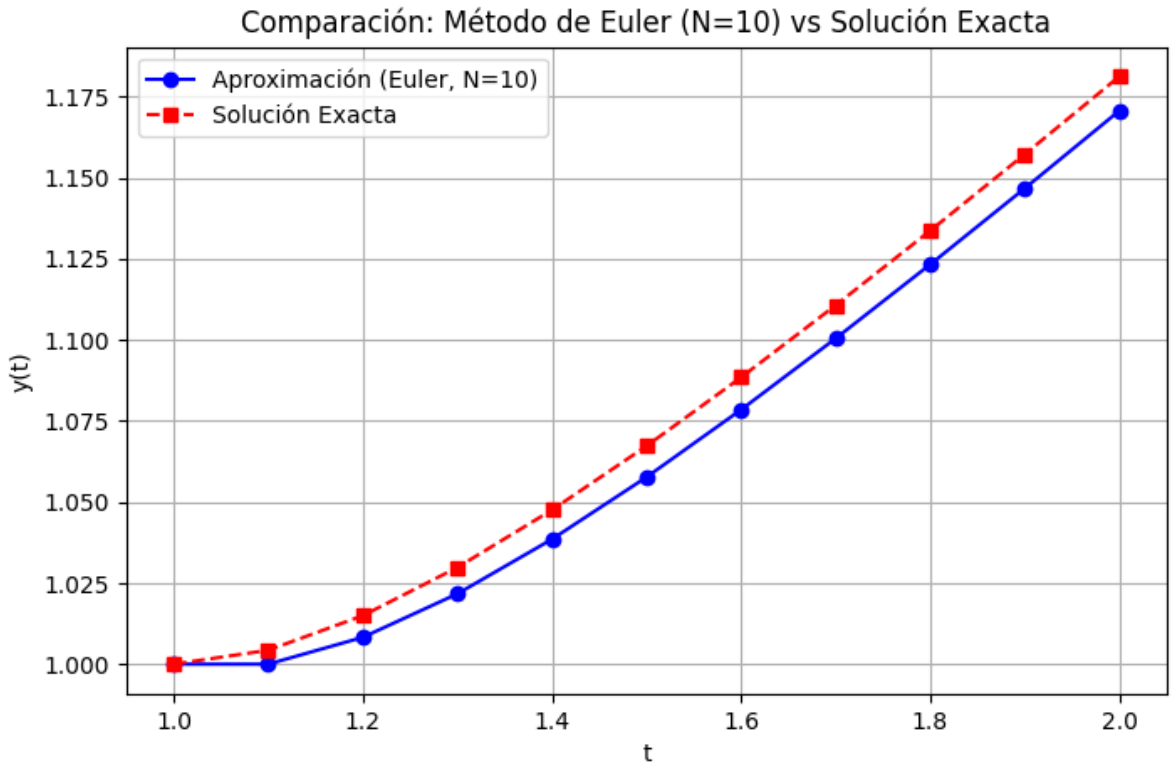
4. Calcular el error real en las aproximaciones del ejercicio anterior.

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

Gráfica real vs aproximación

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

# Definir la solución exacta de la EDO
def y_real(t):
    return t / (1 + np.log(t))  # Nueva solución exacta corregida

# Calcular los valores exactos en los mismos puntos de malla
y_exact = [y_real(t) for t in ts]

# Graficar la comparación
```

```
plt.figure(figsize=(8, 5))
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación (Euler, N=10)")
plt.plot(ts, y_exact, marker="s", linestyle="--", color="r", label="Solución Exacta")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Comparación: Método de Euler (N=10) vs Solución Exacta")
plt.legend()
plt.grid()
plt.show()
```



Error real en las aproximaciones

```
import numpy as np
import pandas as pd

# Calcular el error absoluto |y_real - y_aprox| en cada punto
error_abs = [abs(y_r - y_a) for y_r, y_a in zip(y_exact, ys)]

# Crear DataFrame con los valores
```

```
df_error = pd.DataFrame({
    "i": range(len(ts)),
    "t_i": ts,
    "Euler (N=10)": ys,
    "Solución Exacta": y_exact,
    "Error Absoluto": error_abs
})

# Mostrar la tabla
display(df_error)
```

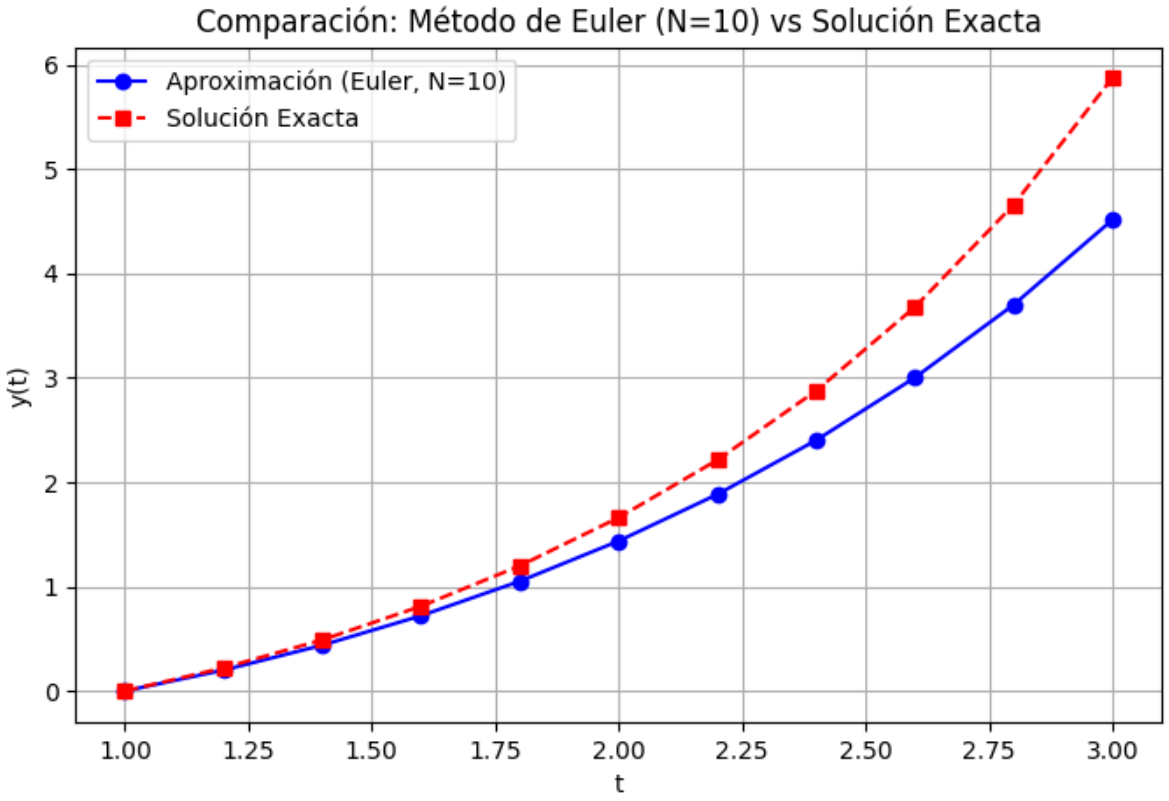
	i	t_i	Euler (N=10)	Solución Exacta	Error Absoluto
0	0	1.0	1.000000	1.000000	0.000000
1	1	1.1	1.000000	1.004282	0.004282
2	2	1.2	1.008264	1.014952	0.006688
3	3	1.3	1.021689	1.029814	0.008124
4	4	1.4	1.038515	1.047534	0.009019
5	5	1.5	1.057668	1.067262	0.009594
6	6	1.6	1.078461	1.088433	0.009972
7	7	1.7	1.100432	1.110655	0.010223
8	8	1.8	1.123262	1.133654	0.010392
9	9	1.9	1.146724	1.157228	0.010505
10	10	2.0	1.170652	1.181232	0.010581

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

Gráfica real vs aproximación

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

# Definir la nueva solución exacta de la EDO
def y_real(t):
    return t * np.tan(np.log(t))  # Nueva función exacta

# Calcular los valores exactos en los mismos puntos de malla
y_exact = [y_real(t) for t in ts]

# Graficar la comparación
plt.figure(figsize=(8, 5))
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación (Euler, N=10)")
```

```
plt.plot(ts, y_exact, marker="s", linestyle="--", color="r", label="Solución Exacta")
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Comparación: Método de Euler (N=10) vs Solución Exacta")
plt.legend()
plt.grid()
plt.show()
```



Error real en las aproximaciones

```
import numpy as np
import pandas as pd

# Calcular el error absoluto |y_real - y_aprox| en cada punto
error_abs = [abs(y_r - y_a) for y_r, y_a in zip(y_exact, ys)]

# Crear DataFrame con los valores
df_error = pd.DataFrame({
```

```
    "i": range(len(ts)),
    "t_i": ts,
    "Euler (N=10)": ys,
    "Solución Exacta": y_exact,
    "Error Absoluto": error_abs
})

# Mostrar la tabla
display(df_error)
```

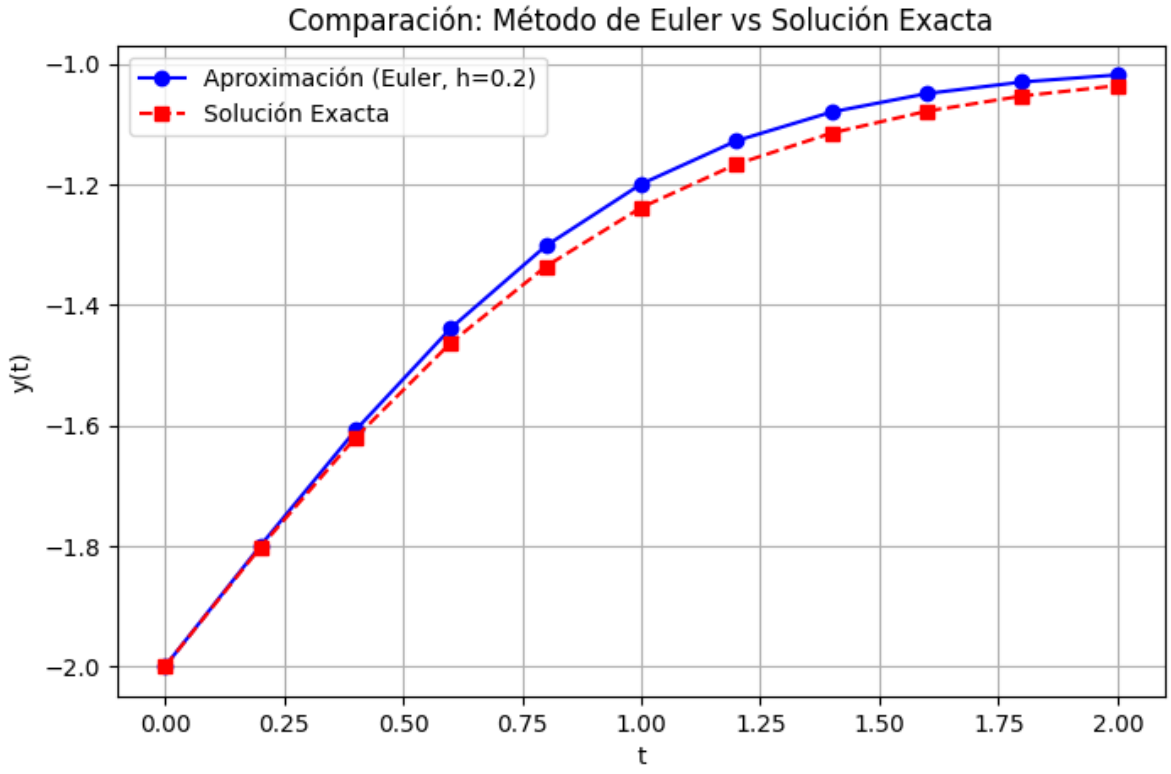
	i	t_i	Euler (N=10)	Solución Exacta	Error Absoluto
0	0	1.0	0.000000	0.000000	0.000000
1	1	1.2	0.200000	0.221243	0.021243
2	2	1.4	0.438889	0.489682	0.050793
3	3	1.6	0.721243	0.812753	0.091510
4	4	1.8	1.052038	1.199439	0.147401
5	5	2.0	1.437251	1.661282	0.224031
6	6	2.2	1.884261	2.213502	0.329241
7	7	2.4	2.402270	2.876551	0.474282
8	8	2.6	3.002837	3.678475	0.675638
9	9	2.8	3.700601	4.658665	0.958064
10	10	3.0	4.514277	5.874100	1.359823

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

Gráfica real vs aproximación

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

# Definir la nueva solución exacta de la EDO
def y_real(t):
    return -3 + (2 / (1 + np.exp(-2*t)))  # Nueva función exacta

# Calcular los valores exactos en los mismos puntos de malla
y_exact = [y_real(t) for t in ts]

# Graficar la comparación
plt.figure(figsize=(8, 5))
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación (Euler, h=0.2)")
plt.plot(ts, y_exact, marker="s", linestyle="--", color="r", label="Solución Exacta")
```

```
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Comparación: Método de Euler vs Solución Exacta")
plt.legend()
plt.grid()
plt.show()
```



Error real en las aproximaciones

```
import numpy as np
import pandas as pd

# Calcular el error absoluto |y_real - y_aprox| en cada punto
error_abs = [abs(y_r - y_a) for y_r, y_a in zip(y_exact, ys)]

# Crear DataFrame con los valores
df_error = pd.DataFrame({
    "i": range(len(ts)),
    "t_i": ts,
```

```
    "Euler (N=10)": ys,
    "Solución Exacta": y_exact,
    "Error Absoluto": error_abs
})

# Mostrar la tabla
display(df_error)
```

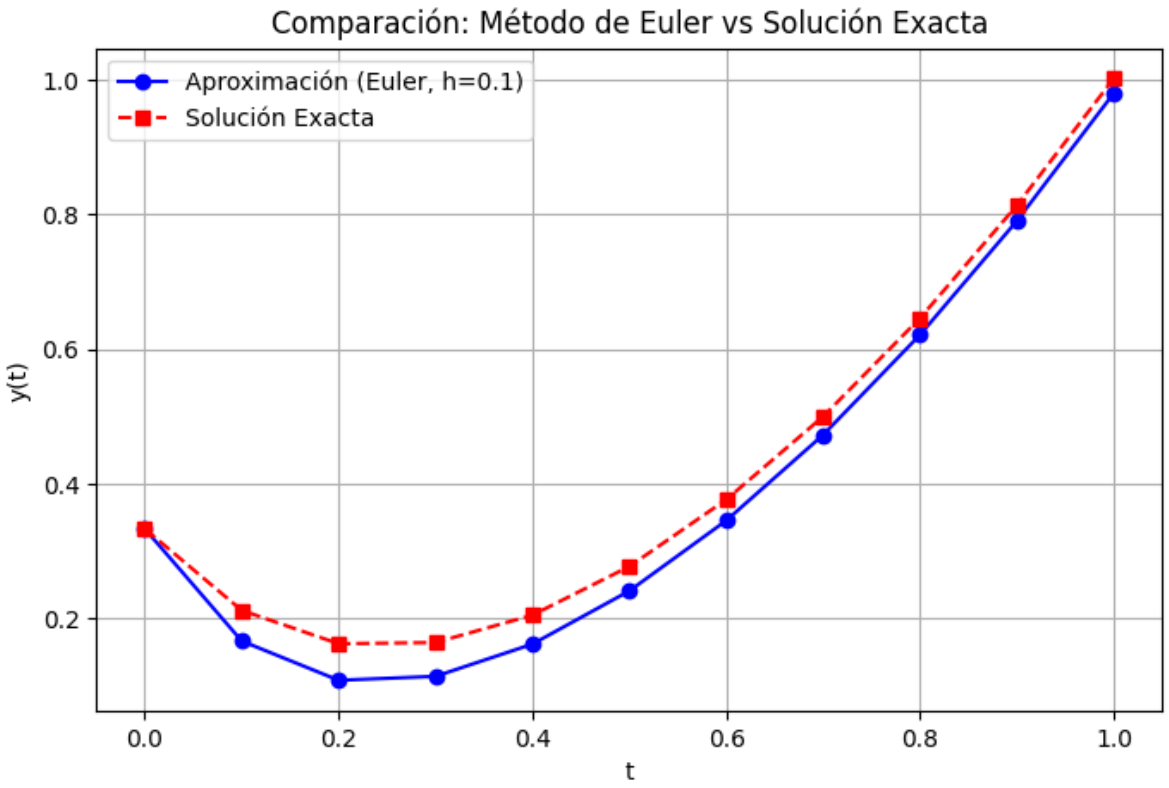
	i	t_i	Euler (N=10)	Solución Exacta	Error Absoluto
0	0	0.0	-2.000000	-2.000000	0.000000
1	1	0.2	-1.800000	-1.802625	0.002625
2	2	0.4	-1.608000	-1.620051	0.012051
3	3	0.6	-1.438733	-1.462950	0.024218
4	4	0.8	-1.301737	-1.335963	0.034226
5	5	1.0	-1.199251	-1.238406	0.039155
6	6	1.2	-1.127491	-1.166345	0.038854
7	7	1.4	-1.079745	-1.114648	0.034903
8	8	1.6	-1.049119	-1.078331	0.029212
9	9	1.8	-1.029954	-1.053194	0.023240
10	10	2.0	-1.018152	-1.035972	0.017821

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

Gráfica real vs aproximación

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

# Definir la nueva solución exacta de la EDO
def y_real(t):
    return t**2 + (1/3) * np.exp(-5*t)  # Nueva función exacta

# Calcular los valores exactos en los mismos puntos de malla
y_exact = [y_real(t) for t in ts]

# Graficar la comparación
plt.figure(figsize=(8, 5))
plt.plot(ts, ys, marker="o", linestyle="-", color="b", label="Aproximación (Euler, h=0.1)")
plt.plot(ts, y_exact, marker="s", linestyle="--", color="r", label="Solución Exacta")
plt.xlabel("t")
plt.ylabel("y(t)")
```

```
plt.title("Comparación: Método de Euler vs Solución Exacta")
plt.legend()
plt.grid()
plt.show()
```



Error real en las aproximaciones

```
import numpy as np
import pandas as pd

# Calcular el error absoluto |y_real - y_aprox| en cada punto
error_abs = [abs(y_r - y_a) for y_r, y_a in zip(y_exact, ys)]

# Crear DataFrame con los valores
df_error = pd.DataFrame({
    "i": range(len(ts)),
    "t_i": ts,
    "Euler (N=10)": ys,
    "Solución Exacta": y_exact,
```

```
    "Error Absoluto": error_abs
})

# Mostrar la tabla
display(df_error)
```

	i	t_i	Euler (N=10)	Solución Exacta	Error Absoluto
0	0	0.0	0.333333	0.333333	0.000000
1	1	0.1	0.166667	0.212177	0.045510
2	2	0.2	0.108333	0.162626	0.054293
3	3	0.3	0.114167	0.164377	0.050210
4	4	0.4	0.162083	0.205112	0.043028
5	5	0.5	0.241042	0.277362	0.036320
6	6	0.6	0.345521	0.376596	0.031075
7	7	0.7	0.472760	0.500066	0.027305
8	8	0.8	0.621380	0.646105	0.024725
9	9	0.9	0.790690	0.813703	0.023013
10	10	1.0	0.980345	1.002246	0.021901

5. Interpolación lineal

Con los resultados del ejercicio 3, interpolar linealmente 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** $y(0.93)$

```
from scipy.interpolate import interp1d
import pandas as pd

# Crear interpolación lineal a partir de los datos de Euler
interp_euler = interp1d(ts, ys, kind='linear', fill_value="extrapolate")

# Valores a interpolar
t_values = [0.25, 0.93]

# Calcular valores interpolados usando Euler
y_interpolated = interp_euler(t_values)

# Calcular valores reales usando la solución exacta
```

```
y_real_values = [y_real(t) for t in t_values]

# Calcular el error absoluto entre la interpolación y la solución exacta
error_interpolado = [abs(y_r - y_i) for y_r, y_i in zip(y_real_values, y_interpolated)]

# Crear DataFrame con los resultados
df_interpolacion = pd.DataFrame({
    "t": t_values,
    "Interpolación Euler": y_interpolated,
    "Solución Exacta": y_real_values,
    "Error Absoluto": error_interpolado
})

# Mostrar la tabla con los valores interpolados y comparación con la solución exacta
display(df_interpolacion)
```

	t	Interpolación Euler	Solución Exacta	Error Absoluto
0	0.25	1.0	-0.647175	1.647175
1	0.93	1.0	1.002772	0.002772

b.$y(t) = y(1.25)$ **y** $y(1.93)$

```
from scipy.interpolate import interp1d
import numpy as np
import pandas as pd

# Crear interpolación lineal a partir de los datos de Euler para el segundo caso
interp_euler_b = interp1d(ts, ys, kind='linear', fill_value="extrapolate")

# Valores a interpolar
t_values_b = [1.25, 1.93]

# Calcular valores interpolados usando Euler
y_interpolated_b = interp_euler_b(t_values_b)

# Calcular valores reales usando la solución exacta de este caso
def y_real_b(t):
    return t * np.tan(np.log(t))  # Nueva función exacta

y_real_values_b = [y_real_b(t) for t in t_values_b]
```

```
# Calcular el error absoluto entre la interpolación y la solución exacta
error_interpolado_b = [abs(y_r - y_i) for y_r, y_i in zip(y_real_values_b, y_interpolated_b)]

# Crear DataFrame con los resultados
df_interpolacion_b = pd.DataFrame({
    "t": t_values_b,
    "Interpolación Euler": y_interpolated_b,
    "Solución Exacta": y_real_values_b,
    "Error Absoluto": error_interpolado_b
})

# Mostrar la tabla con los valores interpolados y comparación con la solución exacta
display(df_interpolacion_b)
```

	t	Interpolación Euler	Solución Exacta	Error Absoluto
0	1.25	0.259722	0.283653	0.023931
1	1.93	1.302427	1.490228	0.187801

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

```
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d

# Definir la solución exacta de la EDO
def y_real_c(t):
    return -3 + (2 / (1 + np.exp(-2*t)))  # Nueva función exacta

# Crear interpolación lineal a partir de los datos de Euler para el tercer caso
interp_euler_c = interp1d(ts, ys, kind='linear', fill_value="extrapolate")

# Valores a interpolar
t_values_c = [2.10, 2.75]

# Calcular valores interpolados usando Euler
y_interpolated_c = interp_euler_c(t_values_c)

# Calcular valores reales usando la solución exacta
y_real_values_c = [y_real_c(t) for t in t_values_c]
```

```
# Calcular el error absoluto entre la interpolación y la solución exacta
error_interpolado_c = [abs(y_r - y_i) for y_r, y_i in zip(y_real_values_c, y_interpolated_c)]

# Crear DataFrame con los resultados
df_interpolacion_c = pd.DataFrame({
    "t": t_values_c,
    "Interpolación Euler": y_interpolated_c,
    "Solución Exacta": y_real_values_c,
    "Error Absoluto": error_interpolado_c
})

# Mostrar la tabla con los valores interpolados y comparación con la solución exacta
display(df_interpolacion_c)
```

	t	Interpolación Euler	Solución Exacta	Error Absoluto
0	2.10	-1.012251	-1.029548	0.017297
1	2.75	-0.973894	-1.008140	0.034246

d.$y(t) = y(0.54)$ **y** $y(0.94)$

```
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d

# Definir la solución exacta de la EDO
def y_real_d(t):
    return t**2 + (1/3) * np.exp(-5*t)  # Nueva función exacta

# Crear interpolación lineal a partir de los datos de Euler para el cuarto caso
interp_euler_d = interp1d(ts, ys, kind='linear', fill_value="extrapolate")

# Valores a interpolar
t_values_d = [0.54, 0.94]

# Calcular valores interpolados usando Euler
y_interpolated_d = interp_euler_d(t_values_d)

# Calcular valores reales usando la solución exacta
y_real_values_d = [y_real_d(t) for t in t_values_d]
```

```python
# Calcular el error absoluto entre la interpolación y la solución exacta
error_interpolado_d = [abs(y_r - y_i) for y_r, y_i in zip(y_real_values_d, y_interpolated_d)]

# Crear DataFrame con los resultados
df_interpolacion_d = pd.DataFrame({
    "t": t_values_d,
    "Interpolación Euler": y_interpolated_d,
    "Solución Exacta": y_real_values_d,
    "Error Absoluto": error_interpolado_d
})

# Mostrar la tabla con los valores interpolados y comparación con la solución exacta
display(df_interpolacion_d)
```

	t	Interpolación Euler	Solución Exacta	Error Absoluto
0	0.54	0.282833	0.314002	0.031169
1	0.94	0.866552	0.886632	0.020080