Métodos Numericos

ODE Método de Euler

José Sarango

Tabla de Contenidos

Conjunto de ejercicios

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

```
import logging
from sys import stdout
from datetime import datetime
from typing import Callable
from math import exp, cos, sin,tan,log

logging.basicConfig(
    level=logging.INFO,
    format="[%(asctime)s][%(levelname)s] %(message)s",
    stream=stdout,
    datefmt="%m-%d %H:%M:%S",
)

logging.info(datetime.now())

def ODE_euler(
    *,
    a: float,
    b: float,
    f: Callable[[float, float], float],
```

```python
    y_t0: float,
    h: float,
) -> tuple[list[float], list[float]]:
    N = int((b - a) / h)
    t = a
    ts = [t]
    ys = [y_t0]

    for _ in range(N):
        y = ys[-1]
        y += h * f(t, y)
        ys.append(y)

        t += h
        ts.append(t)
    return ys, ts
```

```
[08-09 17:22:20][INFO] 2024-08-09 17:22:20.578977
```

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

```python
def problem_a(t, y):
    return t * exp(3 * t) - 2 * y

a = 0.0
b = 1.0
y_t0 = 0.0
h = 0.5
ys_a, ts_a = ODE_euler(a=a, b=b, f=problem_a, y_t0=y_t0, h=h)
logging.info(f"Aproximaciones por el metodo de euler:")
for t, y in zip(ts_a, ys_a):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:21][INFO] Aproximaciones por el metodo de euler:
[08-09 17:22:21][INFO] t = 0.00, y = 0.0000
[08-09 17:22:21][INFO] t = 0.50, y = 0.0000
[08-09 17:22:21][INFO] t = 1.00, y = 1.1204
```

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

```
def problem_b(t, y):
    return 1 + (t - y)**2
a = 2.0
b = 3.0
y_t0 = 1.0
h = 0.5
ys_b, ts_b = ODE_euler(a=a, b=b, f=problem_b, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_b, ys_b):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:22][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:22][INFO] t = 2.00, y = 1.0000
[08-09 17:22:22][INFO] t = 2.50, y = 2.0000
[08-09 17:22:22][INFO] t = 3.00, y = 2.6250
```

c) $y' = 1 + y/t, 1 \le t \le 2, y(1) = 2$**, con $ h = 0.25 $**

```
def problem_c(t, y):
    return 1 + y / t

a = 1.0
b = 2.0
y_t0 = 2.0
h = 0.25

ys_c, ts_c = ODE_euler(a=a, b=b, f=problem_c, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_c, ys_c):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:22][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:22][INFO] t = 1.00, y = 2.0000
[08-09 17:22:22][INFO] t = 1.25, y = 2.7500
```

```
[08-09 17:22:22][INFO] t = 1.50, y = 3.5500
[08-09 17:22:22][INFO] t = 1.75, y = 4.3917
[08-09 17:22:22][INFO] t = 2.00, y = 5.2690
```

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

```
def problem_d(t, y):
    return cos(2 * t) + sin(3 * t)

# Parámetros
a = 0.0
b = 1.0
y_t0 = 1.0
h = 0.25
ys_d, ts_d = ODE_euler(a=a, b=b, f=problem_d, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_d, ys_d):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:23][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:23][INFO] t = 0.00, y = 1.0000
[08-09 17:22:23][INFO] t = 0.25, y = 1.2500
[08-09 17:22:23][INFO] t = 0.50, y = 1.6398
[08-09 17:22:23][INFO] t = 0.75, y = 2.0243
[08-09 17:22:23][INFO] t = 1.00, y = 2.2365
```

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) $y(t) = 1/5te^{3t} - 1/25e^{3t} + 1/25e^{-2t}$

```
def problem_a(t, y):
    return t * exp(3 * t) - 2 * y

def exact_a(t):
    return (1 / 5) * t * exp(3 * t) - (1 / 25) * exp(3 * t) + (1 / 25) * exp(-2 * t)
a = 0.0
```

```python
b = 1.0
y_t0 = 0.0
h = 0.5
ys_a, ts_a = ODE_euler(a=a, b=b, f=problem_a, y_t0=y_t0, h=h)
logging.info(f"Resultados del problema (a) usando el método de Euler:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_relativo = abs((exact_y - y) / exact_y) if exact_y != 0 else float('inf')
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error relativo = {error_
```

```
[08-09 17:22:25][INFO] Resultados del problema (a) usando el método de Euler:
[08-09 17:22:25][INFO] t = 0.00, y = 0.0000, exacta = 0.0000, error relativo = inf
[08-09 17:22:25][INFO] t = 0.50, y = 0.0000, exacta = 0.2836, error relativo = 1.0000
[08-09 17:22:25][INFO] t = 1.00, y = 1.1204, exacta = 3.2191, error relativo = 0.6519
```

```python
f = lambda t, y: t * exp(3 * t) - 2 * y
exact_a = lambda t: (1 / 5) * t * exp(3 * t) - (1 / 25) * exp(3 * t) + (1 / 25) * exp(-2 * t)
a = 0
b = 1
y_t0 =0
h = 0.5
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
logging.info(f"Solucion:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_real = abs(exact_y - y)
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error real = {error_real
```

```
[08-09 17:22:25][INFO] Solucion:
[08-09 17:22:25][INFO] t = 0.00, y = 0.0000, exacta = 0.0000, error real = 0.0000
[08-09 17:22:25][INFO] t = 0.50, y = 0.0000, exacta = 0.2836, error real = 0.2836
[08-09 17:22:25][INFO] t = 1.00, y = 1.1204, exacta = 3.2191, error real = 2.0987
```

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

```python
def problem_b(t, y):
    return 1 + (t - y)**2

def exact_b(t):
```

```python
    return t + 1 / (1 - t)
a = 2.0
b = 3.0
y_t0 = 1.0
h = 0.5
ys_b, ts_b = ODE_euler(a=a, b=b, f=problem_b, y_t0=y_t0, h=h)
logging.info(f"Resultados del problema (b) usando el método de Euler:")
for t, y in zip(ts_b, ys_b):
    exact_y = exact_b(t)
    error_relativo = abs((exact_y - y) / exact_y) if exact_y != 0 else float('inf')
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error relativo = {error_
```

```
[08-09 17:22:26][INFO] Resultados del problema (b) usando el método de Euler:
[08-09 17:22:26][INFO] t = 2.00, y = 1.0000, exacta = 1.0000, error relativo = 0.0000
[08-09 17:22:26][INFO] t = 2.50, y = 2.0000, exacta = 1.8333, error relativo = 0.0909
[08-09 17:22:26][INFO] t = 3.00, y = 2.6250, exacta = 2.5000, error relativo = 0.0500
```

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

```python
def problem_c(t, y):
    return 1 + y / t
def exact_c(t):
    return t * log(t) + 2 * t
a = 1.0
b = 2.0
y_t0 = 2.0
h = 0.25
ys_c, ts_c = ODE_euler(a=a, b=b, f=problem_c, y_t0=y_t0, h=h)
logging.info(f"Resultados del problema (c) usando el método de Euler:")
for t, y in zip(ts_c, ys_c):
    exact_y = exact_c(t)
    error_relativo = abs((exact_y - y) / exact_y) if exact_y != 0 else float('inf')
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error relativo = {error_
```

```
[08-09 17:22:27][INFO] Resultados del problema (c) usando el método de Euler:
[08-09 17:22:27][INFO] t = 1.00, y = 2.0000, exacta = 2.0000, error relativo = 0.0000
[08-09 17:22:27][INFO] t = 1.25, y = 2.7500, exacta = 2.7789, error relativo = 0.0104
[08-09 17:22:27][INFO] t = 1.50, y = 3.5500, exacta = 3.6082, error relativo = 0.0161
[08-09 17:22:27][INFO] t = 1.75, y = 4.3917, exacta = 4.4793, error relativo = 0.0196
[08-09 17:22:27][INFO] t = 2.00, y = 5.2690, exacta = 5.3863, error relativo = 0.0218
```

d) $y(t) = 1/2sen(2t) - 1/3cos(3t) + 4/3$

```
def problem_d(t, y):
    return cos(2 * t) + sin(3 * t)
def exact_d(t):
    return (1/5) * sin(2*t) - (1/10) * cos(2*t) - (1/9) * cos(3*t) + (1/27) * sin(3*t) + 1
a = 0.0
b = 1.0
y_t0 = 1.0
h = 0.25
ys_d, ts_d = ODE_euler(a=a, b=b, f=problem_d, y_t0=y_t0, h=h)
logging.info(f"Resultados del problema (d) usando el método de Euler:")
for t, y in zip(ts_d, ys_d):
    exact_y = exact_d(t)
    error_relativo = abs((exact_y - y) / exact_y) if exact_y != 0 else float('inf')
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error relativo = {error_
```

```
[08-09 17:22:29][INFO] Resultados del problema (d) usando el método de Euler:
[08-09 17:22:29][INFO] t = 0.00, y = 1.0000, exacta = 0.7889, error relativo = 0.2676
[08-09 17:22:29][INFO] t = 0.25, y = 1.2500, exacta = 0.9521, error relativo = 0.3129
[08-09 17:22:29][INFO] t = 0.50, y = 1.6398, exacta = 1.1433, error relativo = 0.4342
[08-09 17:22:29][INFO] t = 0.75, y = 2.0243, exacta = 1.2910, error relativo = 0.5679
[08-09 17:22:29][INFO] t = 1.00, y = 2.2365, exacta = 1.3387, error relativo = 0.6706
```

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

a)$y' = y/t - (y/t)^2, 1 \leq t \leq 2, y(1) = 1$**, con $ h = 0.1$**

```
f = lambda t, y: (y / t) - (y / t)**2
a = 1
b = 2
y_t0 = 1
h = 0.1
ys_b, ts_b = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_b, ys_b):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:30][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:30][INFO] t = 1.00, y = 1.0000
[08-09 17:22:30][INFO] t = 1.10, y = 1.0000
[08-09 17:22:30][INFO] t = 1.20, y = 1.0083
[08-09 17:22:30][INFO] t = 1.30, y = 1.0217
[08-09 17:22:30][INFO] t = 1.40, y = 1.0385
[08-09 17:22:30][INFO] t = 1.50, y = 1.0577
[08-09 17:22:30][INFO] t = 1.60, y = 1.0785
[08-09 17:22:30][INFO] t = 1.70, y = 1.1004
[08-09 17:22:30][INFO] t = 1.80, y = 1.1233
[08-09 17:22:30][INFO] t = 1.90, y = 1.1467
[08-09 17:22:30][INFO] t = 2.00, y = 1.1707
```

b)$y' = 1 + y/t + (y/t)^2, 1 \leq t \leq 3, y(1) = 0$**, con $ h = 0.2 $**

```python
f = lambda t, y: 1+y/t+(y/t)**2
a = 1
b = 3
y_t0 = 0
h = 0.2
ys_b, ts_b = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_b, ys_b):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:30][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:30][INFO] t = 1.00, y = 0.0000
[08-09 17:22:30][INFO] t = 1.20, y = 0.2000
[08-09 17:22:30][INFO] t = 1.40, y = 0.4389
[08-09 17:22:30][INFO] t = 1.60, y = 0.7212
[08-09 17:22:30][INFO] t = 1.80, y = 1.0520
[08-09 17:22:30][INFO] t = 2.00, y = 1.4373
[08-09 17:22:30][INFO] t = 2.20, y = 1.8843
[08-09 17:22:30][INFO] t = 2.40, y = 2.4023
[08-09 17:22:30][INFO] t = 2.60, y = 3.0028
[08-09 17:22:30][INFO] t = 2.80, y = 3.7006
[08-09 17:22:30][INFO] t = 3.00, y = 4.5143
```

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

```
f = lambda t, y: -(y+1)*(y+3)
a = 0
b = 2
y_t0 = -2
h = 0.2
ys_b, ts_b = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_b, ys_b):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:31][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:31][INFO] t = 0.00, y = -2.0000
[08-09 17:22:31][INFO] t = 0.20, y = -1.8000
[08-09 17:22:31][INFO] t = 0.40, y = -1.6080
[08-09 17:22:31][INFO] t = 0.60, y = -1.4387
[08-09 17:22:31][INFO] t = 0.80, y = -1.3017
[08-09 17:22:31][INFO] t = 1.00, y = -1.1993
[08-09 17:22:31][INFO] t = 1.20, y = -1.1275
[08-09 17:22:31][INFO] t = 1.40, y = -1.0797
[08-09 17:22:31][INFO] t = 1.60, y = -1.0491
[08-09 17:22:31][INFO] t = 1.80, y = -1.0300
[08-09 17:22:31][INFO] t = 2.00, y = -1.0182
```

d)$y' = -5y + 5t^2 + 2t, 0 \leq t \leq 1, y(0) = 1/3$**, con $ h = 0.1 $**

```
f = lambda t, y: -5*y+5*t**2+2*t
a = 0
b = 1
y_t0 = 1/3
h = 0.1
ys_b, ts_b = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)

logging.info(f"Aproximaciones por el metodo de Euler:")
for t, y in zip(ts_b, ys_b):
    logging.info(f"t = {t:.2f}, y = {y:.4f}")
```

```
[08-09 17:22:32][INFO] Aproximaciones por el metodo de Euler:
[08-09 17:22:32][INFO] t = 0.00, y = 0.3333
[08-09 17:22:32][INFO] t = 0.10, y = 0.1667
[08-09 17:22:32][INFO] t = 0.20, y = 0.1083
[08-09 17:22:32][INFO] t = 0.30, y = 0.1142
[08-09 17:22:32][INFO] t = 0.40, y = 0.1621
[08-09 17:22:32][INFO] t = 0.50, y = 0.2410
[08-09 17:22:32][INFO] t = 0.60, y = 0.3455
[08-09 17:22:32][INFO] t = 0.70, y = 0.4728
[08-09 17:22:32][INFO] t = 0.80, y = 0.6214
[08-09 17:22:32][INFO] t = 0.90, y = 0.7907
[08-09 17:22:32][INFO] t = 1.00, y = 0.9803
```

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.

```python
from math import log, tan, exp
```

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

```python
f = lambda t, y: (y / t) - (y / t)**2
exact_a = lambda t: t / (1 + log(t))
a = 1.0
b = 2.0
y_t0 = 1.0
h = 0.1
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
logging.info(f"Solucion:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_real = abs(exact_y - y)
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error real = {error_rea
```

```
[08-09 17:22:33][INFO] Solucion:
[08-09 17:22:33][INFO] t = 1.00, y = 1.0000, exacta = 1.0000, error real = 0.0000
[08-09 17:22:33][INFO] t = 1.10, y = 1.0000, exacta = 1.0043, error real = 0.0043
[08-09 17:22:33][INFO] t = 1.20, y = 1.0083, exacta = 1.0150, error real = 0.0067
[08-09 17:22:33][INFO] t = 1.30, y = 1.0217, exacta = 1.0298, error real = 0.0081
```

```
[08-09 17:22:33][INFO] t = 1.40, y = 1.0385, exacta = 1.0475, error real = 0.0090
[08-09 17:22:33][INFO] t = 1.50, y = 1.0577, exacta = 1.0673, error real = 0.0096
[08-09 17:22:33][INFO] t = 1.60, y = 1.0785, exacta = 1.0884, error real = 0.0100
[08-09 17:22:33][INFO] t = 1.70, y = 1.1004, exacta = 1.1107, error real = 0.0102
[08-09 17:22:33][INFO] t = 1.80, y = 1.1233, exacta = 1.1337, error real = 0.0104
[08-09 17:22:33][INFO] t = 1.90, y = 1.1467, exacta = 1.1572, error real = 0.0105
[08-09 17:22:33][INFO] t = 2.00, y = 1.1707, exacta = 1.1812, error real = 0.0106
```

b) $y(t) = t * tan(lnt)$

```python
f = lambda t, y: 1+y/t+(y/t)**2
exact_a = lambda t: t * tan(log(t))
a = 1.0
b = 3
y_t0 = 0
h = 0.2
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
logging.info(f"Solucion:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_real = abs(exact_y - y)
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error real = {error_rea
```

```
[08-09 17:22:34][INFO] Solucion:
[08-09 17:22:34][INFO] t = 1.00, y = 0.0000, exacta = 0.0000, error real = 0.0000
[08-09 17:22:34][INFO] t = 1.20, y = 0.2000, exacta = 0.2212, error real = 0.0212
[08-09 17:22:34][INFO] t = 1.40, y = 0.4389, exacta = 0.4897, error real = 0.0508
[08-09 17:22:34][INFO] t = 1.60, y = 0.7212, exacta = 0.8128, error real = 0.0915
[08-09 17:22:34][INFO] t = 1.80, y = 1.0520, exacta = 1.1994, error real = 0.1474
[08-09 17:22:34][INFO] t = 2.00, y = 1.4373, exacta = 1.6613, error real = 0.2240
[08-09 17:22:34][INFO] t = 2.20, y = 1.8843, exacta = 2.2135, error real = 0.3292
[08-09 17:22:34][INFO] t = 2.40, y = 2.4023, exacta = 2.8766, error real = 0.4743
[08-09 17:22:34][INFO] t = 2.60, y = 3.0028, exacta = 3.6785, error real = 0.6756
[08-09 17:22:34][INFO] t = 2.80, y = 3.7006, exacta = 4.6587, error real = 0.9581
[08-09 17:22:34][INFO] t = 3.00, y = 4.5143, exacta = 5.8741, error real = 1.3598
```

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

```
f = lambda t, y: -(y+1)*(y+3)
exact_a = lambda t: -3+2/(1+exp(-2*t))
a = 0
b = 2
y_t0 = -2
h = 0.2
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
logging.info(f"Solucion:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_real = abs(exact_y - y)
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error real = {error_rea
```

```
[08-09 17:22:34][INFO] Solucion:
[08-09 17:22:34][INFO] t = 0.00, y = -2.0000, exacta = -2.0000, error real = 0.0000
[08-09 17:22:34][INFO] t = 0.20, y = -1.8000, exacta = -1.8026, error real = 0.0026
[08-09 17:22:34][INFO] t = 0.40, y = -1.6080, exacta = -1.6201, error real = 0.0121
[08-09 17:22:34][INFO] t = 0.60, y = -1.4387, exacta = -1.4630, error real = 0.0242
[08-09 17:22:34][INFO] t = 0.80, y = -1.3017, exacta = -1.3360, error real = 0.0342
[08-09 17:22:34][INFO] t = 1.00, y = -1.1993, exacta = -1.2384, error real = 0.0392
[08-09 17:22:34][INFO] t = 1.20, y = -1.1275, exacta = -1.1663, error real = 0.0389
[08-09 17:22:34][INFO] t = 1.40, y = -1.0797, exacta = -1.1146, error real = 0.0349
[08-09 17:22:34][INFO] t = 1.60, y = -1.0491, exacta = -1.0783, error real = 0.0292
[08-09 17:22:34][INFO] t = 1.80, y = -1.0300, exacta = -1.0532, error real = 0.0232
[08-09 17:22:34][INFO] t = 2.00, y = -1.0182, exacta = -1.0360, error real = 0.0178
```

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

```
f = lambda t, y: -5*y+5*t**2+2*t
exact_a = lambda t: t**2+1/3*exp(-5*t)
a = 0
b = 1
y_t0 =1/3
h = 0.1
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
logging.info(f"Solucion:")
for t, y in zip(ts_a, ys_a):
    exact_y = exact_a(t)
    error_real = abs(exact_y - y)
    logging.info(f"t = {t:.2f}, y = {y:.4f}, exacta = {exact_y:.4f}, error real = {error_rea
```

```
[08-09 17:22:34][INFO] Solucion:
[08-09 17:22:34][INFO] t = 0.00, y = 0.3333, exacta = 0.3333, error real = 0.0000
[08-09 17:22:34][INFO] t = 0.10, y = 0.1667, exacta = 0.2122, error real = 0.0455
[08-09 17:22:34][INFO] t = 0.20, y = 0.1083, exacta = 0.1626, error real = 0.0543
[08-09 17:22:34][INFO] t = 0.30, y = 0.1142, exacta = 0.1644, error real = 0.0502
[08-09 17:22:34][INFO] t = 0.40, y = 0.1621, exacta = 0.2051, error real = 0.0430
[08-09 17:22:34][INFO] t = 0.50, y = 0.2410, exacta = 0.2774, error real = 0.0363
[08-09 17:22:34][INFO] t = 0.60, y = 0.3455, exacta = 0.3766, error real = 0.0311
[08-09 17:22:34][INFO] t = 0.70, y = 0.4728, exacta = 0.5001, error real = 0.0273
[08-09 17:22:34][INFO] t = 0.80, y = 0.6214, exacta = 0.6461, error real = 0.0247
[08-09 17:22:34][INFO] t = 0.90, y = 0.7907, exacta = 0.8137, error real = 0.0230
[08-09 17:22:34][INFO] t = 1.00, y = 0.9803, exacta = 1.0022, error real = 0.0219
```

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.

```python
def interpolate_linear(ts: List[float], ys: List[float], t: float) -> float:
    if t < ts[0] or t > ts[-1]:
        raise ValueError("El valor t está fuera del rango de los datos")

    for i in range(len(ts) - 1):
        if ts[i] <= t <= ts[i + 1]:
            t0, y0 = ts[i], ys[i]
            t1, y1 = ts[i + 1], ys[i + 1]
            return y0 + (t - t0) / (t1 - t0) * (y1 - y0)
    if t < ts[0]:
        return ys[0]
    else:
        return ys[-1]
def real_a(t: float) -> float:
    return t / (1 + log(t))

def real_b(t: float) -> float:
    return t * tan(log(t))

def real_c(t: float) -> float:
    return -3 + 2 / (1 + exp(-2 * t))

def real_d(t: float) -> float:
    return t ** 2 + (1 / 3) * exp(-5 * t)
```

```python
def interpolate_and_compare(ts: List[float], ys: List[float], t_values: List[float], real_fu
    results = []
    for t in t_values:
        try:
            interpolated_y = interpolate_linear(ts, ys, t)
            real_y = real_func(t)
            error = abs(real_y - interpolated_y)
            results.append((t, interpolated_y, real_y, error))
        except ValueError as e:
            logging.error(f"Error en la interpolación: {e}")
            results.append((t, None, real_func(t), None))
    return results
```

a) $y(0.25)$ **y** $y(0.93)$

```python
a, b = 1.0, 2.0
y_t0 = 1.0
h = 0.1
f = lambda t, y: (y / t) - (y / t) ** 2
ys_a, ts_a = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
t_values_a = [1.25, 1.93]
results_a = interpolate_and_compare(ts_a, ys_a, t_values_a, real_a)
logging.info("Resultados de interpolación y comparación con valores reales:")
for t, interpolated, real, error in results_a:
    if interpolated is not None:
        logging.info(f"Problema (a): t = {t:.2f}, Interpolado = {interpolated:.6f}, Real = {
    else:
        logging.info(f"Problema (a): t = {t:.2f}, Interpolado = N/A, Real = {real:.6f}, Erro
```

```
[08-09 17:26:02][INFO] Resultados de interpolación y comparación con valores reales:
[08-09 17:26:02][INFO] Problema (a): t = 1.25, Interpolado = 1.014977, Real = 1.021957, Erro
[08-09 17:26:02][INFO] Problema (a): t = 1.93, Interpolado = 1.153902, Real = 1.164390, Erro
```

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

```python
a, b = 1.0, 3.0
y_t0 = 0
h = 0.2
```

```python
f = lambda t, y: 1 + y/t+(y/t)**2
ys_b, ts_b = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
t_values_b = [1.25, 1.93]
results_b = interpolate_and_compare(ts_b, ys_b, t_values_b, real_b)

logging.info("Resultados de interpolación y comparación con valores reales:")

for t, interpolated, real, error in results_b:
    if interpolated is not None:
        logging.info(f"Problema (b): t = {t:.2f}, Interpolado = {interpolated:.6f}, Real = {r
    else:
        logging.info(f"Problema (b): t = {t:.2f}, Interpolado = N/A, Real = {real:.6f}, Erro
```

```
[08-09 17:26:03][INFO] Resultados de interpolación y comparación con valores reales:
[08-09 17:26:03][INFO] Problema (b): t = 1.25, Interpolado = 0.259722, Real = 0.283653, Erro
[08-09 17:26:03][INFO] Problema (b): t = 1.93, Interpolado = 1.302427, Real = 1.490228, Erro
```

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

```python
a, b = 0, 2.0
y_t0 = -2.0
h = 0.2
f = lambda t, y: -(y + 1)*(y + 3)
ys_c, ts_c = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
t_values_c = [1.3, 1.93]
results_c = interpolate_and_compare(ts_c, ys_c, t_values_c, real_c)
logging.info("Resultados de interpolación y comparación con valores reales:")
for t, interpolated, real, error in results_c:
    if interpolated is not None:
        logging.info(f"Problema (c): t = {t:.2f}, Interpolado = {interpolated:.6f}, Real = {r
    else:
        logging.info(f"Problema (c): t = {t:.2f}, Interpolado = N/A, Real = {real:.6f}, Erro
```

```
[08-09 17:26:04][INFO] Resultados de interpolación y comparación con valores reales:
[08-09 17:26:04][INFO] Problema (c): t = 1.30, Interpolado = -1.103618, Real = -1.138277, Er
[08-09 17:26:04][INFO] Problema (c): t = 1.93, Interpolado = -1.022283, Real = -1.041267, Er
```

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

```
a, b = 0.0, 1.0
y_t0 = 1/3
h = 0.1
f = lambda t, y: -5*y+5*t**2+2*t
ys_d, ts_d = ODE_euler(a=a, b=b, f=f, y_t0=y_t0, h=h)
t_values_d = [0.54, 0.94]
results_d = interpolate_and_compare(ts_d, ys_d, t_values_d, real_d)
logging.info("Resultados de interpolación y comparación con valores reales:")
for t, interpolated, real, error in results_d:
    if interpolated is not None:
        logging.info(f"Problema (d): t = {t:.2f}, Interpolado = {interpolated:.6f}, Real = {r
    else:
        logging.info(f"Problema (d): t = {t:.2f}, Interpolado = N/A, Real = {real:.6f}, Erro
```

```
[08-09 17:26:04][INFO] Resultados de interpolación y comparación con valores reales:
[08-09 17:26:04][INFO] Problema (d): t = 0.54, Interpolado = 0.282833, Real = 0.314002, Erro
[08-09 17:26:04][INFO] Problema (d): t = 0.94, Interpolado = 0.866552, Real = 0.886632, Erro
```

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

```
def ODE_euler_nth(
    *,
    a: float,
    b: float,
    f: Callable[[float, float], float],
    f_derivatives: List[Callable[[float, float], float]],
    y_t0: float,
    N: int
) -> tuple[list[float], list[float], float]:
    h = (b - a) / N
    t = a
    ts = [t]
    ys = [y_t0]

    for _ in range(N):
        y = ys[-1]
```

```
        T = f(t, y)
        ders = [
            h / factorial(m + 2) * mth_derivative(t, y)
            for m, mth_derivative in enumerate(f_derivatives)
        ]
        T += sum(ders)
        y += h * T
        ys.append(y)

        t += h
        ts.append(t)
    return ys, ts, h
```

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

```
f = lambda t, y: t * exp(3 * t) - 2 * y
f_p = lambda t, y: exp(3 * t) * (3 * t + 1) - 2 * (t * exp(3 * t) - 2 * y)

y_t0 = 0
a = 0
b = 1

ys_nth, ts_nth, h = ODE_euler_nth(a=a, b=b, y_t0=y_t0, f=f, N=2, f_derivatives=[f_p])
print("Problema a:")
print(f"h = {h}")
print(f"t: {ts_nth}")
print(f"y: {ys_nth}")
print()
```

```
Problema a:
h = 0.5
t: [0, 0.5, 1.0]
y: [0, 0.125, 2.0232389682729033]
```

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

```python
f = lambda t, y: 1 + (t - y) ** 2
f_p = lambda t, y: -2 * (t - y) * (1 + (t - y) ** 2)

y_t0 = 1
a = 2
b = 3

ys_nth, ts_nth, h = ODE_euler_nth(a=a, b=b, y_t0=y_t0, f=f, N=2, f_derivatives=[f_p])
print("Problema a:")
print(f"h = {h}")
print(f"t: {ts_nth}")
print(f"y: {ys_nth}")
print()
```

```
Problema a:
h = 0.5
t: [2, 2.5, 3.0]
y: [1, 1.5, 2.0]
```

c)$y' = 1 + y/t, 1 \leq t \leq 2, y(1) = 2$**, con $ h = 0.25$**

```python
f = lambda t, y: 1 + y / t
f_p = lambda t, y: -y / t**2 + 1 / t

y_t0 = 2
a = 1
b = 2

ys_nth, ts_nth, h = ODE_euler_nth(a=a, b=b, y_t0=y_t0, f=f, N=4, f_derivatives=[f_p])
print("Problema a:")
print(f"h = {h}")
print(f"t: {ts_nth}")
print(f"y: {ys_nth}")
print()
```

```
Problema a:
h = 0.25
t: [1, 1.25, 1.5, 1.75, 2.0]
y: [2, 2.71875, 3.483125, 4.286102430555555, 5.122524181547619]
```

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

```
f = lambda t, y: cos(2 * t) + sin(3 * t)
f_p = lambda t, y: -2 * sin(2 * t) + 3 * cos(3 * t)

y_t0 = 1
a = 0
b = 1

ys_nth, ts_nth, h = ODE_euler_nth(a=a, b=b, y_t0=y_t0, f=f, N=4, f_derivatives=[f_p])
print("Problema a:")
print(f"h = {h}")
print(f"t: {ts_nth}")
print(f"y: {ys_nth}")
print()
```

```
Problema a:
h = 0.25
t: [0, 0.25, 0.5, 0.75, 1.0]
y: [1, 1.34375, 1.7721870657725847, 2.110676064996487, 2.201643950842383]
```

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

```
from math import exp, cos, sin, factorial
from typing import Callable, List

# Definir las funciones y derivadas para cada problema

# a) y' = t * e^(3t) - 2y, y(0) = 0, h = 0.5
def f_a(t: float, y: float) -> float:
    return t * exp(3 * t) - 2 * y

def df_a(t: float, y: float) -> float:
    return exp(3 * t) * (3 * t + 1) - 2 * (t * exp(3 * t) - 2 * y)

def ddf_a(t: float, y: float) -> float:
    return 3 * exp(3 * t) * (3 * t + 2) - 2 * (exp(3 * t) * (3 * t + 1))

def dddf_a(t: float, y: float) -> float:
```

```python
    return 9 * exp(3 * t) * (3 * t + 3) - 6 * (3 * exp(3 * t) * (3 * t + 2))


# b) y' = 1 + (t - y)^2, y(2) = 1, h = 0.5
def f_b(t: float, y: float) -> float:
    return 1 + (t - y) ** 2

def df_b(t: float, y: float) -> float:
    return -2 * (t - y) * (1 + (t - y) ** 2)

def ddf_b(t: float, y: float) -> float:
    return 2 * (1 + (t - y) ** 2) - 2 * (t - y) * (-2 * (t - y))

def dddf_b(t: float, y: float) -> float:
    return 12 * (t - y) * (-2 * (t - y)) + 8 * (t - y) * (1 + (t - y) ** 2)


# c) y' = 1 + y / t, y(1) = 2, h = 0.25
def f_c(t: float, y: float) -> float:
    return 1 + y / t

def df_c(t: float, y: float) -> float:
    return -y / t**2 + 1 / t

def ddf_c(t: float, y: float) -> float:
    return 2 * y / t**3 - 1 / t**2

def dddf_c(t: float, y: float) -> float:
    return -6 * y / t**4 + 2 / t**3


# d) y' = cos(2t) + sin(3t), y(0) = 1, h = 0.25
def f_d(t: float, y: float) -> float:
    return cos(2 * t) + sin(3 * t)

def df_d(t: float, y: float) -> float:
    return -2 * sin(2 * t) + 3 * cos(3 * t)

def ddf_d(t: float, y: float) -> float:
    return -4 * cos(2 * t) - 9 * sin(3 * t)

def dddf_d(t: float, y: float) -> float:
```

```python
    return 8 * sin(2 * t) - 27 * cos(3 * t)


# Método de Taylor de orden 4
def ODE_euler_nth(
    *,
    a: float,
    b: float,
    f: Callable[[float, float], float],
    f_derivatives: List[Callable[[float, float], float]],
    y_t0: float,
    N: int
) -> tuple[list[float], list[float], float]:
    h = (b - a) / N
    t = a
    ts = [t]
    ys = [y_t0]

    for _ in range(N):
        y = ys[-1]
        T = f(t, y)
        ders = [
            h ** (m + 1) / factorial(m + 2) * mth_derivative(t, y)
            for m, mth_derivative in enumerate(f_derivatives)
        ]
        T += sum(ders)
        y += h * T
        ys.append(y)

        t += h
        ts.append(t)
    return ys, ts, h
problems = [
    {"a": 0, "b": 1, "f": f_a, "f_derivatives": [df_a, ddf_a, dddf_a], "y_t0": 0, "N": 2},
    {"a": 2, "b": 3, "f": f_b, "f_derivatives": [df_b, ddf_b, dddf_b], "y_t0": 1, "N": 2},
    {"a": 1, "b": 2, "f": f_c, "f_derivatives": [df_c, ddf_c, dddf_c], "y_t0": 2, "N": 4},
    {"a": 0, "b": 1, "f": f_d, "f_derivatives": [df_d, ddf_d, dddf_d], "y_t0": 1, "N": 4},
]
results = [ODE_euler_nth(**problem) for problem in problems]

for i, (ys, ts, h) in enumerate(results):
    print(f"Problema {chr(97 + i)}:")
```

```python
    print(f"h = {h}")
    print(f"t: {ts}")
    print(f"y: {ys}")
    print()
```

```
Problema a:
h = 0.5
t: [0, 0.5, 1.0]
y: [0, 0.18489583333333331, 2.3041147886173525]

Problema b:
h = 0.5
t: [2, 2.5, 3.0]
y: [1, 1.6458333333333333, 2.2593370602454668]

Problema c:
h = 0.25
t: [1, 1.25, 1.5, 1.75, 2.0]
y: [2, 2.7249348958333335, 3.4950996961805556, 4.303565100742335, 5.145247904523946]

Problema d:
h = 0.25
t: [0, 0.25, 0.5, 0.75, 1.0]
y: [1, 1.3289388020833333, 1.7296672968020275, 2.039934166759473, 2.1159884664152244]
```

Link del repositorio GITHUB: https://github.com/armando-2002/Metodos_Numericos.git