Tabla de Contenidos

ESCUELA POLITÉCNICA NACIONAL	Tarea No.
Metodos Numericos – Computación	**6**
NOMBRE: Ivonne Carolina Ayala	

[Tarea 06] Unidad 03-A | Serie de Taylor y Polinomios de Lagrange

Conjunto de Ejercicios

Determine el orden de la mejor aproximación para las siguientes funciones, usando la Serie de Taylor y el Polinomio de Lagrange:

1. $\frac{1}{25x^2+1}$, $x_0 = 0$
2. $\arctan(x)$, $x_0 = 1$

Instrucciones:

- Escriba las fórmulas de los diferentes polinomios.
- Grafique las diferentes aproximaciones.

1. $\frac{1}{25x^2+1}$, $x_0 = 0$

Taylor

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

x = sp.symbols('x')
f = 1 / (25 * (x**2) + 1)
```

```
derivada= sp.diff(f, x)
print(derivada)
```

```
-50*x/(25*x**2 + 1)**2
```

La derivada para x0 es 0

```
f = -50*x/(25*x**2 + 1)**2
derivada= sp.diff(f, x)
print(derivada)
```

```
5000*x**2/(25*x**2 + 1)**3 - 50/(25*x**2 + 1)**2
```

La derivada para x0 es 25

```
f = 5000*x**2/(25*x**2 + 1)**3 - 50/(25*x**2 + 1)**2
derivada= sp.diff(f, x)
print(derivada)
```

```
-750000*x**3/(25*x**2 + 1)**4 + 15000*x/(25*x**2 + 1)**3
```

La derivada para x0 es 0

```
f = -750000*x**3/(25*x**2 + 1)**4 + 15000*x/(25*x**2 + 1)**3
derivada= sp.diff(f,x )
print(derivada)
```

```
150000000*x**4/(25*x**2 + 1)**5 - 4500000*x**2/(25*x**2 + 1)**4 + 15000/(25*x**2 + 1)**3
```

La derivada para x0 es 625

```
x = sp.symbols('x')
f = (150000000 * x**4 / (25 * x**2 + 1)**5) - (4500000 * x**2 / (25 * x**2 + 1)**4) + (15000
derivada = sp.diff(f, x)
print(derivada)
```

```
-37500000000*x**5/(25*x**2 + 1)**6 + 1500000000*x**3/(25*x**2 + 1)**5 - 11250000*x/(25*x**2 +
```

La derivada para x0 es 0

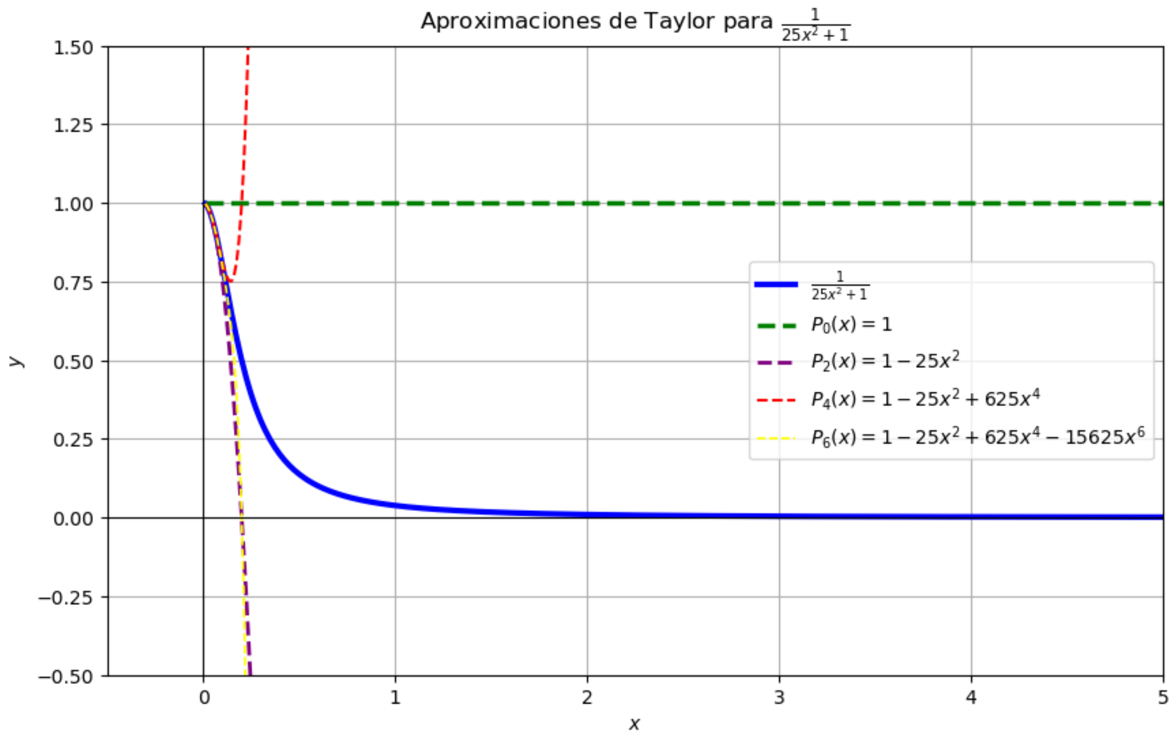
```
x = sp.symbols('x')
f = (-37500000000*x**5/(25*x**2 + 1)**6 + 1500000000*x**3/(25*x**2 + 1)**5 - 11250000*x/(25*x
derivada = sp.diff(f, x)
print(derivada)
```

```
11250000000000*x**6/(25*x**2 + 1)**7 - 562500000000*x**4/(25*x**2 + 1)**6 + 6750000000*x**2/
```

La derivada para x0 es -15625

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

x = np.linspace(0.01, 5, 400)

fOriginal = 1 / (25 * (x**2) + 1)

P0 = np.ones_like(x)
P1 = np.ones_like(x)
P2 = 1 - 25 * (x**2)
P3 = 1 - 25 * (x**2)
P4 = 1 - 25 * (x**2) + 625 * (x**4)
P6 = 1 - 25 * (x**2) + 625 * (x**4) - 15625 * (x**6)

plt.figure(figsize=(10, 6))

plt.plot(x, fOriginal, label=r'$\frac{1}{25x^2 + 1}$', color='blue', linewidth=3)
plt.plot(x, P0, '--', label=r'$P_0(x) = 1$', color='green', linewidth=2.5)
# plt.plot(x, P1 * np.ones_like(x), '--', label=r'$P_1(x) = 1$', color='orange', linewidth=2
plt.plot(x, P2, '--', label=r'$P_2(x) = 1 - 25x^2$', color='purple', linewidth=2)
#plt.plot(x, P3, '--', label=r'$P_3(x) = 1 - 25x^2$', color='pink', linewidth=1.8)
plt.plot(x, P4, '--', label=r'$P_4(x) = 1 - 25x^2 + 625x^4$', color='red', linewidth=1.5)
plt.plot(x, P6, '--', label=r'$P_6(x) = 1 - 25x^2 + 625x^4 - 15625x^6$', color='yellow', line

plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.title(r'Aproximaciones de Taylor para $\frac{1}{25x^2 + 1}$')
plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)
plt.grid(True)
plt.legend()
plt.ylim(-0.5, 1.5)
```

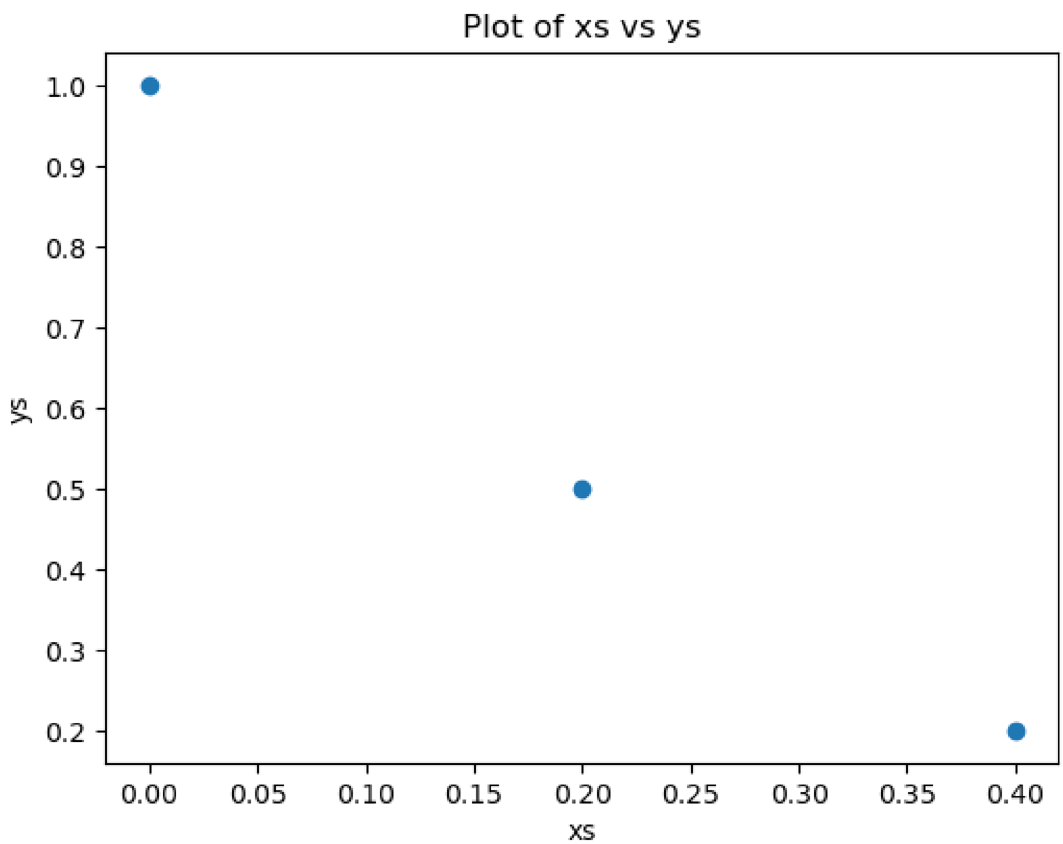
```
plt.xlim(-0.5, 5)
plt.show()
```



Lagrange

Sacamos los puntos de la funcion original:

```
import matplotlib.pyplot as plt

xs = [0, 0.2, 0.4, 0.6]
ys = [1, 0.5, 0.2, 0.1]

plt.scatter(xs, ys)
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Plot of xs vs ys")
plt.show()
```



Polinomios base

(L_0(x)):

$$L_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)}$$

Sustituyendo los valores:

$$L_0(x) = \frac{(x - 0.2)(x - 0.4)(x - 0.6)}{(0 - 0.2)(0 - 0.4)(0 - 0.6)} = \frac{(x - 0.2)(x - 0.4)(x - 0.6)}{-0.048}$$

(L_1(x)):

$$L_1(x) = \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)}$$

Sustituyendo los valores:

$$L_1(x) = \frac{(x-0)(x-0.4)(x-0.6)}{(0.2-0)(0.2-0.4)(0.2-0.6)} = \frac{(x)(x-0.4)(x-0.6)}{0.016}$$

(L_2(x)):

$$L_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}$$

Sustituyendo los valores:

$$L_2(x) = \frac{(x-0)(x-0.2)(x-0.6)}{(0.4-0)(0.4-0.2)(0.4-0.6)} = \frac{(x)(x-0.2)(x-0.6)}{-0.016}$$

(L_3(x)):

$$L_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

Sustituyendo los valores:

$$L_3(x) = \frac{(x-0)(x-0.2)(x-0.4)}{(0.6-0)(0.6-0.2)(0.6-0.4)} = \frac{(x)(x-0.2)(x-0.4)}{0.048}$$

Expresión general

$$P(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + y_3L_3(x)$$

Sustituyendo los valores:

$$P(x) = 1 \cdot L_0(x) + 0.5 \cdot L_1(x) + 0.2 \cdot L_2(x) + 0.1 \cdot L_3(x)$$

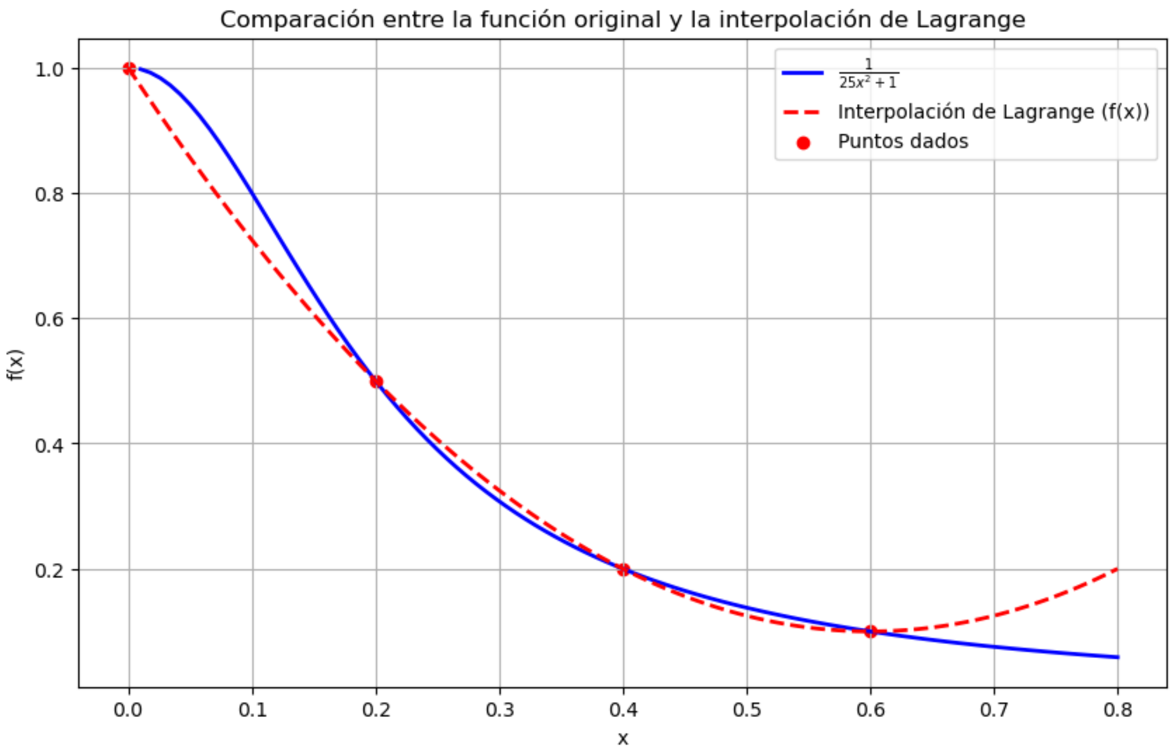
Expansión del polinomio

Sustituyendo los polinomios base:

$$P(x) = 1 \cdot \frac{(x-0.2)(x-0.4)(x-0.6)}{-0.048} + 0.5 \cdot \frac{(x)(x-0.4)(x-0.6)}{0.016} + 0.2 \cdot \frac{(x)(x-0.2)(x-0.6)}{-0.016} + 0.1 \cdot \frac{(x)(x-0.2)(x-}{0.048}$$

```python
def f(x: float) -> float:
    term1 = ((x - 0.2) * (x - 0.4) * (x - 0.6)) / (-0.048)
    term2 = (0.5 * (x) * (x - 0.4) * (x - 0.6)) / (0.016)
    term3 = (0.2 * (x) * (x - 0.2) * (x - 0.6)) / (-0.016)
    term4 = (0.1 * (x) * (x - 0.2) * (x - 0.4)) / (0.048)
    return term1 + term2 + term3 + term4
```

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

x_values = np.linspace(0, 0.8, 100)
y_values = [f(x) for x in x_values]

x = np.linspace(0.01, 0.8, 100)
fOriginal = 1 / (25 * (x**2) + 1)

plt.figure(figsize=(10, 6))

plt.plot(x, fOriginal, label=r'$\frac{1}{25x^2 + 1}$', color='blue', linewidth=2)

plt.plot(x_values, y_values, 'r--', label="Interpolación de Lagrange (f(x))", linewidth=2)

plt.scatter(xs, ys, color="red", label="Puntos dados")

plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Comparación entre la función original y la interpolación de Lagrange")
plt.legend()
plt.grid(True)
plt.show()
```



2. $\arctan(x),\ x_0 = 1$

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

x = sp.symbols('x')
f = sp.atan(x)
x0 = 1

ordenes = [1, 2, 3, 4]
polinomios_taylor = [sp.series(f, x, x0, n+1).removeO() for n in ordenes]

print("Polinomios de Taylor:")
for n, p in zip(ordenes, polinomios_taylor):
    print(f"Orden {n}: {p}")

puntos_x = [0.5, 1, 1.5]
puntos_y = [f.evalf(subs={x: xi}) for xi in puntos_x]
```

```python
def lagrange_polynomial(x_points, y_points):
    """Construye el polinomio de Lagrange"""
    n = len(x_points)
    P = 0
    for i in range(n):
        Li = 1
        for j in range(n):
            if i != j:
                Li *= (x - x_points[j]) / (x_points[i] - x_points[j])
        P += Li * y_points[i]
    return sp.simplify(P)

polinomio_lagrange = lagrange_polynomial(puntos_x, puntos_y)
print("\nPolinomio de Lagrange:")
print(polinomio_lagrange)

taylor_funciones = [sp.lambdify(x, p, 'numpy') for p in polinomios_taylor]
lagrange_func = sp.lambdify(x, polinomio_lagrange, 'numpy')

x_vals = np.linspace(0, 2, 500)
f_func = sp.lambdify(x, f, 'numpy')

plt.figure(figsize=(12, 8))
plt.plot(x_vals, f_func(x_vals), label='Función original $arctan(x)$', linewidth=2.5)

for n, t_func in zip(ordenes, taylor_funciones):
    plt.plot(x_vals, t_func(x_vals), label=f'Polinomio de Taylor (Orden {n})')

plt.plot(x_vals, lagrange_func(x_vals), '--', label='Polinomio de Lagrange', linewidth=2)

plt.title('Aproximaciones de $arctan(x)$ con Taylor y Lagrange', fontsize=16)
plt.xlabel('$x$', fontsize=14)
plt.ylabel('$f(x)$', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

```
Polinomios de Taylor:
Orden 1: x/2 - 1/2 + pi/4
Orden 2: x/2 - (x - 1)**2/4 - 1/2 + pi/4
Orden 3: x/2 + (x - 1)**3/12 - (x - 1)**2/4 - 1/2 + pi/4
Orden 4: x/2 + (x - 1)**3/12 - (x - 1)**2/4 - 1/2 + pi/4
```

```
Polinomio de Lagrange:
-0.248709989093523*x**2 + 1.01656609243357*x + 0.0175420600574023
```