Métodos Numéricos - DEBER 06 - Serie de Taylor y Polinomio de Lagrange

Alicia Pereira

Tabla de Contenidos

1 Conjunto de Ejercicios

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

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

1.1.1 Serie de Taylor

Fórmula:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

Calcular la mejor aproximación para esta función utilizando Serie de Taylor implica calcular la deriva de la función n veces hasta encontrar la mejor aproximación en la fórmula. Para ello, se plante el uso de la librería *SymPy* en Python, la cual tiene una función predeterminada para calcular la serie de Taylor de cualquier función dada.

Parámetros que utiliza:

- Función: La función que deseas expandir.
- Símbolo: La variable respecto a la cual expandir.
- Punto: El punto alrededor del cual se expande.
- Orden: El grado hasta el cual deseas calcular la expansión.

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

```python
x = sp.symbols('x')

f = 1/(25*x**2+1)
x0 = 0
n = 30

print("Serie de Taylor:")
print(sp.series(f, x, x0, n))
```

```
Serie de Taylor:
1 - 25*x**2 + 625*x**4 - 15625*x**6 + 390625*x**8 - 9765625*x**10 + 244140625*x**12 - 610351
```

En este caso no se incluyen los polinomios de grado impar, dado que sus coeficientes son igual a 0.

$ P_0(x) = 1 $

$ P_2(x) = 1 - 25x^2 $

$ P_4(x) = 1 - 25x^2 + 625x^4 $

$ P_6(x) = 1 - 25x^2 + 625x^4 - 15625x^6 $

$ P_8(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 $

$ P_{10}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} $

$ P_{12}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} $

$ P_{14}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} $

$ P_{16}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} $

$ P_{18}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} $

$ P_{20}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} + 95367431640625x^{20} $

$ P_{22}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} + 95367431640625x^{20} - 2384185791015625x^{22} $

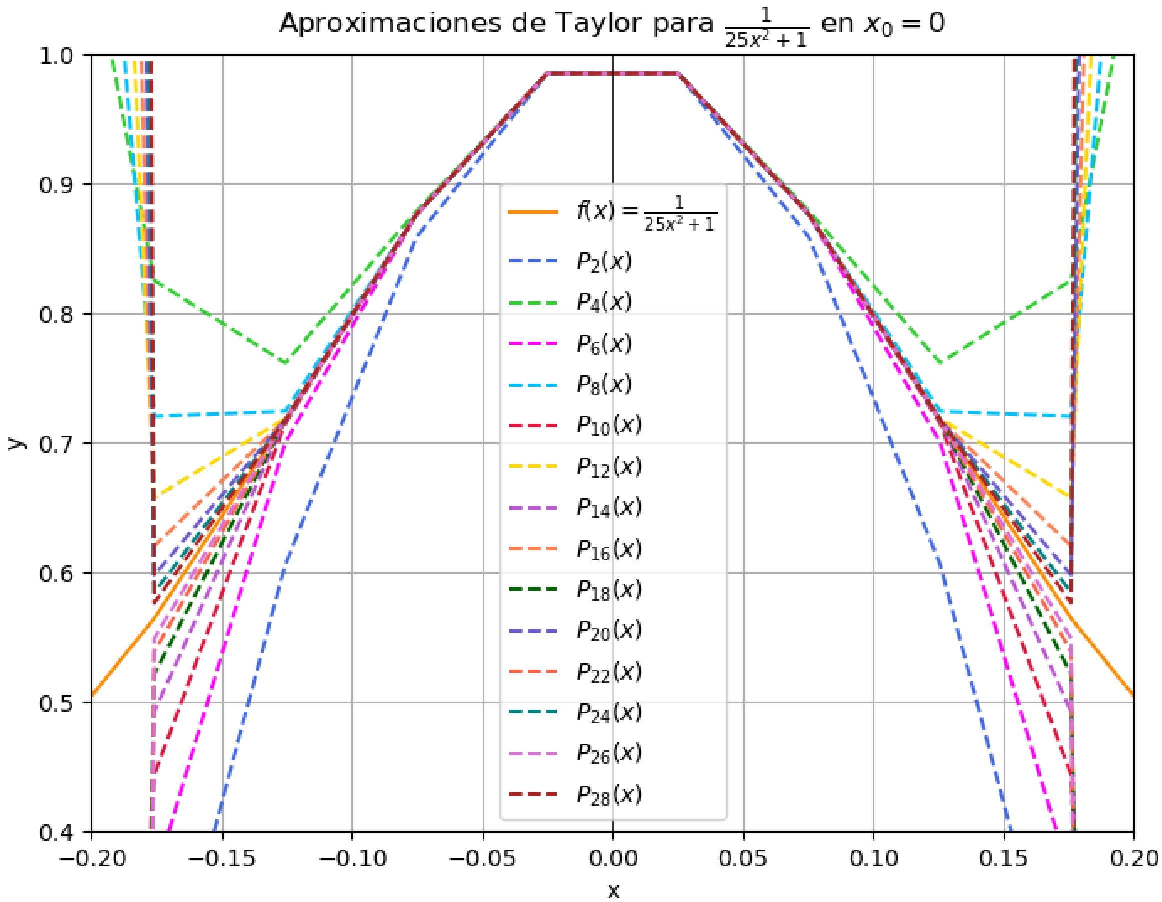
$ P_{24}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} + 95367431640625x^{20} - 2384185791015625x^{22} + 59604644775390625x^{24} $

$ P_{26}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} + 95367431640625x^{20} - 2384185791015625x^{22} + 59604644775390625x^{24} - 1490116119384765625x^{26} $

$ P_{28}(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8 - 9765625x^{10} + 244140625x^{12} - 6103515625x^{14} + 152587890625x^{16} - 3814697265625x^{18} + 95367431640625x^{20} - 2384185791015625x^{22} + 59604644775390625x^{24} - 1490116119384765625x^{26} + 37252902984619140625x^{28} $

```
x = np.linspace(-5, 5, 200)

y =  1/(25*x**2+1)
y1 = 1 - (25)*x**2
y2 = y1 + (625)*(x**4)
y3 = y2 - (15625)*(x**6)
y4 = y3 + (390625)*(x**8)
y5 = y4 - 9765625*x**10
y6 = y5 + 244140625*x**12
y7 = y6 - 6103515625*x**14
y8 = y7 + 152587890625*x**16
y9 = y8 - 3814697265625*x**18
y10 = y9 + 95367431640625*x**20
y11 = y10 - 2384185791015625*x**22
y12 = y11 + 59604644775390625*x**24
```

```python
y13 = y12 - 14901161193847656250*x**26
y14 = y13 + 372529029846191406250*x**28

plt.figure(figsize=(8, 6))

plt.plot(x, y, label=r'$f(x) = \frac{1}{25x^{2}+1}$', color='darkorange')
plt.plot(x, y1, label=r'$P_2(x)$', color='royalblue', linestyle='--')
plt.plot(x, y2, label=r'$P_4(x)$', color='limegreen', linestyle='--')
plt.plot(x, y3, label=r'$P_6(x)$', color='magenta', linestyle='--')
plt.plot(x, y4, label=r'$P_8(x)$', color='deepskyblue', linestyle='--')
plt.plot(x, y5, label=r'$P_{10}(x)$', color='crimson', linestyle='--')
plt.plot(x, y6, label=r'$P_{12}(x)$', color='gold', linestyle='--')
plt.plot(x, y7, label=r'$P_{14}(x)$', color='mediumorchid', linestyle='--')
plt.plot(x, y8, label=r'$P_{16}(x)$', color='coral', linestyle='--')
plt.plot(x, y9, label=r'$P_{18}(x)$', color='darkgreen', linestyle='--')
plt.plot(x, y10, label=r'$P_{20}(x)$', color='slateblue', linestyle='--')
plt.plot(x, y11, label=r'$P_{22}(x)$', color='tomato', linestyle='--')
plt.plot(x, y12, label=r'$P_{24}(x)$', color='teal', linestyle='--')
plt.plot(x, y13, label=r'$P_{26}(x)$', color='orchid', linestyle='--')
plt.plot(x, y14, label=r'$P_{28}(x)$', color='firebrick', linestyle='--')

plt.xlabel('x')
plt.ylabel('y')
plt.title(r'Aproximaciones de Taylor para $\frac{1}{25x^{2}+1}$ en $x_0 = 0$')

plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.ylim([0.4, 1])
plt.xlim([-0.2, 0.2])

plt.grid(True)
plt.legend()
plt.show()
```



- Las aproximaciones con pocos términos no siguen la curva de ( ) de manera precisa, sobre todo al alejarse de $x = 0$.
- Dada la naturaleza cuadrática de la función en el denominador,estos polinomios presentan divergencia para valores mayores de $|x|$.
- A medida que aumenta el número de términos, las aproximaciones se ajustan más a la curva real.
- En el intervalo observado de $-0.2 \leq x \leq 0.2$, $P_{14}(x)$ y $P_{20}(x)$ poseen una buena similitud.
- Una aproximación mayor al grado 28 mostrado, lograrán una mejor fidelidad a la curva real en todo el intervalo.

1.1.2 Polinomio de Lagrange

Fórmula:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

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

def f(x):
    return 1 / (25 * x**2 + 1)

xs = [-0.25,-0.099, 0,0.099,0.25]
xs = np.array(xs)
ys = f(xs)


def lagrange(x) -> float:
    n = len(xs)
    L = 0

    for i in range(n):
        Li = 1
        for j in range(n):
            if i != j:
                Li *= (x - xs[j]) / (xs[i] - xs[j])

        L += Li * ys[i]

    return sp.expand(L)
```

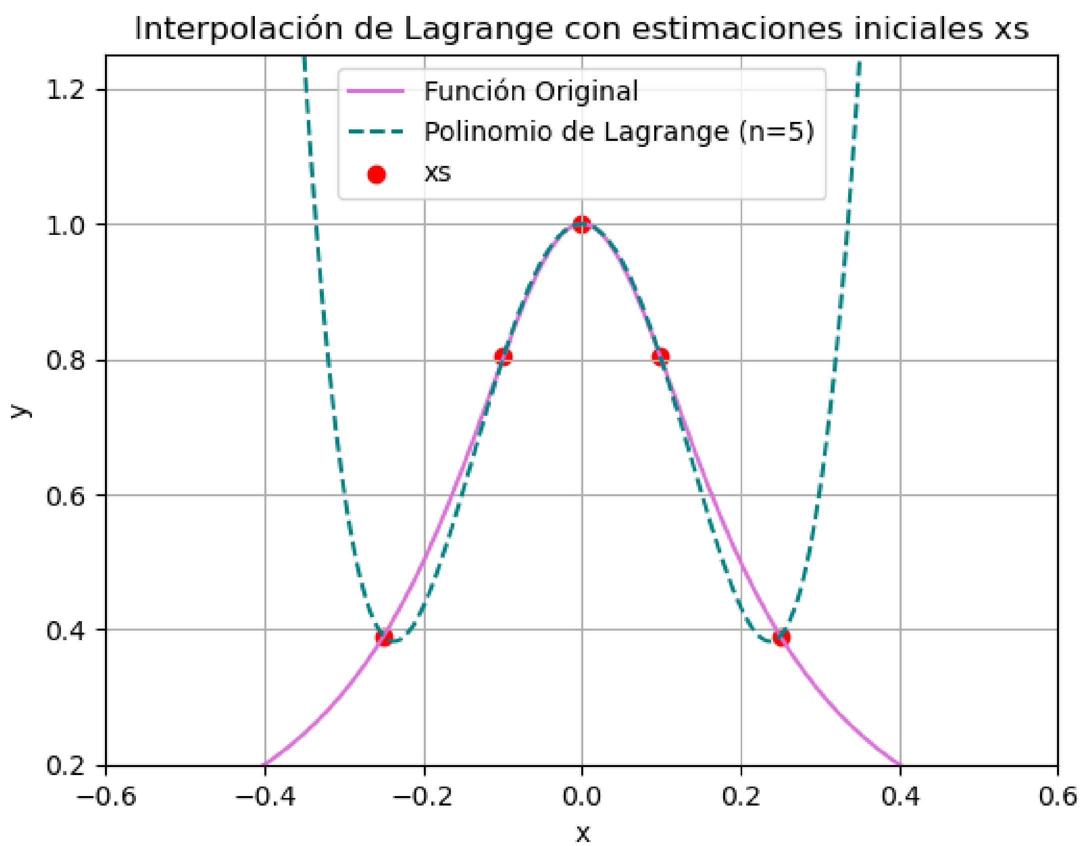
El polinomio de Lagrange que interpola los puntos dados es:

$$L(x) = 195.9016 \cdot x^4 + 1.7764 \times 10^{-15} \cdot x^3 - 22.0000 \cdot x^2 - 2.4980 \times 10^{-16} \cdot x + 1.0$$

Este polinomio pasa exactamente por los puntos:

$$(-0.25, f(-0.25)), (-0.099, f(-0.099)), (0, f(0)), (0.099, f(0.099)), (0.25, f(0.25))$$

```
x_vals = np.linspace(-1, 1, 400)
y_vals = f(x_vals)
lagrange_poly = [lagrange(x) for x in x_vals]

plt.plot(x_vals, y_vals, label="Función Original", color='orchid')
plt.plot(x_vals, lagrange_poly, label=f"Polinomio de Lagrange (n={len(xs)})", color='teal', l
plt.scatter(xs, ys, color='red', label="xs")
plt.legend()
plt.title("Interpolación de Lagrange con estimaciones iniciales xs")
plt.xlabel("x")
plt.ylabel("y")
plt.ylim([0.2, 1.25])
plt.xlim([-0.6, 0.6])
plt.grid(True)
plt.show()
```



1.2 b. $arctan(x), x_0 = 1$

1.2.1 Serie de Taylor

Fórmula:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

Se plantea el uso de la librería *SymPy* para calcular la serie de Taylor.

```
x = sp.symbols('x')
y = sp.atan(x)
x0 = 1
n = 6
print("Serie de Taylor:")
print(sp.series(y, x, x0, n))
```

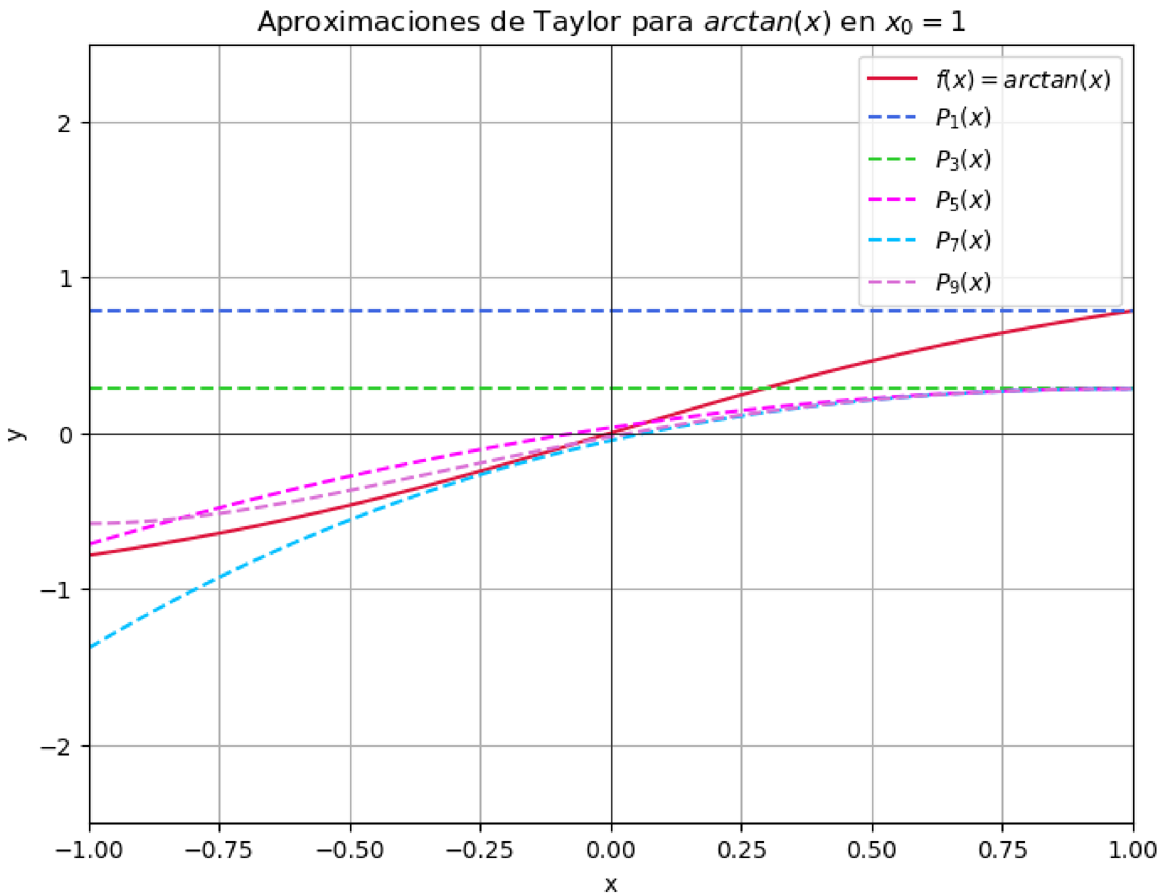
```
Serie de Taylor:
pi/4 - 1/2 - (x - 1)**2/4 + (x - 1)**3/12 - (x - 1)**5/40 + x/2 + O((x - 1)**6, (x, 1))
```

Entonces, los polinomios correspondientes de la serie de Taylor son:

1. $P_1(x) = \frac{\pi}{4}$
2. $P_2(x) = \frac{\pi}{4} - \frac{1}{2}$
3. $P_3(x) = \frac{\pi}{4} - \frac{1}{2} - \frac{(x-1)^2}{4}$
4. $P_4(x) = \frac{\pi}{4} - \frac{1}{2} - \frac{(x-1)^2}{4} + \frac{(x-1)^3}{12}$
5. $P_5(x) = \frac{\pi}{4} - \frac{1}{2} - \frac{(x-1)^2}{4} + \frac{(x-1)^3}{12} - \frac{(x-1)^5}{40}$

```
x = np.linspace(-1, 1, 400)
y = np.arctan(x)
y1 = np.pi/4 + 0 * x
y2 = y1 - 1/2
y3 = y2 - (x - 1)**2/4
y4 = y3 + (x - 1)**3/12
y5 = y4 - (x - 1)**5/40

plt.figure(figsize=(8, 6))

plt.plot(x, y, label=r'$f(x) = arctan(x)$', color='crimson')
```

```python
plt.plot(x, y1, label=r'$P_1(x)$', color='royalblue', linestyle='--')
plt.plot(x, y2, label=r'$P_2(x)$', color='limegreen', linestyle='--')
plt.plot(x, y3, label=r'$P_3(x)$', color='magenta', linestyle='--')
plt.plot(x, y4, label=r'$P_4(x)$', color='deepskyblue', linestyle='--')
plt.plot(x, y5, label=r'$P_5(x)$', color='orchid', linestyle='--')


plt.xlabel('x')
plt.ylabel('y')
plt.title(r'Aproximaciones de Taylor para $arctan(x)$ en $x_0 = 1$')

plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.ylim([-2.5, 2.5])
plt.xlim([-1, 1])

plt.grid(True)
plt.legend()
plt.show()
```



1.2.2 Polinomio de Lagrange

Fórmula:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

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

def f(x):
    return np.arctan(x)

xs = [-0.25,-0.099, 0,0.099,0.25]
```

```
xs = np.array(xs)
ys = f(xs)


def lagrange(x) -> float:
    n = len(xs)
    L = 0

    for i in range(n):
        Li = 1
        for j in range(n):
            if i != j:
                Li *= (x - xs[j]) / (xs[i] - xs[j])

        L += Li * ys[i]

    return sp.expand(L)
```

El polinomio de Lagrange que interpola los puntos dados es:

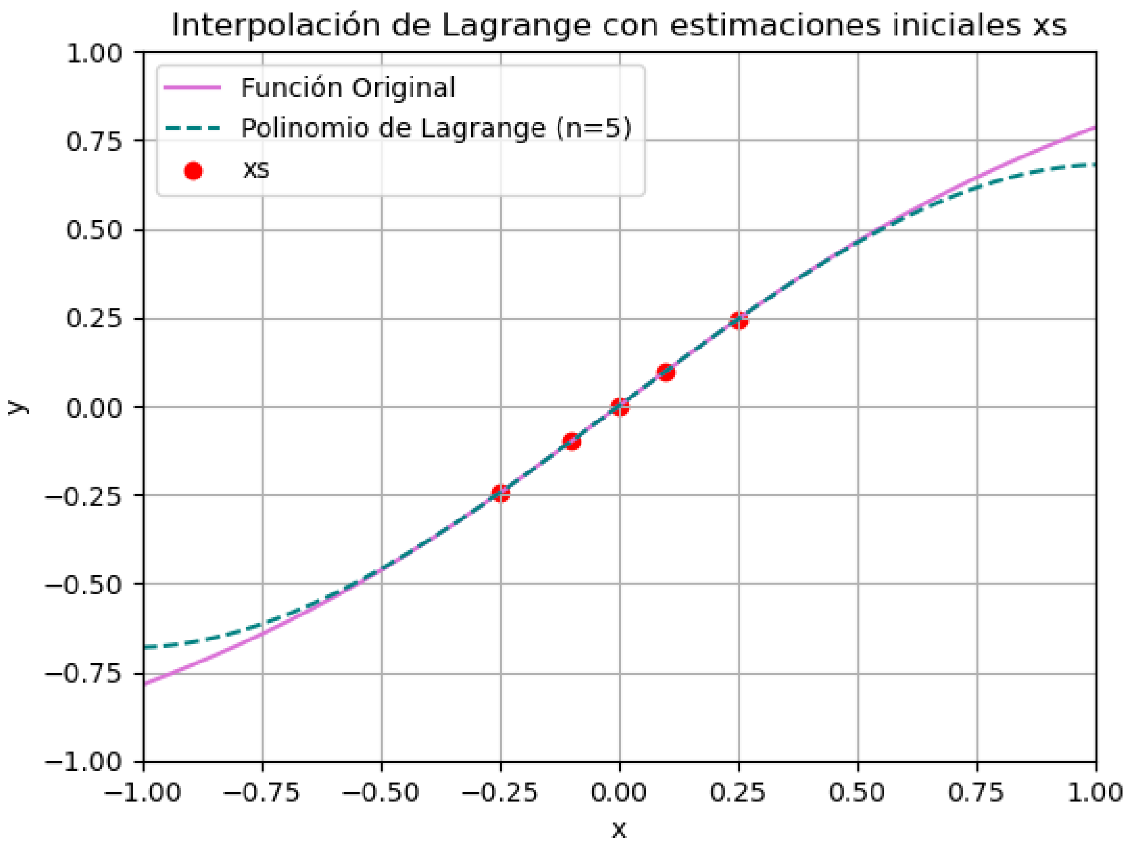
$$L(x) = -0.3195 \cdot x^3 + 0.9999 \cdot x$$

Este polinomio pasa exactamente por los puntos:

$$(-0.25, \arctan(-0.25)), (-0.099, \arctan(-0.099)), (0, \arctan(0)), (0.099, \arctan(0.099)), (0.25, \arctan(0.25))$$

```
x_vals = np.linspace(-1, 1, 400)
y_vals = f(x_vals)
lagrange_poly = [lagrange(x) for x in x_vals]

plt.plot(x_vals, y_vals, label="Función Original", color='orchid')
plt.plot(x_vals, lagrange_poly, label=f"Polinomio de Lagrange (n={len(xs)})", color='teal', l
plt.scatter(xs, ys, color='red', label="xs")
plt.legend()
plt.title("Interpolación de Lagrange con estimaciones iniciales xs")
plt.xlabel("x")
plt.ylabel("y")
plt.ylim([-1, 1])
plt.xlim([-1, 1])
```

```
plt.grid(True)
plt.show()
```



GitHub: git@github.com: dayapt04

GitHub Métodos Númericos - Repositorio