TAREA N° 6

Nombre:

Joel Stalin Tinitana Carrion

Fecha:

02/06/2025

Tema:

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**:

$$f(x)=\frac{1}{125x^2+1}, x_0=0$$
$$f(x)=\arctan(x), x_0=1$$

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

Polinomio de Taylor

Fórmulas

$$f(x)=P_n(x)+R_n(x)$$

Donde el polinomio de Taylor de orden (n) es:

$$P_n(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

O, de forma resumida:

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

Condiciones del Polinomio de Taylor

$$f(x)\in C^n[a,b] \text{(la función es continua con derivadas hasta orden } n\text{)}$$

$$f^{(n+1)} \text{ existe en } [a,b]$$

$$x_0 \in [a, b]$$

Interpretación

El polinomio de Taylor aproxima una función (f(x)) por una suma de potencias centradas en un punto (x_0). A mayor orden (n), mejor será la aproximación local. El término (R_n(x)) es el **error de truncamiento**.

Polinomio de Lagrange

Fórmula general

$$P(x) = \sum_{k=0}^{n} f(x_k) L_k(x)$$

Donde cada base (L_k(x)) se define como:

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

Interpretación

El polinomio de Lagrange es una herramienta para **interpolar** una función a partir de (n + 1) puntos conocidos. Se construye sin derivadas, y garantiza que el polinomio pase por todos los puntos dados. Es especialmente útil cuando se conoce solo el valor de la función y no su derivada.

Resolución con polinomio de Taylor

Ejercicio 1

Función:

$$f(x) = \frac{1}{125x^2 + 1}, x_0 = 0$$

Derivadas:

$$f(x) = (125x^2 + 1)^{-1}$$

$$f'(x) = \frac{-250x}{(125x^2 + 1)^2}$$

$$f''(x)=\frac{250(375x^2-1)}{(125x^2+1)^3}$$

$$f^{(3)}(x)=\frac{750000x(125x^2-3)}{(125x^2+1)^4}$$

$$f^{(4)}(x)=1562500\text{(calculada con software)}$$

Evaluación en (x_0 = 0):

$$f(0)=1, f'(0)=0, f''(0)=-250, f^{(3)}(0)=0, f^{(4)}(0)=1562500$$

Polinomio de Taylor de orden 4 centrado en (x = 0):

$$T_4(x)=f(0)+\frac{f''(0)}{2!}x^2+\frac{f^{(4)}(0)}{4!}x^4$$

$$T_4(x)=1-125x^2+\frac{1562500}{24}x^4=1-125x^2+65104.17x^4$$

Ejercicio 2

Función:

$$f(x)=\arctan(x), x_0=1$$

Derivadas:

$$f'(x)=\frac{1}{1+x^2}$$

$$f''(x)=\frac{-2x}{(1+x^2)^2}$$

$$f^{(3)}(x)=\frac{-2(1+x^2)^2+8x^2(1+x^2)}{(1+x^2)^4}$$

Evaluación en (x_0 = 1):

$$f(1)=\frac{\pi}{4}, f'(1)=\frac{1}{2}, f''(1)=-\frac{1}{2}, f^{(3)}(1)=\frac{1}{2}$$

Polinomio de Taylor de orden 3 centrado en (x = 1):

$$T_3(x)=f(1)+f'(1)(x-1)+\frac{f''(1)}{2}(x-1)^2+\frac{f^{(3)}(1)}{6}(x-1)^3$$

$$T_3(x)=\frac{\pi}{4}+\frac{1}{2}(x-1)-\frac{1}{4}(x-1)^2+\frac{1}{12}(x-1)^3$$

Resolución con el Polinomio de Lagrange

Ejercicio 1

Función:

$$f(x)=\frac{1}{125x^2+1}$$

Puntos de interpolación:

Tomamos tres puntos equidistantes centrados en (x_0 = 0), por ejemplo:

$$x_0=-0.1, x_1=0, x_2=0.1$$

Calculamos los valores de la función:

$$f(x_0)=\frac{1}{125(-0.1)^2+1}=\frac{1}{1.25+1}=\frac{1}{2.25}\approx 0.4444$$

$$f(x_1)=f(0)=1$$

$$f(x_2)=\frac{1}{125(0.1)^2+1}=\frac{1}{2.25}\approx 0.4444$$

Fórmulas del Polinomio de Lagrange:

$$L(x)=f(x_0)\ell_0(x)+f(x_1)\ell_1(x)+f(x_2)\ell_2(x)$$

Donde:

$$\ell_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \ell_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \ell_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

Reemplazando los valores numéricos se puede construir el polinomio completo.

Ejercicio 2

Función:

$$f(x)=\arctan(x)$$

Puntos de interpolación:

Elegimos:

$$x_0 = 0.8, x_1 = 1.0, x_2 = 1.2$$

Calculamos:

$$f(x_0) = \arctan(0.8) \approx 0.6747$$

$$f(x_1) = \arctan(1.0) = \frac{\pi}{4} \approx 0.7854$$

$$f(x_2) = \arctan(1.2) \approx 0.8761$$

Polinomio de Lagrange:

$$L(x) = f(x_0)\ell_0(x) + f(x_1)\ell_1(x) + f(x_2)\ell_2(x)$$

Donde:

$$\ell_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, \ell_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, \ell_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

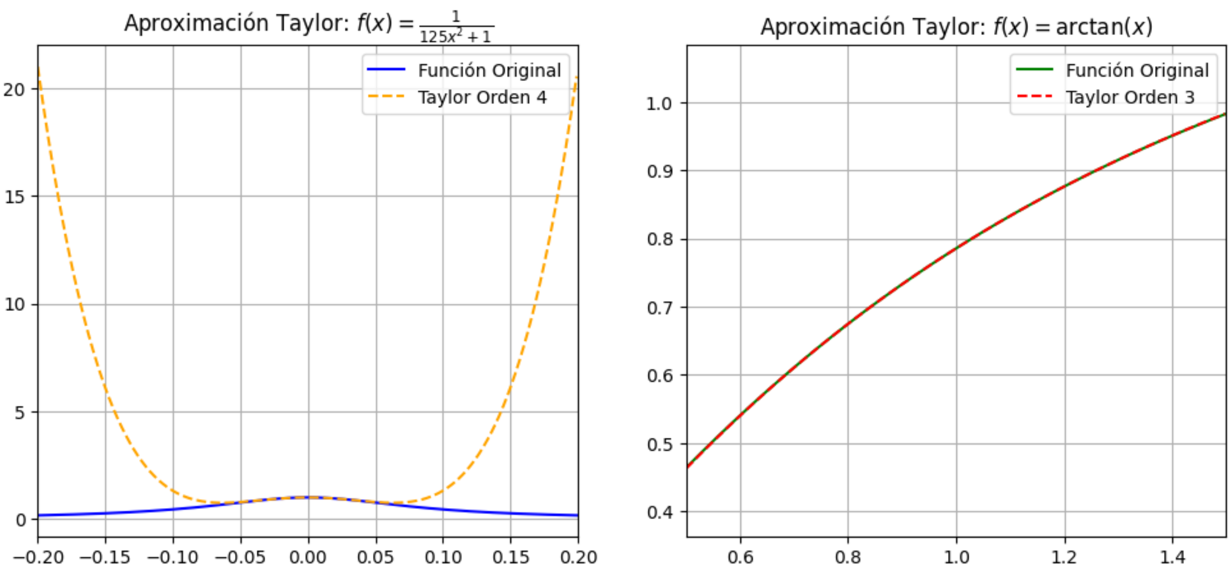
Sustituyendo los valores numéricos también se puede construir la expresión completa del polinomio interpolante.

Codigo de taylor

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

# Variables simbólicas
x = sp.Symbol('x')

# FUNCIONES ORIGINALES
f1_expr = 1 / (125 * x**2 + 1)
f2_expr = sp.atan(x)

# TAYLOR (orden 4 para f1 en x=0, orden 3 para f2 en x=1)
taylor_f1 = sp.series(f1_expr, x, 0, 5).removeO()
taylor_f2 = sp.series(f2_expr, x, 1, 4).removeO()

# FUNCIONES LAMBDA PARA EVALUACIÓN
f1 = sp.lambdify(x, f1_expr, 'numpy')
t1 = sp.lambdify(x, taylor_f1, 'numpy')

f2 = sp.lambdify(x, f2_expr, 'numpy')
t2 = sp.lambdify(x, taylor_f2, 'numpy')

# Rango de valores
```

```python
x1_vals = np.linspace(-0.2, 0.2, 400)
x2_vals = np.linspace(0.5, 1.5, 400)

y1_vals = f1(x1_vals)
yt1_vals = t1(x1_vals)

y2_vals = f2(x2_vals)
yt2_vals = t2(x2_vals)

# Animación
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
ax1.set_title("Aproximación Taylor: $f(x) = \\frac{1}{125x^2 + 1}$")
ax2.set_title("Aproximación Taylor: $f(x) = \\arctan(x)$")
ax1.grid(True)
ax2.grid(True)

line_f1, = ax1.plot([], [], label='Función Original', color='blue')
line_t1, = ax1.plot([], [], '--', label='Taylor Orden 4',
color='orange')

line_f2, = ax2.plot([], [], label='Función Original', color='green')
line_t2, = ax2.plot([], [], '--', label='Taylor Orden 3', color='red')

ax1.set_xlim(x1_vals.min(), x1_vals.max())
ax1.set_ylim(min(y1_vals.min(), yt1_vals.min()) - 1,
max(y1_vals.max(), yt1_vals.max()) + 1)

ax2.set_xlim(x2_vals.min(), x2_vals.max())
ax2.set_ylim(min(y2_vals.min(), yt2_vals.min()) - 0.1,
max(y2_vals.max(), yt2_vals.max()) + 0.1)

ax1.legend()
ax2.legend()

def animate(i):
    line_f1.set_data(x1_vals[:i], y1_vals[:i])
    line_t1.set_data(x1_vals[:i], yt1_vals[:i])
    line_f2.set_data(x2_vals[:i], y2_vals[:i])
    line_t2.set_data(x2_vals[:i], yt2_vals[:i])
    return line_f1, line_t1, line_f2, line_t2

ani = animation.FuncAnimation(fig, animate, frames=400, interval=20,
blit=True)

from matplotlib import rc
rc('animation', html='jshtml')
ani
```

```
<matplotlib.animation.FuncAnimation at 0x26e50da5710>
```

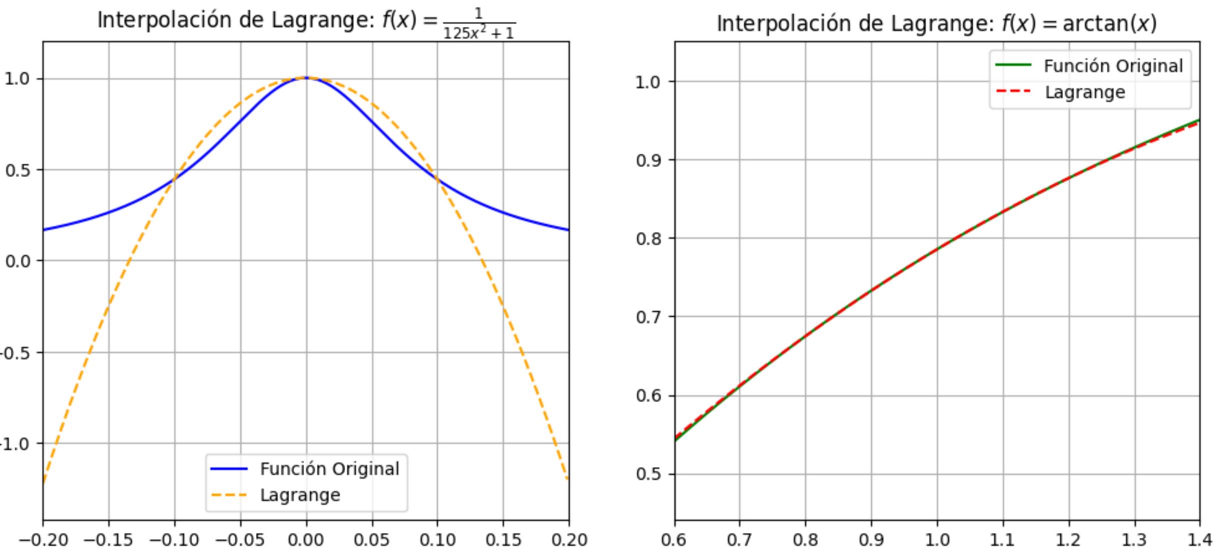


Código de Lagrange

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

# Puntos de interpolación para f(x) = 1 / (125x^2 + 1)
x_vals1 = np.array([-0.1, 0, 0.1])
y_vals1 = 1 / (125 * x_vals1**2 + 1)

# Puntos de interpolación para f(x) = arctan(x)
x_vals2 = np.array([0.8, 1.0, 1.2])
y_vals2 = np.arctan(x_vals2)

# Función para calcular el polinomio de Lagrange
def lagrange_poly(x_vals, y_vals, x_eval):
    total = 0
    n = len(x_vals)
    for i in range(n):
        xi, yi = x_vals[i], y_vals[i]
        li = 1
        for j in range(n):
            if i != j:
                li *= (x_eval - x_vals[j]) / (xi - x_vals[j])
        total += yi * li
    return total

# Rango para animación
x_plot1 = np.linspace(-0.2, 0.2, 400)
x_plot2 = np.linspace(0.6, 1.4, 400)

original_f1 = 1 / (125 * x_plot1**2 + 1)
```

```python
lagrange_f1 = [lagrange_poly(x_vals1, y_vals1, x) for x in x_plot1]

original_f2 = np.arctan(x_plot2)
lagrange_f2 = [lagrange_poly(x_vals2, y_vals2, x) for x in x_plot2]

# Crear figura y ejes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
ax1.set_title("Interpolación de Lagrange: $f(x) = \\frac{1}{125x^2 + 
1}$")
ax2.set_title("Interpolación de Lagrange: $f(x) = \\arctan(x)$")
ax1.grid(True)
ax2.grid(True)

# Inicializar líneas vacías
line_f1, = ax1.plot([], [], label='Función Original', color='blue')
line_l1, = ax1.plot([], [], '--', label='Lagrange', color='orange')

line_f2, = ax2.plot([], [], label='Función Original', color='green')
line_l2, = ax2.plot([], [], '--', label='Lagrange', color='red')

ax1.set_xlim(x_plot1.min(), x_plot1.max())
ax1.set_ylim(min(min(original_f1), min(lagrange_f1)) - 0.2, 
max(max(original_f1), max(lagrange_f1)) + 0.2)
ax2.set_xlim(x_plot2.min(), x_plot2.max())
ax2.set_ylim(min(min(original_f2), min(lagrange_f2)) - 0.1, 
max(max(original_f2), max(lagrange_f2)) + 0.1)

ax1.legend()
ax2.legend()

# Función de animación
def animate_lagrange(i):
    line_f1.set_data(x_plot1[:i], original_f1[:i])
    line_l1.set_data(x_plot1[:i], lagrange_f1[:i])
    line_f2.set_data(x_plot2[:i], original_f2[:i])
    line_l2.set_data(x_plot2[:i], lagrange_f2[:i])
    return line_f1, line_l1, line_f2, line_l2

# Crear animación
ani_lagrange = animation.FuncAnimation(fig, animate_lagrange, 
frames=400, interval=20, blit=True)

from matplotlib import rc
rc('animation', html='jshtml')
ani_lagrange
```

```
<matplotlib.animation.FuncAnimation at 0x26e5b476e10>
```



Interpolación de Lagrange: $f(x) = \frac{1}{125x^2+1}$
Función Original
Lagrange
1.0
0.5
0.0
−0.5
−1.0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Interpolación de Lagrange: $f(x) = \arctan(x)$
Función Original
Lagrange
1.0
0.9
0.8
0.7
0.6
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4