Escuela Politécnica Nacional

Nombre: Francisco Ulloa

Grupo: GR1CC

Asignatura: Métodos Numéricos

Repositorio: https://github.com/Fu5CHAR/Metodos_numericos_2025B_Ulloa-Francisco/tree/main

Tarea: Tarea 6

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

1. $\frac{1}{5x^2+1}, x_0 = 0$ **Derivadas**

$f'(x) = \frac{-50x}{25x^2+1}^2$

$f''(x) = 50\frac{75x^2-1}{(25x^2+1)^3}$

$f'''(x) = -15000x\frac{25x^2-1}{(125x^2+1)^4}$

$f^{(4)}(x) = 15000\frac{3125x^4-250x^2+1}{(25x^2+1)^3}$

Polinomio de Taylor

$P_0(0) = f(0) = \frac{1}{25(0^2)+1}$

$P_0(0) = 1$

$P_1(0) = P_0(0) + f'(0)(x-0)$

$P_1(0) = 1 - \frac{50(0)}{(25(0)^2+1)^2}(x)$

$P_1(0) = 1$

$P_2(0) = P_1(0) + \frac{f''(0)}{2!}(x-0)^2$

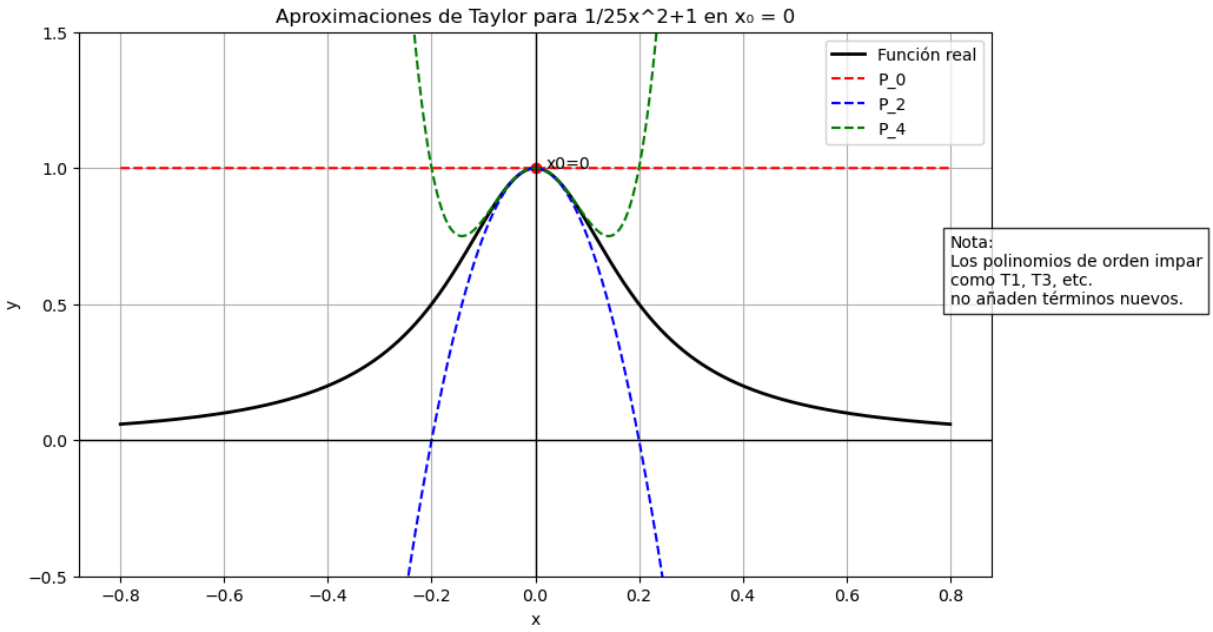
$P_2(0) = 1 + \frac{1}{2}\frac{50(75(0)^2-1)}{(25(0)^2+1)^3}(x)^2$

$P_2(0) = 1 - 25x^2$

$$P_3(0) = P_2(0) + \frac{f'''(0)}{3!}(x-0)^3$$
$$P_3(0) = P_2(0) + \frac{1}{6}(-15000(0)\frac{25(0)^2-1}{(25(0)^2+1)^4}) * (x)$$
$$p_3(0) = 1 - 25x^2$$

$$P_4(0) = P_3(0) + \frac{f^(4)}{4!}(x-0)^4$$
$$P_4(0) = p_3(0) + \frac{15000x^4}{24}$$
$$P_4(0) = 1 - 25x^2 + 625x^4$$

In [34]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator

x = np.linspace(-0.8, 0.8, 5000)
y_real = (1 / ((25*(x)**2)+1))

#Polinomios de Taylor en x=0

P_0= np.ones_like(x)
P_2= 1 - (25*x**2)
P_4= 1 - (25*x**2) + (625*x**4)

#gráficar
plt.figure(figsize=(10, 6))
plt.plot(x, y_real, label='Función real', color='black', linewidth=2)
plt.plot(x, P_0,'--r',label='P_0')
plt.plot(x, P_2, '--b',label='P_2')
plt.plot(x, P_4,'--g' ,label='P_4')
plt.scatter(0, 1, color='red')
plt.text(0+0.02, 1, 'x0=0', fontsize=10)

ax = plt.gca()
plt.axhline(0, color='black',linewidth=1)
plt.axvline(0, color='black',linewidth=1)
ax.yaxis.set_major_locator(MultipleLocator(0.5))
plt.ylim(-0.5, 1.5)

plt.legend()
plt.title('Aproximaciones de Taylor para 1/25x^2+1 en x₀ = 0')
plt.text(
    0.80, 0.50,
    "Nota:\nLos polinomios de orden impar\ncomo T1, T3, etc.\nno añaden términos nu
    fontsize=10,
    bbox=dict(facecolor='white', alpha=0.8)
)
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```



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

Derivadas

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

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

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

$$f^{(4)}(x) = \frac{24x(1-x^2)}{(1+x^2)^4}.$$

Polinomios de Taylor centrados en $x_0 = 1$

Orden 0: $P_0(x) = f(1) = \frac{\pi}{4}$.

Orden 1: $P_1(x) = f(1) + f'(1)(x-1) = \frac{\pi}{4} + \frac{1}{2}(x-1)$.

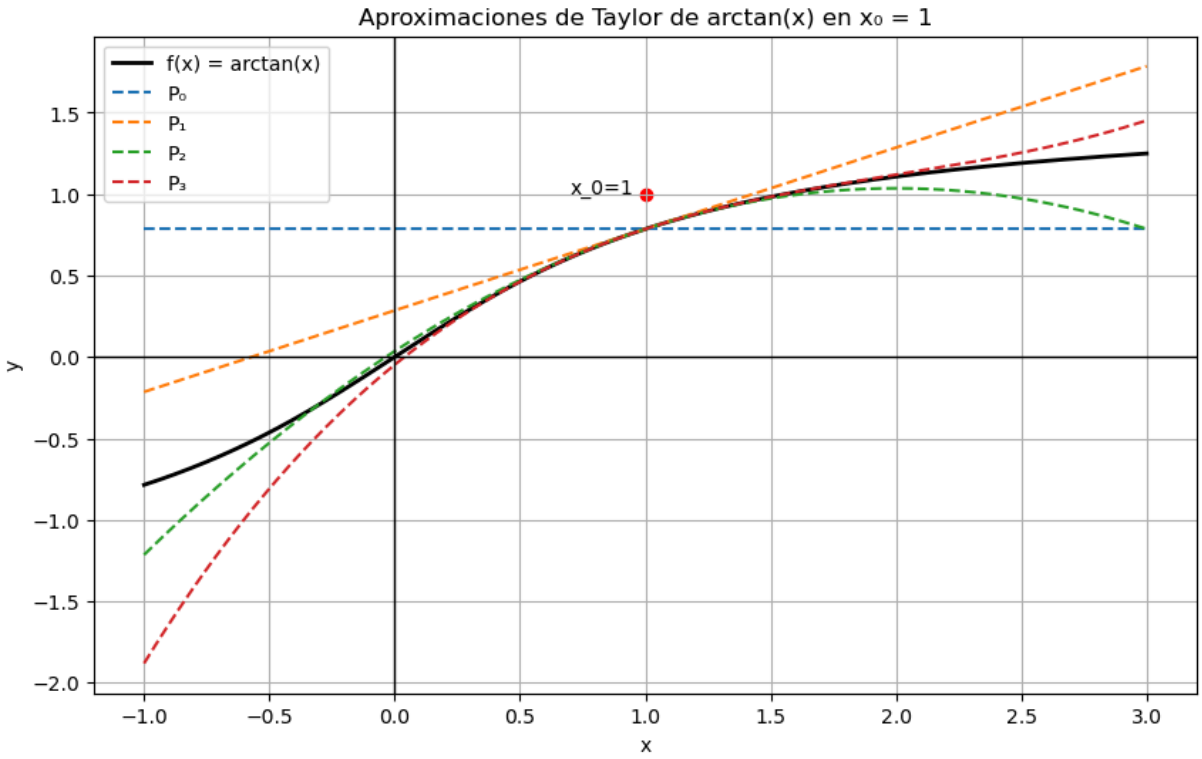
Orden 2: $P_2(x) = P_1(x) + \frac{f''(1)}{2!}(x-1)^2 = \frac{\pi}{4} + \frac{1}{2}(x-1) - \frac{1}{4}(x-1)^2$.

Orden 3: $P_3(x) = P_2(x) + \frac{f^{(3)}(1)}{3!}(x-1)^3 = \frac{\pi}{4} + \frac{1}{2}(x-1) - \frac{1}{4}(x-1)^2 + \frac{1}{12}(x-1)^3$.

Orden 4: $P_4(x) = P_3(x) + \frac{f^{(4)}(1)}{4!}(x-1)^4 = P_3(x) \quad (\text{ya que } f^{(4)}(1) = 0)$.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
from math import pi
# ---- Función y polinomios ----
```

```
def f(x):
    return np.arctan(x)
# Polinomios de Taylor centrados en x0 = 1
x0 = 1

f0 = pi/4
P0 = lambda x: f0
P1 = lambda x: f0 + 1/2*(x - x0)
P2 = lambda x: f0 + 1/2*(x - x0) - 1/4*(x - x0)**2
P3 = lambda x: f0 + 1/2*(x - x0) - 1/4*(x - x0)**2 + 1/12*(x - x0)**3
P4 = lambda x: P3(x)  # f^{(4)}(1) = 0

# ---- Rango de graficación ----
x = np.linspace(-1, 3, 1000)

# ---- Gráfica ----
plt.figure(figsize=(10,6))

plt.plot(x, f(x),'black', label="f(x) = arctan(x)", linewidth=2)
plt.plot(x, np.ones_like(x)*P0(0), '--', label="P₀")
plt.plot(x, P1(x), '--', label="P₁")
plt.plot(x, P2(x), '--', label="P₂")
plt.plot(x, P3(x), '--', label="P₃")
plt.scatter(1, 1, color='red')
plt.text(0+0.7, 1, 'x0=1', fontsize=10)

# ---- Ejes ----
plt.axhline(0, color="black", linewidth=1)
plt.axvline(0, color="black", linewidth=1)



plt.title("Aproximaciones de Taylor de arctan(x) en x₀ = 1")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



Polinomios de Langrange

$P(x) = \sum_{i=0}^{n} Y_i * L_i(x) \backslash\ P(x) = \sum_{i=0}^{n} Y_i(\prod_{j=0, j\neq i}^{n} \frac{X - x_j}{x - X_j})$

1. $f(x) = \frac{1}{25x^2+1}$

Puntos escogidos para esta función:
$P_0 = (-0.6; 0.1), P_1 = (-0.4; 0.2), P_2 = (0; 1) P_3 = (0.4; 0.2) P_4 = (0.6; 0.1)$

Calculo de L_i :

$$L_0 = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)}$$
$$L_0 = \frac{125x(x^2-0.16)(x-0.6)}{18}$$

$$l_1 = \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)}$$
$$L_1 = \frac{-125x(x^2-0.36)(x-0.4)}{8}$$

$$L_2 = \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)}$$
$$L_2 = \frac{625(x^2-0.36)(x^2-0.16)}{36}$$

$$L_3 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)}$$
$$L_3 = \frac{-125x(x^2-0.36)(x+0.4)}{8}$$

$$L_4 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)}$$

$$L_4 = \frac{125x(x^2-0.16)(x-0.6)}{18}$$

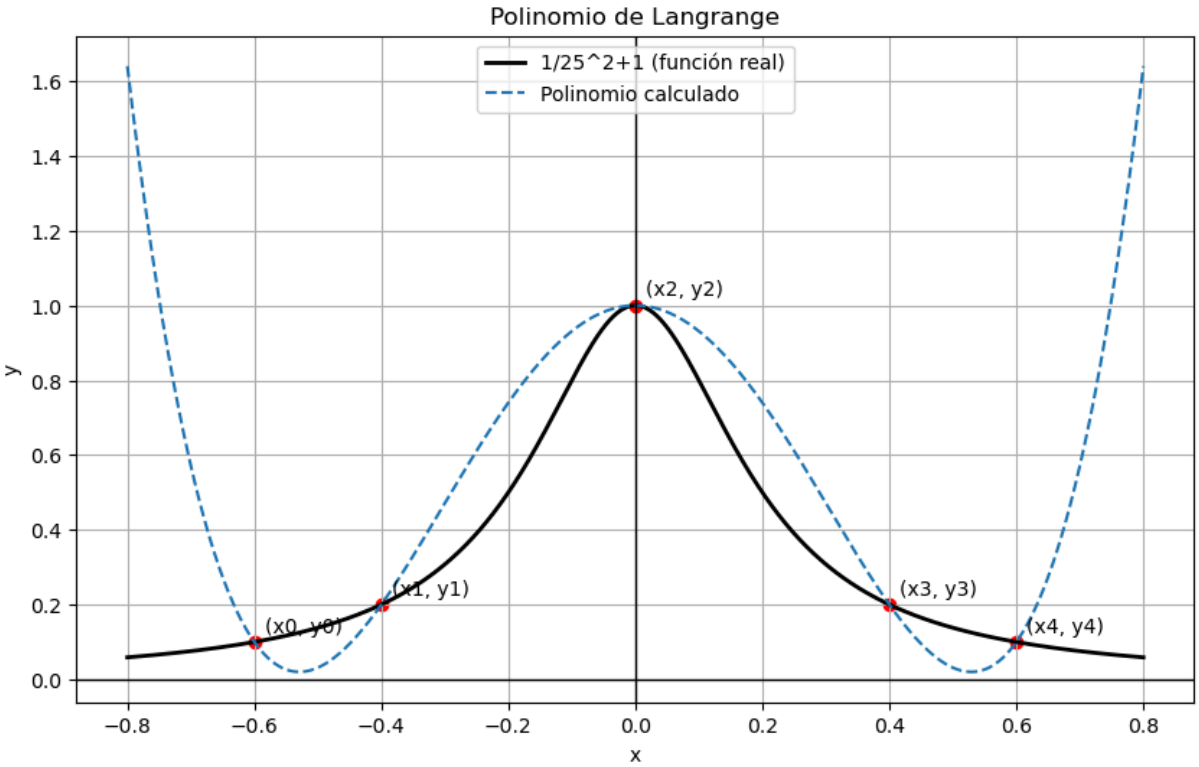
Calculo de $P(x)$:

$$P(x) = y_0 * L_0 + y_1 * L_1 + y_2 * L_2 + y_3 * L_3 + y_4 * L_4$$

$$P(x) = (0.1) * \frac{125x(x^2-0.16)(x-0.6)}{18} + (0.2) * \frac{-125x(x^2-0.36)(x-0.4)}{8} + (1) * \frac{625(x^2-0.36)(x^2-0.16)}{36} +$$

$$P(x) = 12.5x^2 - 7x + 1$$

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
from math import pi

x = np.linspace(-0.8, 0.8, 5000)
y_real=(1 / ((25*(x)**2)+1))

def P(x):
 return 12.5*x**4 - 7*x**2 + 1


# ---- Gráfica ----
plt.figure(figsize=(10,6))

plt.plot(x, y_real,'black', label="1/25^2+1 (función real)", linewidth=2)
plt.plot(x, P(x), '--', label="Polinomio calculado")
plt.scatter(-0.6, 0.1, color='red')
plt.scatter(-0.4, 0.2, color='red')
plt.scatter(0, 1, color='red')
plt.scatter(0.4, 0.2, color='red')
plt.scatter(0.6, 0.1, color='red')
plt.annotate('(x0, y0)', xy=(-0.6, 0.1), xytext=(5, 5), textcoords='offset points',
plt.annotate('(x1, y1)', xy=(-0.4, 0.2), xytext=(5, 5), textcoords='offset points',
plt.annotate('(x2, y2)', xy=(0, 1), xytext=(5, 5), textcoords='offset points', font
plt.annotate('(x3, y3)', xy=(0.4, 0.2), xytext=(5, 5), textcoords='offset points',
plt.annotate('(x4, y4)', xy=(0.6, 0.1), xytext=(5, 5), textcoords='offset points',
# ---- Ejes ----
plt.axhline(0, color="black", linewidth=1)
plt.axvline(0, color="black", linewidth=1)



plt.title("Polinomio de Langrange")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



1. $f(x) = \frac{1}{25x^2+1}$

Puntos escogidos para esta función:

$P_0 = (-1.5; -1), P_1 = (-0.2; -0.2), P_2 = (0; 0) P_3 = (0.2; 0.2) P_4 = (1.5; 1)$

Calculo de L_i :

$$L_0 = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)}$$

$$L_0 = \frac{x(x^2-0.4)(x-1.5)}{9.945}$$

$$l_1 = \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)}$$

$$L_1 = \frac{x(x^2-2.25)(x-0.2)}{-0.1768}$$

$$L_2 = \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)}$$

$$L_2 = \frac{(x^2-2.25)(x^2-0.4)}{0.09}$$

$$L_3 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)}$$

$$L_3 = \frac{(x^2-2.25)(x+0.2)}{-0.1768}$$

$$L_4 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)}$$

$$L_4 = \frac{125x(x^2-0.4)(x+1.5)}{9.945}$$

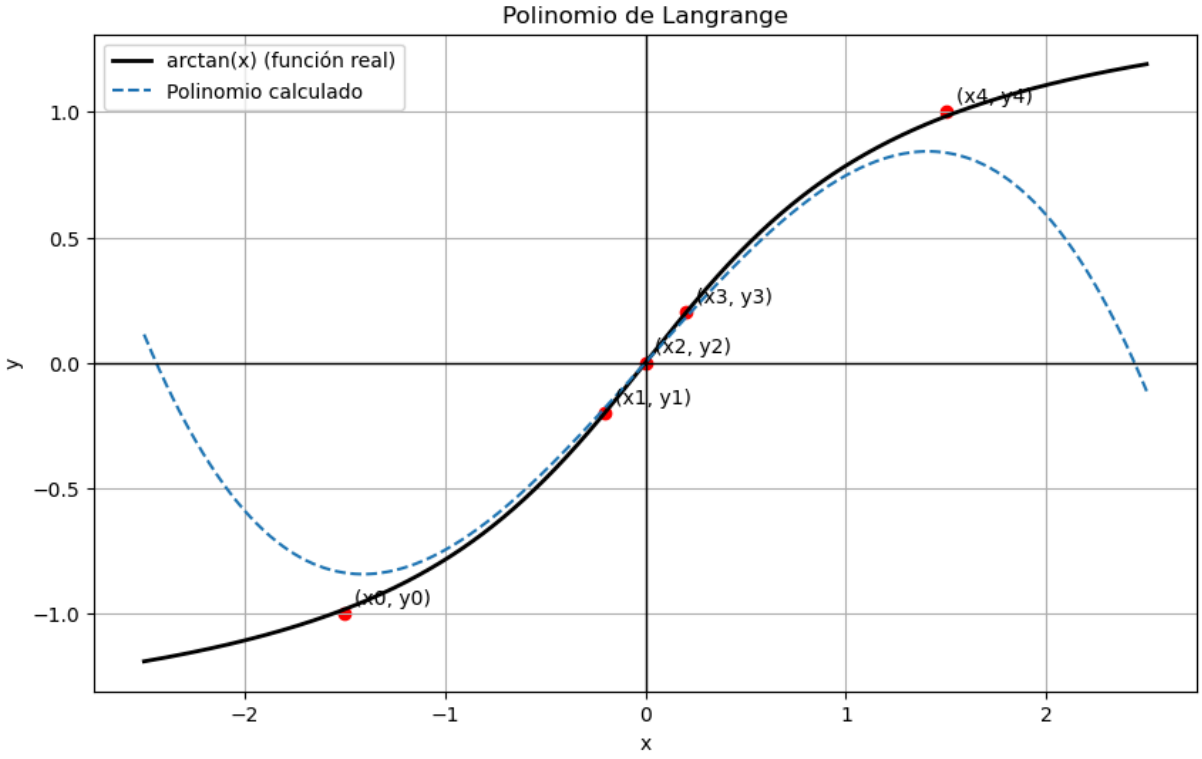
Calculo de $P(x)$:

$$P(x) = y_0 * L_0 + y_1 * L_1 + y_2 * L_2 + y_3 * L_3 + y_4 * L_4$$

$$P(x) = (-1)\frac{x(x^2-0.4)(x-1.5)}{9.945} + \frac{x(x^2-2.25)(x-0.2)}{-0.1768} + (0) * \frac{(x^2-2.25)(x^2-0.4)}{0.09} + (0.2) * \frac{(x^2-2.25)(x+0}{-0.1768}$$

$$P(x) = -0.15083x^3 + 0.89744x$$

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
from math import pi

x = np.linspace(-2.5, 2.5, 9000)
y_real=np.arctan(x)

def P(x):
 return -0.15083*x**3 + 0.89744*x


# ---- Gráfica ----
plt.figure(figsize=(10,6))

plt.plot(x, y_real,'black', label="arctan(x) (función real)", linewidth=2)
plt.plot(x, P(x), '--', label="Polinomio calculado")
plt.scatter(-1.5, -1, color='red')
plt.scatter(-0.2, -0.2, color='red')
plt.scatter(0, 0, color='red')
plt.scatter(0.2, 0.2, color='red')
plt.scatter(1.5, 1, color='red')
plt.annotate('(x0, y0)', xy=(-1.5, -1), xytext=(5, 5), textcoords='offset points',
plt.annotate('(x1, y1)', xy=(-0.2, -0.2), xytext=(5, 5), textcoords='offset points'
plt.annotate('(x2, y2)', xy=(0, 0), xytext=(5, 5), textcoords='offset points', font
plt.annotate('(x3, y3)', xy=(0.2, 0.2), xytext=(5, 5), textcoords='offset points',
plt.annotate('(x4, y4)', xy=(1.5, 1), xytext=(5, 5), textcoords='offset points', fo
# ---- Ejes ----
plt.axhline(0, color="black", linewidth=1)
plt.axvline(0, color="black", linewidth=1)



plt.title("Polinomio de Langrange")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```


Polinomio de Langrange
arctan(x) (función real)
Polinomio calculado
(x4, y4)
(x3, y3)
(x2, y2)
(x1, y1)
(x0, y0)
1.0
0.5
0.0
−0.5
−1.0
y
−2
−1
0
1
2
x