Escuela Politécnica Nacional

Nombre: Francisco Ulloa
Fecha: Quito, 12 de noviembre de 2025
Tema: Polinomios de Taylor y Langrange
Repositorio:
https://github.com/Fu5CHAR/Metodos_numericos_2025B_Ulloa-Francisco/tree/main

1.b. $f(x) = \ln(x),\ x_0 = 1$

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

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

$$f'''(x) = \frac{2}{x^3}$$

$$f^{(4)}(x) = -\frac{6}{x^4}$$

Polinomios de Taylor centrados en $x_0 = 1$:

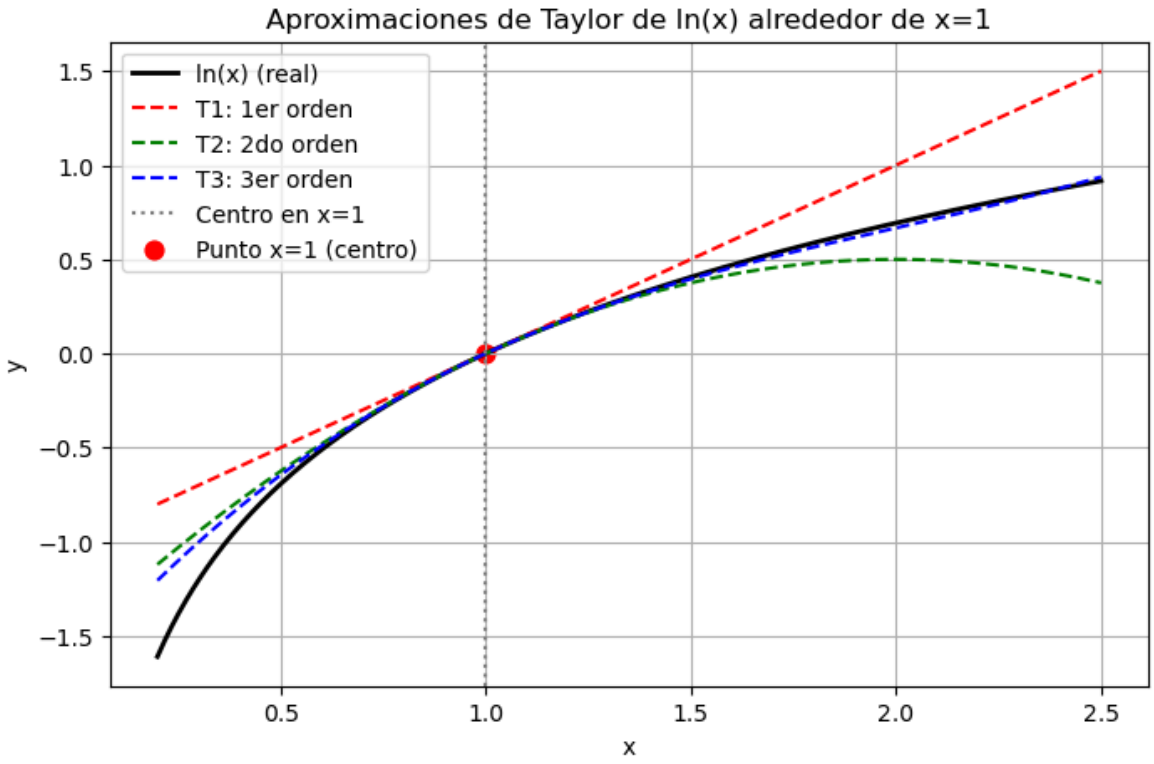
$$T_1 = (x-1)$$

$$T_2 = (x-1) - \frac{(x-1)^2}{2!}$$

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

In [1]:
```python
# %%
# Importar librerías
import numpy as np
import matplotlib.pyplot as plt

# %%
# Definir dominio (x > 0 porque ln(x) no está definido en 0)
x = np.linspace(0.2, 2.5, 400)

# Función real
y_real = np.log(x)

# Series de Taylor de ln(x) centradas en x = 1
T1 = (x - 1)
T2 = (x - 1) - (x - 1)**2 / 2
T3 = (x - 1) - (x - 1)**2 / 2 + (x - 1)**3 / 6

# %%
# Gráfica
plt.figure(figsize=(8,5))
plt.plot(x, y_real, 'k', label='ln(x) (real)', linewidth=2)
plt.plot(x, T1, '--r', label='T1: 1er orden')
```

```
plt.plot(x, T2, '--g', label='T2: 2do orden')
plt.plot(x, T3, '--b', label='T3: 3er orden')
plt.axvline(1, color='gray', linestyle=':', label='Centro en x=1')
plt.scatter(1, np.log(1), color='red', s=60, label='Punto x=1 (centro)')

# Detalles de la gráfica
plt.title("Aproximaciones de Taylor de ln(x) alrededor de x=1")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



1.b $f(x) = \ln(x), x_0 = 10$

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

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

$f'''(x) = \frac{2}{x^3}$

Polinomios de Taylor

$T_1 = \ln(10) + \frac{1}{10}(x - 10)$

In [ ]:

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

# %%
# Centro de expansión
x0 = 10

# Definir dominio (alrededor de x0)
x = np.linspace(5, 15, 400)
```

```
y_real = np.log(x)

# Series de Taylor de ln(x) centradas en x0 = 10
# ln(x) ≈ ln(x0) + (1/x0)(x - x0) - (1/(2x0^2))(x - x0)^2 + (1/(3x0^3))(x - x0)^

T1 = np.log(x0) + (1/x0)*(x - x0)
T2 = np.log(x0) + (1/x0)*(x - x0) - (1/(2*x0**2))*(x - x0)**2
T3 = np.log(x0) + (1/x0)*(x - x0) - (1/(2*x0**2))*(x - x0)**2 + (1/(3*x0**3))*(x

# %%
# Gráfica
plt.figure(figsize=(8,5))
plt.plot(x, y_real, 'k', label='ln(x) (real)', linewidth=2)
plt.plot(x, T1, '--r', label='T1: 1er orden')
plt.plot(x, T2, '--g', label='T2: 2do orden')
plt.plot(x, T3, '--b', label='T3: 3er orden')
plt.axvline(x0, color='gray', linestyle=':', label='Centro en x=10')

# 🔴 Agregar un punto en x = 10
plt.scatter(x0, np.log(x0), color='red', s=60, label='Punto x=10 (centro)')

# Detalles
plt.title("Aproximaciones de Taylor de ln(x) alrededor de x=10")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

2. $f(x) = \frac{1}{1-x}, x_0 = 0.5$

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

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

$$f'''(x) = \frac{6}{(1-x)^4}$$

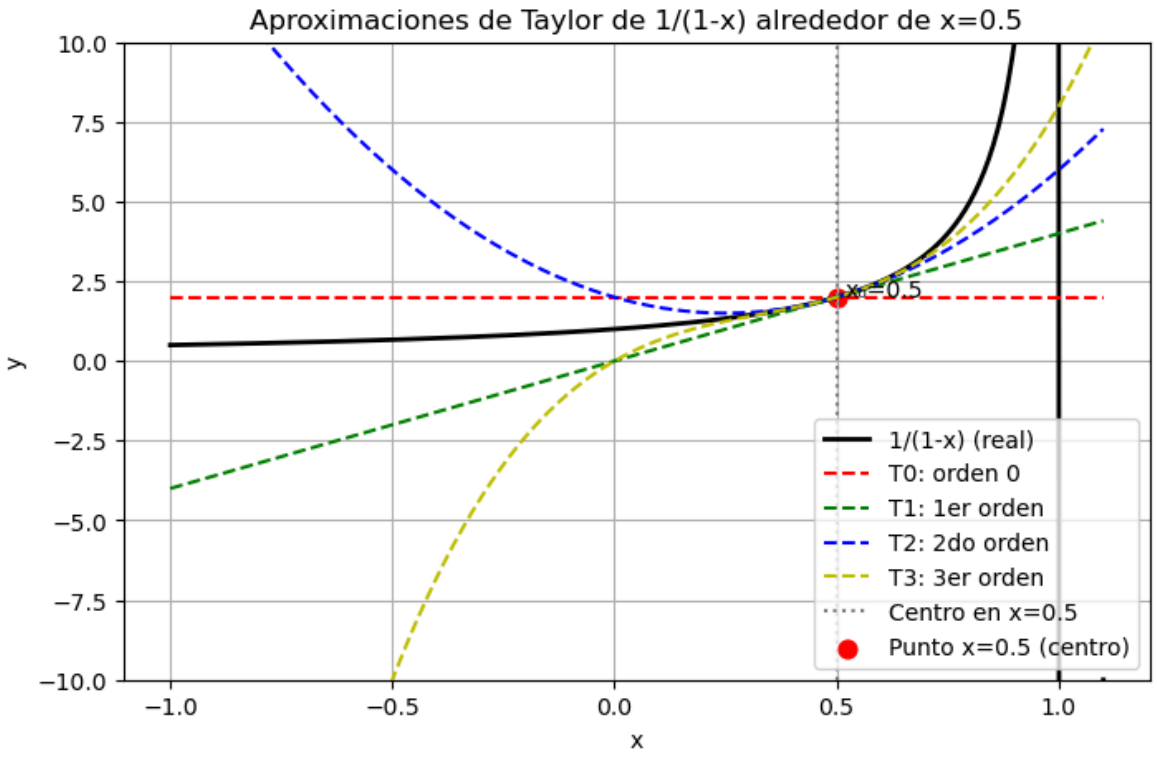
Polinomios de Taylor

$$T_0 = 2$$
$$T_1 = 2 + 4(x - 0.5)$$
$$T_2 = 2 + 4(x - 0.5) + 8(x - 0.5)^2$$
$$T_3 = 2 + 4(x - 0.5) + 8(x - 0.5)^2 + 16(x - 0.5)^3$$

In [10]:

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

# Definir dominio (evitar x = 1)
x = np.linspace(-1, 1.1, 400)
x = x[x != 1]

# Función real
y_real = 1 / (1 - x)

# Series de Taylor de 1/(1-x) centradas en x = 0.5
T0 = np.ones_like(x) * 2
T1 = 2 + 4 * (x - 0.5)
```

```
T2 = 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2
T3 = 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2 + 16 * (x - 0.5)**3

# Gráfica
plt.figure(figsize=(8,5))
plt.plot(x, y_real, 'k', label='1/(1-x) (real)', linewidth=2)
plt.plot(x, T0, '--r', label='T0: orden 0')
plt.plot(x, T1, '--g', label='T1: 1er orden')
plt.plot(x, T2, '--b', label='T2: 2do orden')
plt.plot(x, T3, '--y', label='T3: 3er orden')

# Línea vertical y punto del centro
plt.axvline(0.5, color='gray', linestyle=':', label='Centro en x=0.5')
plt.scatter(0.5, 2, color='red', s=60, label='Punto x=0.5 (centro)')
plt.text(0.5 + 0.02, 2, 'x₀=0.5', fontsize=10)

# Detalles
plt.ylim(-10, 10)
plt.title("Aproximaciones de Taylor de 1/(1-x) alrededor de x=0.5")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



3. $f(x) = \cos(x), x_0 = 0$

$f'(x) = -\sin(x)$

$f''(x) = -\cos(x)$

$f'''(x) = \sin(x)$

$f^{(4)}(x) = \cos(x)$

Polinomios de Taylor

$$T_0 = 1$$

$$T_1 = 1$$

$$T_2 = 1 - \frac{x^2}{2!}$$

$$T_3 = 1 - \frac{x^2}{2!}$$

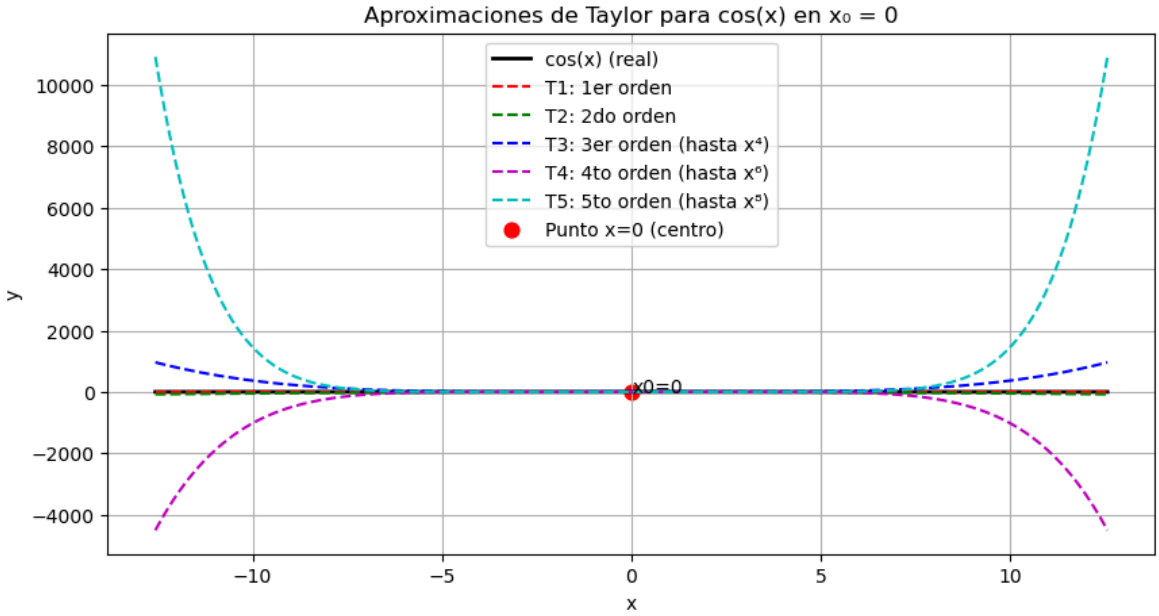
$$T_4 = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

$$T_5 = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

Los términos impares no cambian porque las derivadas impares de $\cos(x)$ son nulas en $x = 0$.

In [29]:
```
import numpy as np
import matplotlib.pyplot as plt

# Definir dominio
x = np.linspace(-4*np.pi, 4*np.pi, 400)
y_real = np.cos(x)

# Series de Taylor centradas en 0
T1 = np.ones_like(x)                              # Orden 1
T2 = 1 - x**2/2                                   # Orden 2
T3 = 1 - x**2/2 + x**4/24                         # Orden 4
T4 = 1 - x**2/2 + x**4/24 - x**6/720              # Orden 6
T5 = 1 - x**2/2 + x**4/24 - x**6/720 + x**8/40320 # Orden 8

# Graficar
plt.figure(figsize=(10,5))
plt.plot(x, y_real, 'k', label='cos(x) (real)', linewidth=2)
plt.plot(x, T1, '--r', label='T1: 1er orden')
plt.plot(x, T2, '--g', label='T2: 2do orden')
plt.plot(x, T3, '--b', label='T3: 3er orden (hasta x⁴)')
plt.plot(x, T4, '--m', label='T4: 4to orden (hasta x⁶)')
plt.plot(x, T5, '--c', label='T5: 5to orden (hasta x⁸)')
plt.scatter(0, np.cos(0), color='red', s=60, label='Punto x=0 (centro)')
plt.text(0+0.02, np.cos(0), 'x0=0', fontsize=10)


plt.legend()
plt.title('Aproximaciones de Taylor para cos(x) en x₀ = 0')
plt.xlabel('x')
plt.ylabel('y')
```

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



Polinomio de Langrange

1. Puntos $A:\ (0,0),\ (30,\frac{1}{2}),\ (60,3\sqrt{2}),\ (90,1).$

Tenemos que$:

$$P_A(x) = 0 \cdot L_0(x) + \frac{1}{2}\,L_1(x) + 3\sqrt{2}\,L_2(x) + 1 \cdot L_3(x),$$

donde: $L_i(x)$

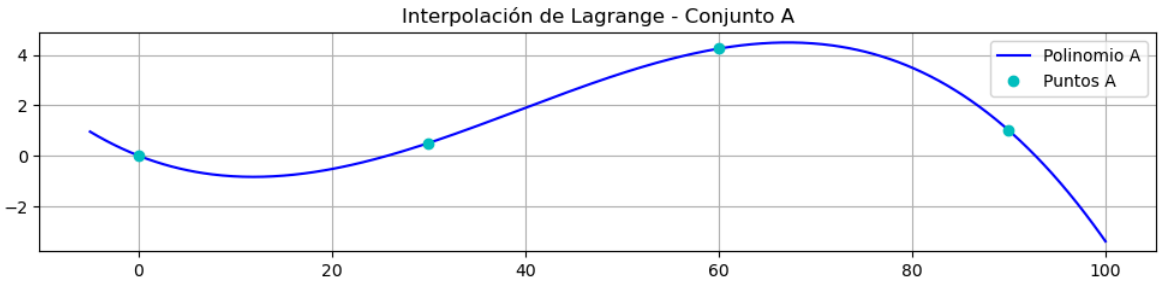
Polinomio expandido:

$$\boxed{P_A(x) = -\frac{\sqrt{2}}{18000}x^3 + \frac{1}{64800}x^3 - \frac{7}{3600}x^2 + \frac{\sqrt{2}}{150}x^2 - \frac{3\sqrt{2}}{20}x + \frac{11}{180}x}$$

In [48]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

x_a = np.array([0, 30, 60, 90])
y_a = np.array([0, 0.5, 3 * np.sqrt(2), 1])
poly_a = lagrange(x_a, y_a)


print("Polinomio de Lagrange - Conjunto A:")
print(np.poly1d(poly_a), "\n")
# === Graficar ===
x_vals = np.linspace(-5, 100, 500)
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(x_vals, poly_a(x_vals), label='Polinomio A', color='blue')
plt.plot(x_a, y_a, 'co', label='Puntos A')
plt.title('Interpolación de Lagrange - Conjunto A')
```

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

```
Polinomio de Lagrange - Conjunto A:
            3            2
-6.314e-05 x + 0.007484 x - 0.151 x
```



Conjunto B: $(1,1),\ (2,2),\ (3,2).$

Tenemos que:

$$P_B(x) = 1 \cdot L_0(x) + 2 \cdot L_1(x) + 2 \cdot L_2(x),$$

$$Donde: L_0(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{(x-2)(x-3)}{2},$$

$$L_1(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3),$$

$$L_2(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{(x-1)(x-2)}{2}.$$

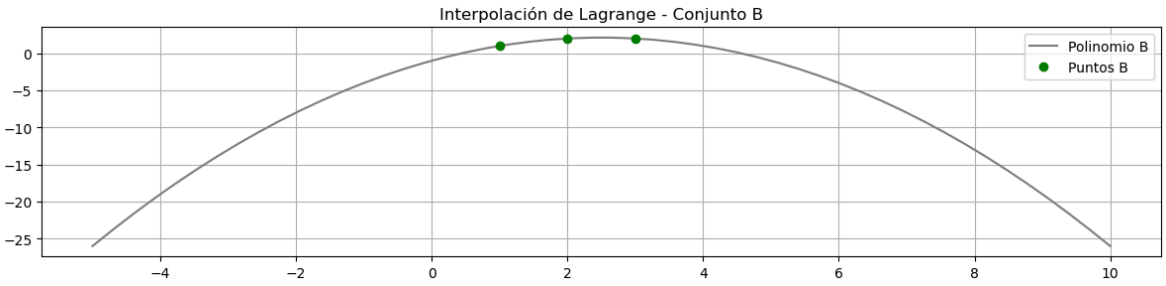
$$P_B(x) = \frac{(x-2)(x-3)}{2} + 2\big[-(x-1)(x-3)\big] + 2 \cdot \frac{(x-1)(x-2)}{2}$$

$$\boxed{P_B(x) = -\frac{1}{2}x^2 + \frac{5}{2}x - 1}$$

In [58]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

x_b = np.array([1, 2, 3])
y_b = np.array([1, 2, 2])
poly_b = lagrange(x_b, y_b)

print("Polinomio de Lagrange - Conjunto B:")
print(np.poly1d(poly_b), "\n")
# === Graficar ===
x_vals = np.linspace(-5, 10, 50)
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 2)
plt.plot(x_vals, poly_b(x_vals), label='Polinomio B', color='gray')
plt.plot(x_b, y_b, 'go', label='Puntos B')
plt.title('Interpolación de Lagrange - Conjunto B')
plt.legend()
plt.grid(True)
```

```
plt.tight_layout()
plt.show()
```

```
Polinomio de Lagrange - Conjunto B:
      2
-0.5 x + 2.5 x - 1
```



C. Puntos $C:\ (-2,5),\ (1,7),\ (3,11),\ (7,34).$

Tenemos que$:

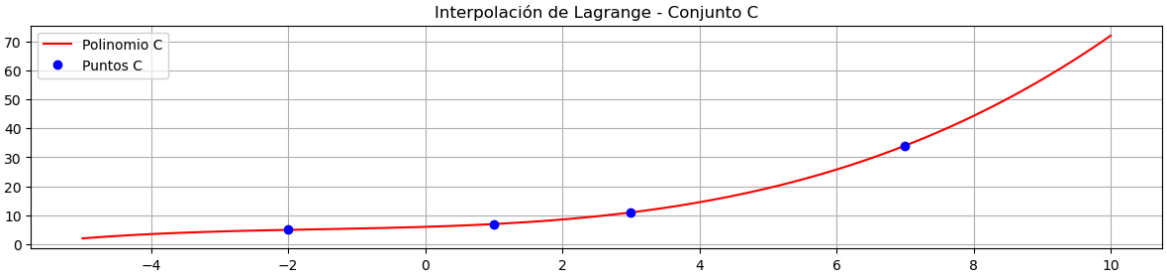
$$P_C(x) = 5\,L_0(x) + 7\,L_1(x) + 11\,L_2(x) + 34\,L_3(x),$$

con $L_0(x) = \frac{(x-1)(x-3)(x-7)}{(-2-1)(-2-3)(-2-7)},\ \ldots$

Polinomio: $\boxed{P_C(x) = \frac{43}{1080}x^3 + \frac{101}{540}x^2 + \frac{793}{1080}x + \frac{1087}{180}}$

In [59]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

# === Conjunto  ===
x_c = np.array([-2, 1, 3, 7])
y_c = np.array([5, 7, 11, 34])
poly_c = lagrange(x_c, y_c)

print("Polinomio de Lagrange - Conjunto C:")
print(np.poly1d(poly_c), "\n")
#graficar
x_vals = np.linspace(-5, 10, 500)

plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 3)
plt.plot(x_vals, poly_c(x_vals), label='Polinomio C', color='red')
plt.plot(x_c, y_c, 'bo', label='Puntos C')
plt.title('Interpolación de Lagrange - Conjunto C')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Polinomio de Lagrange - Conjunto C:
         3         2
0.03981 x + 0.187 x + 0.7343 x + 6.039
```


Interpolación de Lagrange - Conjunto C
Polinomio C
Puntos C
70
60
50
40
30
20
10
0
−4
−2
0
2
4
6
8
10