TallerN3SerieTaylor_Lagrange

November 12, 2025

1 Escuela Politécnica Nacional

1.1 Métodos Numéricos

Nombre: Lenin Amangandi

Tema: Series de Taylor y polinomios de Lagrange

Link al repositorio Taller N3

1.2 1. Grafique las curvas de las series de Taylor de varios órdenes para los siguientes casos:

1.2.1 Caso 1: cos(x) alrededor de

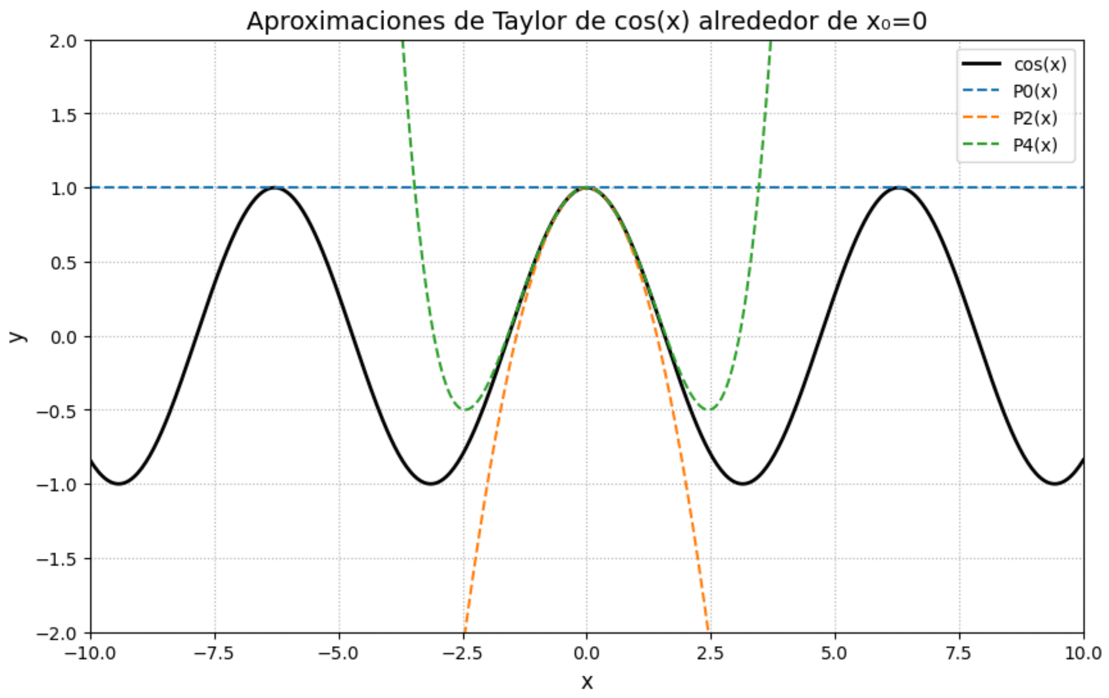
$$x_0 = 0$$

, n=4

- **Orden 0:**
$$P_0(x) = 1$$
- **Orden 1:**
$$P_1(x) = 1$$
- **Orden 2:**
$$P_2(x) = 1 - \frac{x^2}{2}$$
- **Orden 3:**
$$P_3(x) = 1 - \frac{x^2}{2}$$
- **Orden 4:**
$$P_4(x) = 1 - \frac{x^2}{2} + \frac{x^4}{24}$$

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

x_min = -10
x_max = 10
y_min = -2
y_max = 2
num_puntos = 500
```

```python
def f_cos(x):
    return np.cos(x)

def P0(x):
    return np.ones_like(x)

def P1(x):
    return np.ones_like(x)

def P2(x):
    return 1 - (x**2)/2

def P3(x):
    return 1 - (x**2)/2

def P4(x):
    return 1 - (x**2)/2 + (x**4)/24


x = np.linspace(x_min, x_max, num_puntos)

plt.figure(figsize=(10, 6))
plt.plot(x, f_cos(x), 'k', label='cos(x)', linewidth=2)
plt.plot(x, P0(x), '--', label='P0(x)')
plt.plot(x, P2(x), '--', label='P2(x)')
plt.plot(x, P4(x), '--', label='P4(x)')

plt.title("Aproximaciones de Taylor de cos(x) alrededor de x =0", fontsize=14)
plt.xlabel("x", fontsize=12)
plt.ylabel("y", fontsize=12)
plt.grid(True, linestyle=':')
plt.legend()
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.show()
```



1.2.2 Caso 2:

$$\frac{1}{1-x}$$

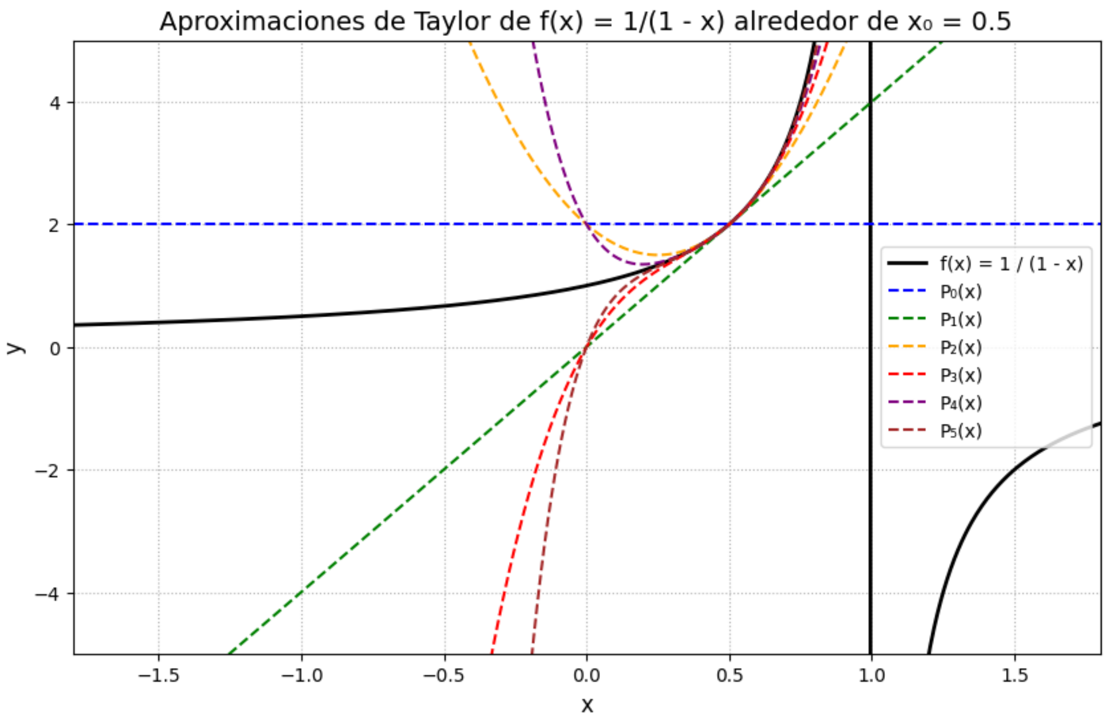
alrededor de

$$x_0 = 0.5$$

, n = 5

- Orden 0:
$$P_0(x) = 2$$
- Orden 1:
$$P_1(x) = 2 + 4(x - 0.5)$$
- Orden 2:
$$P_2(x) = 2 + 4(x - 0.5) + 8(x - 0.5)^2$$
- Orden 3:
$$P_3(x) = 2 + 4(x - 0.5) + 8(x - 0.5)^2 + 16(x - 0.5)^3$$
- Orden 4:
$$P_4(x) = 2 + 4(x - 0.5) + 8(x - 0.5)^2 + 16(x - 0.5)^3 + 32(x - 0.5)^4$$
- Orden 5:
$$P_5(x) = 2 + 4(x - 0.5) + 8(x - 0.5)^2 + 16(x - 0.5)^3 + 32(x - 0.5)^4 + 64(x - 0.5)^5$$

[11]:
```python
import numpy as np
import matplotlib.pyplot as plt

x_min = -1.8
x_max = 1.8
y_min = -5
y_max = 5
num_puntos = 600
x0 = 0.5

def f_racional(x):
    y = np.empty_like(x)
    y[:] = np.nan
    mask = (x != 1)
    y[mask] = 1 / (1 - x[mask])
    return y

def P0(x):
    return np.ones_like(x) * (1 / (1 - x0))

def P1(x): return 2 + 4 * (x - 0.5)
def P2(x): return 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2
def P3(x): return 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2 + 16 * (x - 0.5)**3
def P4(x): return 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2 + 16 * (x - 0.5)**3 + 32␣
 ↪* (x - 0.5)**4
def P5(x): return 2 + 4 * (x - 0.5) + 8 * (x - 0.5)**2 + 16 * (x - 0.5)**3 + 32␣
 ↪* (x - 0.5)**4 + 64 * (x - 0.5)**5

x = np.linspace(x_min, x_max, num_puntos)
x = x[x != 1]

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

plt.plot(x, f_racional(x), 'k', label='f(x) = 1 / (1 - x)', linewidth=2)

plt.plot(x, P0(x), '--', color='blue', label='P (x)')
plt.plot(x, P1(x), '--', color='green', label='P (x)')
plt.plot(x, P2(x), '--', color='orange', label='P (x)')
plt.plot(x, P3(x), '--', color='red', label='P (x)')
plt.plot(x, P4(x), '--', color='purple', label='P (x)')
plt.plot(x, P5(x), '--', color='brown', label='P (x)')

plt.title("Aproximaciones de Taylor de f(x) = 1/(1 - x) alrededor de x  = 0.5",␣
 ↪fontsize=14)
plt.xlabel("x", fontsize=12)
plt.ylabel("y", fontsize=12)
plt.grid(True, linestyle=':')
plt.legend()
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```



1.2.3 Caso 3: ln(x) alrededor de x_0 = 1, n=3

- Orden 0:

$$P_0(x) = ln(1)$$

- Orden 1:

$$P_1(x) = (x-1)$$

- Orden 2:

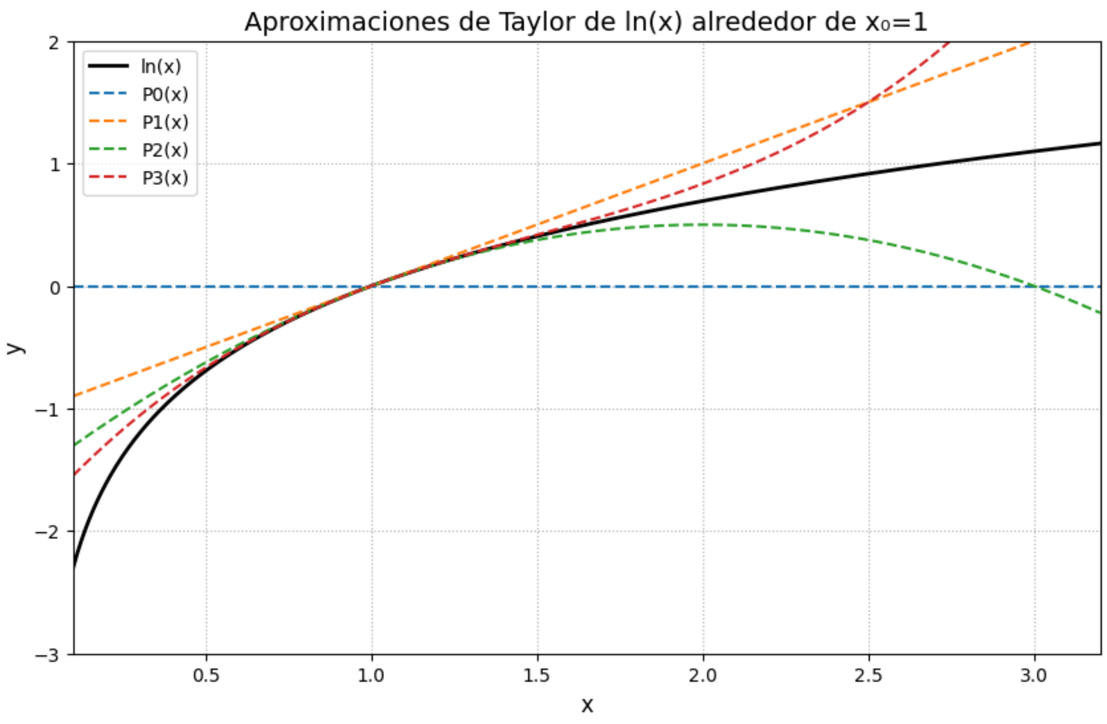
$$P_2(x) = (x-1) - \frac{(x-1)^2}{2}$$

- Orden 3:

$$P_3(x) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3}$$

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

x_min = 0.1
x_max = 3.2
y_min = -3
```

```python
y_max = 2
num_puntos = 500

def f_ln(x):
    return np.log(x)

def P0(x):
    return np.zeros_like(x)

def P1(x):
    return (x - 1)

def P2(x):
    return (x - 1) - ((x - 1)**2)/2

def P3(x):
    return (x - 1) - ((x - 1)**2)/2 + ((x - 1)**3)/3

x = np.linspace(x_min, x_max, num_puntos)

plt.figure(figsize=(10, 6))
plt.plot(x, f_ln(x), 'k', label='ln(x)', linewidth=2)
plt.plot(x, P0(x), '--', label='P0(x)')
plt.plot(x, P1(x), '--', label='P1(x)')
plt.plot(x, P2(x), '--', label='P2(x)')
plt.plot(x, P3(x), '--', label='P3(x)')

plt.title("Aproximaciones de Taylor de ln(x) alrededor de x =1", fontsize=14)
plt.xlabel("x", fontsize=12)
plt.ylabel("y", fontsize=12)
plt.grid(True, linestyle=':')
plt.legend()
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.show()
```



1.2.4 Caso 3.1: ln(x) alrededor de x_0 = 10, n=3

- Orden 0:

$$P_0(x) = ln(10)$$

- Orden 1:

$$P_1(x) = ln(10) + \frac{(x-10)}{10}$$

- Orden 2:

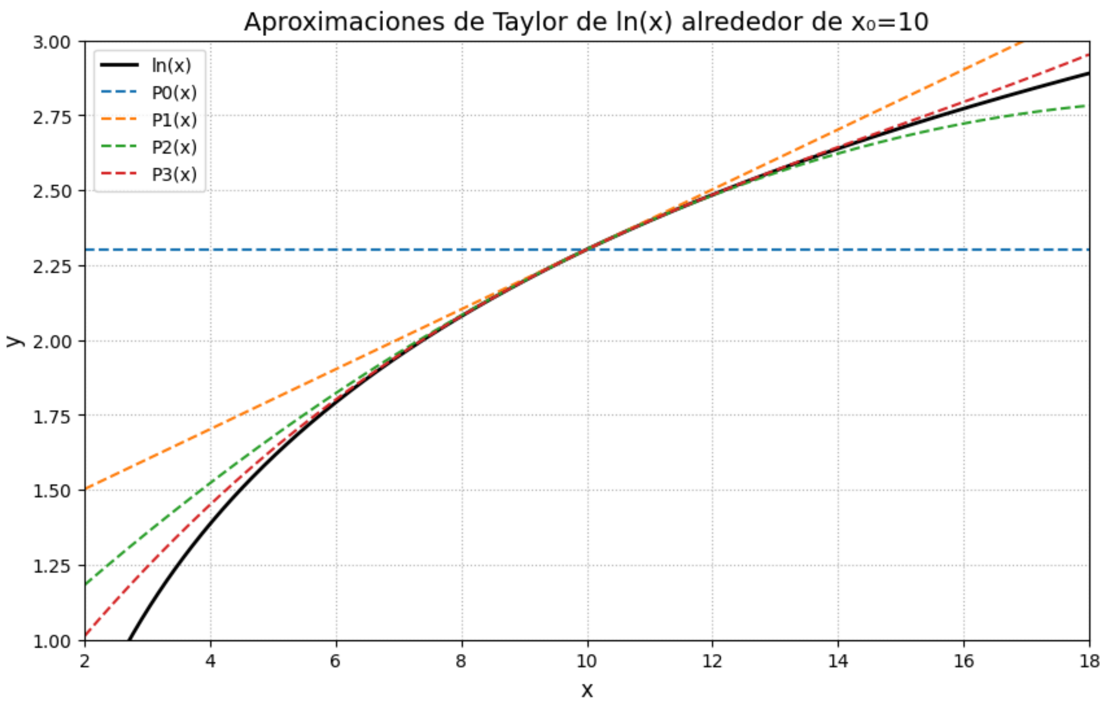
$$P_2(x) = ln(10) + \frac{(x-10)}{10} - \frac{(x-10)^2}{2 \cdot 10^2}$$

- Orden 3:

$$P_3(x) = ln(10) + \frac{(x-10)}{10} - \frac{(x-10)^2}{2 \cdot 10^2} + \frac{(x-10)^3}{3 \cdot 10^3}$$

[16]:
```
import numpy as np
import matplotlib.pyplot as plt

x_min = 2
x_max = 18
y_min = 1
y_max = 3
num_puntos = 600

def f_ln(x):
    return np.log(x)
```

```python
def P0(x):
    return np.log(10) * np.ones_like(x)

def P1(x):
    return np.log(10) + (x - 10)/10

def P2(x):
    return np.log(10) + (x - 10)/10 - ((x - 10)**2) / (2 * 10**2)

def P3(x):
    return np.log(10) + (x - 10)/10 - ((x - 10)**2) / (2 * 10**2) + ((x -␣
 ↪10)**3) / (3 * 10**3)

x = np.linspace(x_min, x_max, num_puntos)

plt.figure(figsize=(10, 6))
plt.plot(x, f_ln(x), 'k', label='ln(x)', linewidth=2)
plt.plot(x, P0(x), '--', label='P0(x)')
plt.plot(x, P1(x), '--', label='P1(x)')
plt.plot(x, P2(x), '--', label='P2(x)')
plt.plot(x, P3(x), '--', label='P3(x)')

plt.title("Aproximaciones de Taylor de ln(x) alrededor de x =10", fontsize=14)
plt.xlabel("x", fontsize=12)
plt.ylabel("y", fontsize=12)
plt.grid(True, linestyle=':')
plt.legend()
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.show()
```



1.3 2. Encuentre el polinomio de Lagrange para los siguientes datos y grafique:

1.3.1 Caso 1) (0,0), (30,0.5), (60,√3/2),(90.1)

$$(x_0, y_0) = (0, 0), \quad (x_1, y_1) = (30, 0.5), \quad (x_2, y_2) = (60, \tfrac{\sqrt{3}}{2}), \quad (x_3, y_3) = (90, 1)$$

- Para (i = 0)

$$L_0(x) = \frac{(x-30)(x-60)(x-90)}{(0-30)(0-60)(0-90)} = \frac{(x-30)(x-60)(x-90)}{(-30)(-60)(-90)}$$

- Para (i = 1)

$$L_1(x) = \frac{(x-0)(x-60)(x-90)}{(30-0)(30-60)(30-90)} = \frac{x(x-60)(x-90)}{30 \cdot (-30) \cdot (-60)}$$

- Para (i = 2)

$$L_2(x) = \frac{(x-0)(x-30)(x-90)}{(60-0)(60-30)(60-90)} = \frac{x(x-30)(x-90)}{60 \cdot 30 \cdot (-30)}$$

- Para (i = 3)

$$L_3(x) = \frac{(x-0)(x-30)(x-60)}{(90-0)(90-30)(90-60)} = \frac{x(x-30)(x-60)}{90 \cdot 60 \cdot 30}$$

Polinomio interpolador es:

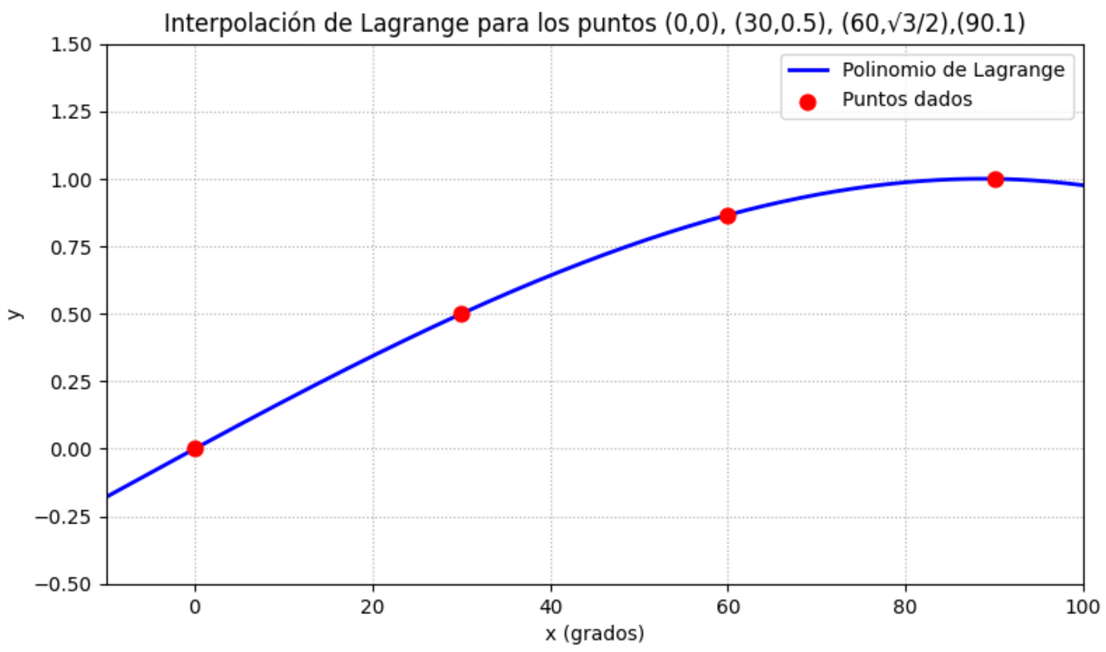
$$P(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x)$$

$$P(x) = \frac{1}{2}L_1(x) + \frac{\sqrt{3}}{2}L_2(x) + 1 \cdot L_3(x)$$

- $$P(x) = 0.5 \cdot \frac{x(x-60)(x-90)}{54000} + \frac{\sqrt{3}}{2} \cdot \left(-\frac{x(x-30)(x-90)}{54000}\right) + \frac{x(x-30)(x-60)}{162000}$$

- $$P(x) = \frac{0.5\,x(x-60)(x-90)}{54000} - \frac{\frac{\sqrt{3}}{2}\,x(x-30)(x-90)}{54000} + \frac{x(x-30)(x-60)}{162000}$$

[3]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

x_pts = np.array([0.0, 30.0, 60.0, 90.0])
y_pts = np.array([0.0, 0.5, np.sqrt(3)/2, 1.0])

P = lagrange(x_pts, y_pts)

x_min, x_max = -10, 100
y_min, y_max = -0.5, 1.5

x_vals = np.linspace(x_min, x_max, 800)
y_vals = P(x_vals)

plt.figure(figsize=(9, 5))
plt.plot(x_vals, y_vals, label='Polinomio de Lagrange', color='blue',␣
 ↪linewidth=2)
plt.scatter(x_pts, y_pts, color='red', label='Puntos dados', zorder=5, s=60)

plt.xlabel("x (grados)")
plt.ylabel("y")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.title("Interpolación de Lagrange para los puntos (0,0), (30,0.5), (60,√3/
 ↪2),(90.1)")
plt.grid(True, linestyle=':')
plt.legend()
plt.show()
```



1.3.2 Caso 2) (1,1),(2,2),(3,2)

$$(x_0, y_0) = (1, 1), \quad (x_1, y_1) = (2, 2), \quad (x_2, y_2) = (3, 2)$$

- Para (i=0) (con (x_0=1)):

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

- Para (i=1) (con (x_1=2)):

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

- Para (i=2) (con (x_2=3)):

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

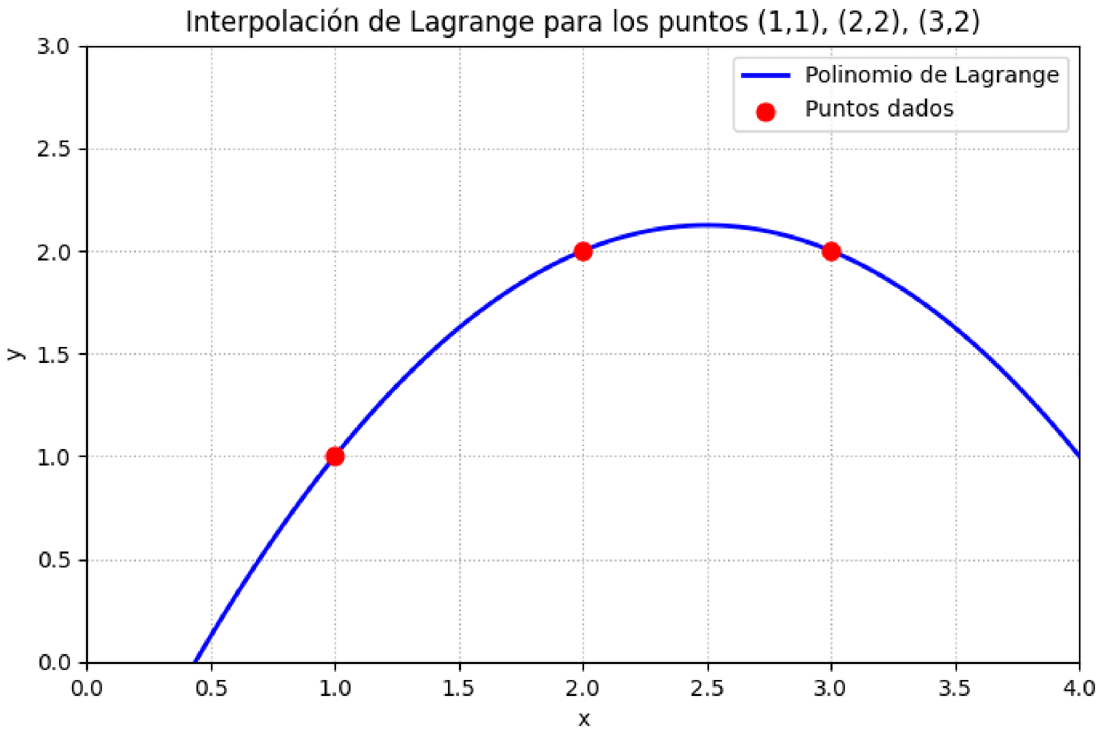
Polinomio interpolador:

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

$$P(x) = \frac{(x-2)(x-3)}{2} \; - \; 2\,(x-1)(x-3) \; + \; 2 \cdot \frac{(x-1)(x-2)}{2}$$

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

[4]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

x_pts = np.array([1.0, 2.0, 3.0])
y_pts = np.array([1.0, 2.0, 2.0])

P = lagrange(x_pts, y_pts)

x_min, x_max = 0, 4
y_min, y_max = 0, 3

x_vals = np.linspace(x_min, x_max, 400)
y_vals = P(x_vals)

plt.figure(figsize=(8, 5))
plt.plot(x_vals, y_vals, label='Polinomio de Lagrange', color='blue',␣
 ↪linewidth=2)
plt.scatter(x_pts, y_pts, color='red', label='Puntos dados', zorder=5, s=60)

plt.title("Interpolación de Lagrange para los puntos (1,1), (2,2), (3,2)")
plt.xlabel("x")
plt.ylabel("y")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.grid(True, linestyle=':')
plt.legend()
plt.show()
```



1.3.3 Caso 3) (-2,5), (1,7), (3,11), (7,34)

$$(x_0, y_0) = (-2, 5), \quad (x_1, y_1) = (1, 7), \quad (x_2, y_2) = (3, 11), \quad (x_3, y_3) = (7, 34)$$

- Para (i=0) (con (x_0=-2))

$$L_0(x) = \frac{(x-1)(x-3)(x-7)}{(-2-1)(-2-3)(-2-7)} = \frac{(x-1)(x-3)(x-7)}{(-3)(-5)(-9)} = \frac{(x-1)(x-3)(x-7)}{-135}$$

- Para (i=1) (con (x_1=1))

$$L_1(x) = \frac{(x+2)(x-3)(x-7)}{(1-(-2))(1-3)(1-7)} = \frac{(x+2)(x-3)(x-7)}{3 \cdot (-2) \cdot (-6)} = \frac{(x+2)(x-3)(x-7)}{36}$$

- Para (i=2) (con (x_2=3))

$$L_2(x) = \frac{(x+2)(x-1)(x-7)}{(3-(-2))(3-1)(3-7)} = \frac{(x+2)(x-1)(x-7)}{5 \cdot 2 \cdot (-4)} = \frac{(x+2)(x-1)(x-7)}{-40}$$

- Para (i=3) (con (x_3=7))

$$L_3(x) = \frac{(x+2)(x-1)(x-3)}{(7-(-2))(7-1)(7-3)} = \frac{(x+2)(x-1)(x-3)}{9 \cdot 6 \cdot 4} = \frac{(x+2)(x-1)(x-3)}{216}$$

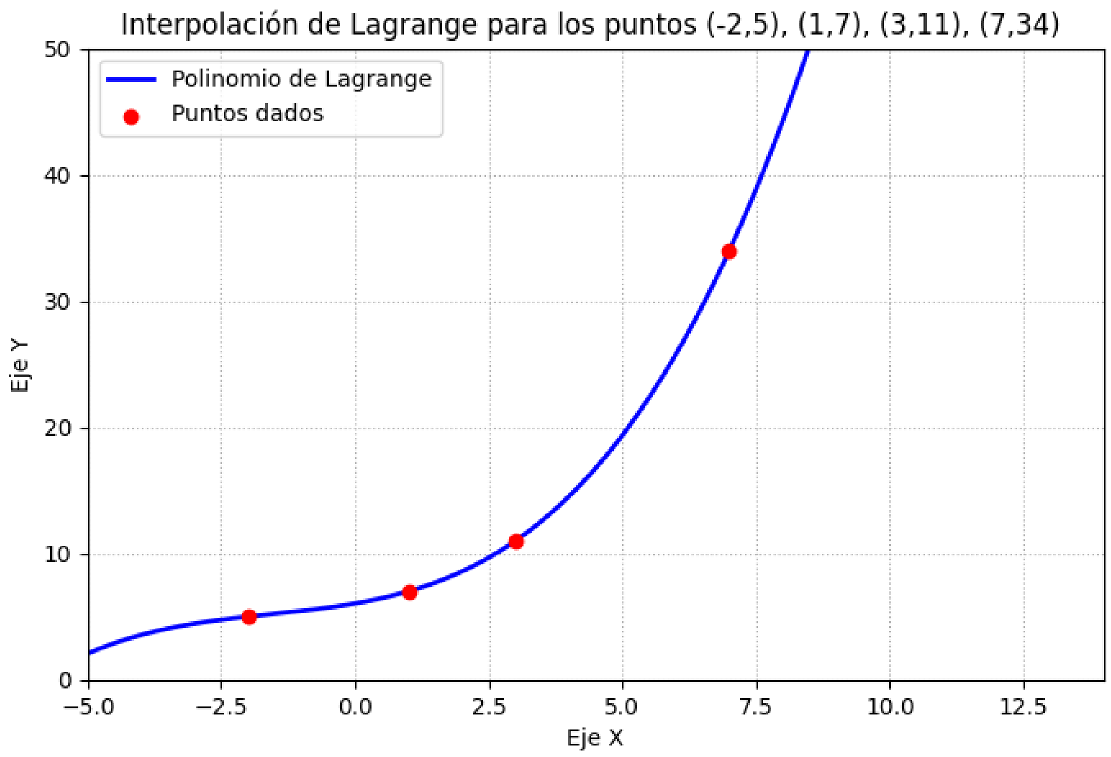
Polinomio interpolador:

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

$$P(x) = 5\cdot\frac{(x-1)(x-3)(x-7)}{-135} + 7\cdot\frac{(x+2)(x-3)(x-7)}{36} + 11\cdot\frac{(x+2)(x-1)(x-7)}{-40} + 34\cdot\frac{(x+2)(x-1)(x-3)}{216}$$

$$P(x) = -\frac{1}{27}(x-1)(x-3)(x-7) + \frac{7}{36}(x+2)(x-3)(x-7) - \frac{11}{40}(x+2)(x-1)(x-7) + \frac{17}{108}(x+2)(x-1)(x-3)$$

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

x_pts = np.array([-2, 1, 3, 7])
y_pts = np.array([5, 7, 11, 34])

P = lagrange(x_pts, y_pts)

x_min, x_max = -5, 14
y_min, y_max = 0, 50

x_vals = np.linspace(x_min, x_max, 500)
y_vals = P(x_vals)

plt.figure(figsize=(8, 5))
plt.plot(x_vals, y_vals, label='Polinomio de Lagrange', color='blue',␣
  ↪linewidth=2)
plt.scatter(x_pts, y_pts, color='red', label='Puntos dados', zorder=5)

plt.title("Interpolación de Lagrange para los puntos (-2,5), (1,7), (3,11),␣
  ↪(7,34) ")
plt.xlabel("Eje X")
plt.ylabel("Eje Y")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.grid(True, linestyle=':')
plt.legend()
plt.show()
```



Interpolación de Lagrange para los puntos (-2,5), (1,7), (3,11), (7,34)
Polinomio de Lagrange
Puntos dados
Eje Y
Eje X
50
40
30
20
10
0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5