Taller 05

Nombre: David Alejandro Díaz Pineda

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

A) Interpole los puntos:

$p1 = (5.4, 3.2)$
$p2 = (9.5, 0.7)$
$p3 = (12.3, -3.6)$

In [2]:
```
# Derivadas parciales para regresión lineal
# ##############################################################################
def der_parcial_1(xs: list, ys: list) -> tuple[float, float, float]:
    """Retorna los coeficientes de la ecuación de la derivada parcial con respec
    c_1 * a_1 + c_0 * a_0 = c_ind


    ## Parameters

    ``xs``: lista de valores de x.

    ``ys``: lista de valores de y.

    ## Return

    ``c_1``: coeficiente del parámetro 1.

    ``c_0``: coeficiente del parámetro 0.

    ``c_ind``: coeficiente del término independiente.

    """

    # coeficiente del término independiente
    c_ind = sum(ys)

    # coeficiente del parámetro 1
    c_1 = sum(xs)

    # coeficiente del parámetro 0
    c_0 = len(xs)

    return (c_1, c_0, c_ind)


def der_parcial_0(xs: list, ys: list) -> tuple[float, float, float]:
    """Retorna los coeficientes de la ecuación de la derivada parcial con respec
    c_1 * a_1 + c_0 * a_0 = c_ind
```

```
    ## Parameters

    ``xs``: lista de valores de x.

    ``ys``: lista de valores de y.

    ## Return

    ``c_1``: coeficiente del parámetro 1.

    ``c_0``: coeficiente del parámetro 0.

    ``c_ind``: coeficiente del término independiente.

    """
    c_1 = 0
    c_0 = 0
    c_ind = 0
    for xi, yi in zip(xs, ys):
        # coeficiente del término independiente
        c_ind += xi * yi

        # coeficiente del parámetro 1
        c_1 += xi * xi

        # coeficiente del parámetro 0
        c_0 += xi

    return (c_1, c_0, c_ind)
```

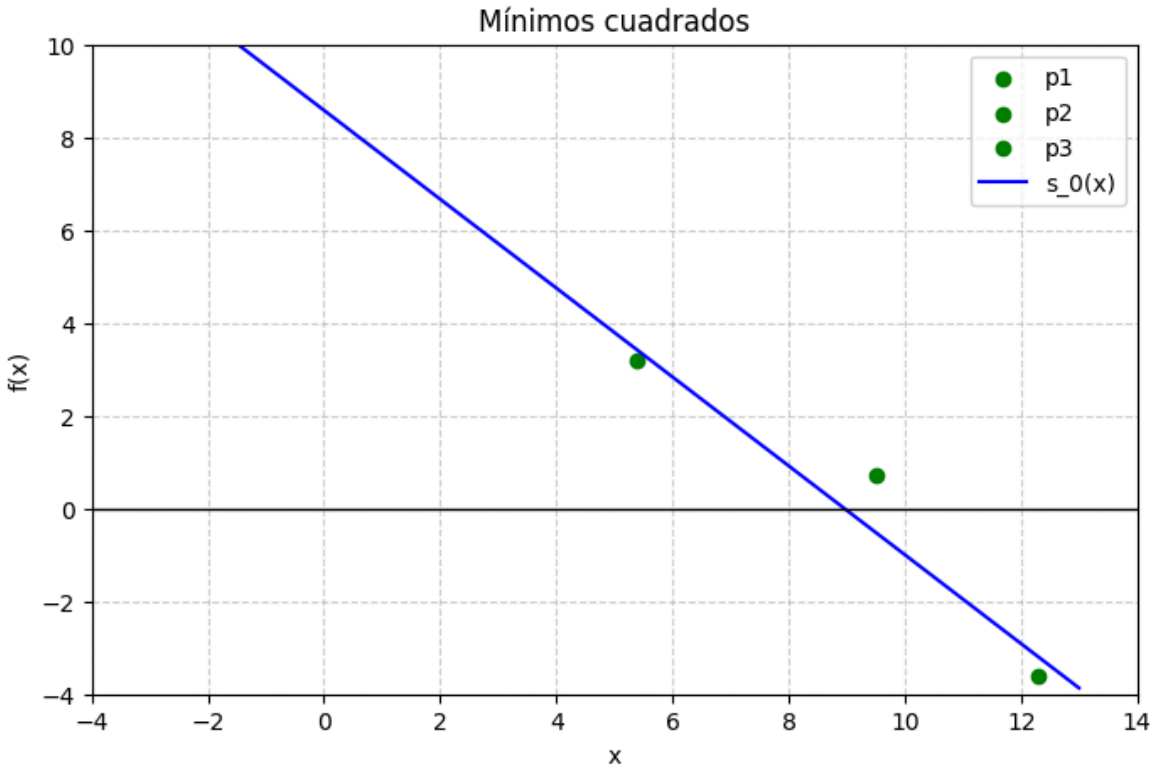
In [3]:
```
from src import ajustar_min_cuadrados

# UTILICE:
ajustar_min_cuadrados([1, 4, 5, 6], [2, 3, 5, 7], [der_parcial_1, der_parcial_0]
```

```
[12-15 21:25:30][INFO] dalz2| 2025-12-15 21:25:30.421595
[12-15 21:25:30][INFO] Se ajustarán 2 parámetros.
```

Out[3]:
```
array([0.92857143, 0.53571429])
```

In [4]:
```
x1, y1 = 5.4, 3.2
x2, y2 = 9.5, 0.7
x3, y3 = 12.3, -3.6
m, b = -0.9578, 8.584

xs0 = np.linspace(-4, 13, 200)


def s_0(x):
    return m * x + b


plt.figure(figsize=(8, 5))
plt.scatter([x1], [y1], color='green', label='p1')
plt.scatter([x2], [y2], color='green', label='p2')
plt.scatter([x3], [y3], color='green', label='p3')

plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')

plt.axhline(0, color='black', linewidth=1)
```

```
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Mínimos cuadrados')
plt.legend()
plt.xlim(-4, 14)
plt.ylim(-4, 10)
plt.show()
```



B) Interpole el siguiente conjunto de datos:

Conjunto 1:

In [5]:
```
ajustar_min_cuadrados([3.38,
0.35,
2.07,
-0.45,
-0.55,
-1.06,
-2.77,
3.08,
-3.98,
-5,
-3.18,
-1.96,
2.37,
-1.66,
4.09], [-0.04,
1.28,
0.65,
2.18,
1.70,
```

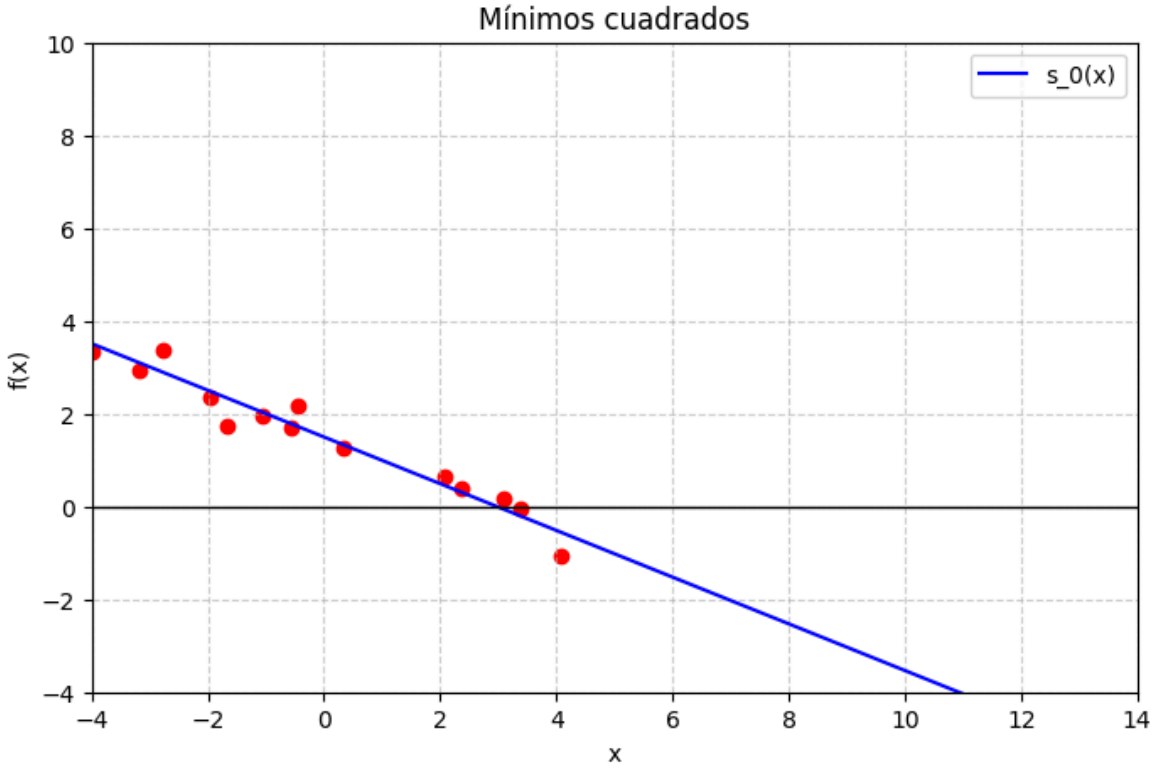
```
1.96,
3.36,
0.18,
3.35,
4.16,
2.95,
2.34,
0.38,
1.75,
-1.06], [der_parcial_1, der_parcial_0])
```

```
[12-15 21:25:31][INFO] Se ajustarán 2 parámetros.
```

Out[5]:
```
array([-0.50323259,  1.49919762])
```

In [6]:
```
xs = [ 3.38,
0.35,
2.07,
-0.45,
-0.55,
-1.06,
-2.77,
3.08,
-3.98,
-5,
-3.18,
-1.96,
2.37,
-1.66,
4.09]

ys = [-0.04,
1.28,
0.65,
2.18,
1.70,
1.96,
3.36,
0.18,
3.35,
4.16,
2.95,
2.34,
0.38,
1.75,
-1.06]

m,b = -0.50323259, 1.49919762
xs0 = np.linspace(-4, 13, 200)


def s_0(x):
    return m * x + b


plt.figure(figsize=(8, 5))
for xi, yi in zip(xs, ys):
    plt.scatter([xi], [yi], color='red')
```

```
plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')

plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Mínimos cuadrados')
plt.legend()
plt.xlim(-4, 14)
plt.ylim(-4, 10)
plt.show()
```



Conjunto 2:

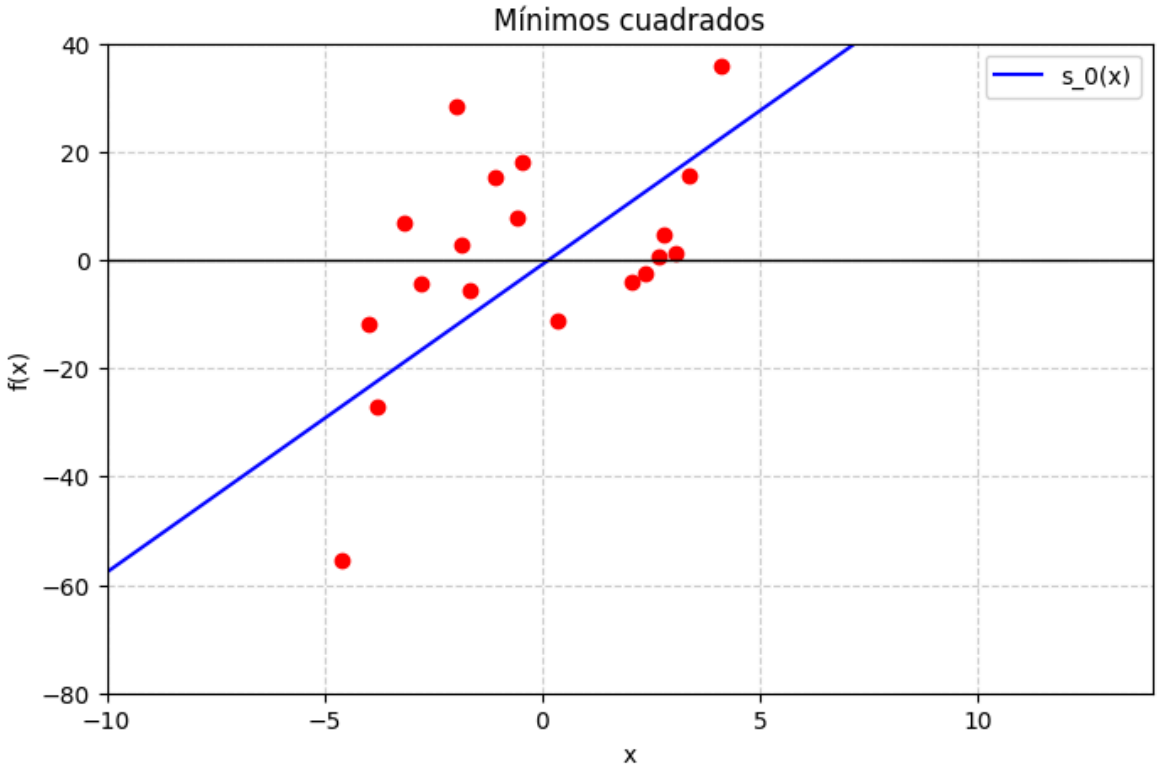
In [7]:
```
ajustar_min_cuadrados([3.38, 0.35, 2.07, -0.45, -0.56, -1.06, -2.78, 3.08, -3.99
 -3.18, -1.97, 2.37, -1.67, 4.09, -4.60, 2.68, 2.78, -3.79, -1.87]
, [15.65, -11.20, -4.00, 18.03, 7.94, 15.32, -4.40, 1.39, -11.92, -90.24,
 6.92, 28.35, -2.41, -5.47, 35.91, -55.53, 0.77, 4.79, -27.05, 2.85]
, [der_parcial_1, der_parcial_0])
```

```
[12-15 21:25:31][INFO] Se ajustarán 2 parámetros.
```

Out[7]:
```
array([ 5.68666556, -0.83754722])
```

In [8]:
```
xs = [3.38, 0.35, 2.07, -0.45, -0.56, -1.06, -2.78, 3.08, -3.99, -5.00,
 -3.18, -1.97, 2.37, -1.67, 4.09, -4.60, 2.68, 2.78, -3.79, -1.87]


ys = [15.65, -11.20, -4.00, 18.03, 7.94, 15.32, -4.40, 1.39, -11.92, -90.24,
 6.92, 28.35, -2.41, -5.47, 35.91, -55.53, 0.77, 4.79, -27.05, 2.85]


m,b = 5.68666556, -0.83754722
xs0 = np.linspace(-10, 13, 200)
```

```
def s_0(x):
    return m * x + b


plt.figure(figsize=(8, 5))
for xi, yi in zip(xs, ys):
    plt.scatter([xi], [yi], color='red')



plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')

plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Mínimos cuadrados')
plt.legend()
plt.xlim(-10, 14)
plt.ylim(-80, 40)
plt.show()
```



Hay mucho error, por eso conviene más usar una ecuación cúbica

In [9]:
```
# ######################################################################
def der_parcial_0(xs: list, ys: list) -> tuple[float, float, float, float, float
    c3 = c2 = c1 = 0.0
    c0 = len(xs)
    c_ind = sum(ys)

    for xi in xs:
        c3 += xi**3
        c2 += xi**2
        c1 += xi
```

```
    return (c3, c2, c1, c0, c_ind)


def der_parcial_1(xs: list, ys: list) -> tuple[float, float, float, float, float
    c3 = c2 = c1 = c0 = c_ind = 0.0

    for xi, yi in zip(xs, ys):
        c3 += xi**4
        c2 += xi**3
        c1 += xi**2
        c0 += xi
        c_ind += xi * yi

    return (c3, c2, c1, c0, c_ind)

def der_parcial_2(xs: list, ys: list) -> tuple[float, float, float, float, float
    c3 = c2 = c1 = c0 = c_ind = 0.0

    for xi, yi in zip(xs, ys):
        c3 += xi**5
        c2 += xi**4
        c1 += xi**3
        c0 += xi**2
        c_ind += (xi**2) * yi

    return (c3, c2, c1, c0, c_ind)

def der_parcial_3(xs: list, ys: list) -> tuple[float, float, float, float, float
    c3 = c2 = c1 = c0 = c_ind = 0.0

    for xi, yi in zip(xs, ys):
        c3 += xi**6
        c2 += xi**5
        c1 += xi**4
        c0 += xi**3
        c_ind += (xi**3) * yi

    return (c3, c2, c1, c0, c_ind)
```

In [10]:
```
ajustar_min_cuadrados([3.38, 0.35, 2.07, -0.45, -0.56, -1.06, -2.78, 3.08, -3.99
 -3.18, -1.97, 2.37, -1.67, 4.09, -4.60, 2.68, 2.78, -3.79, -1.87]
, [15.65, -11.20, -4.00, 18.03, 7.94, 15.32, -4.40, 1.39, -11.92, -90.24,
 6.92, 28.35, -2.41, -5.47, 35.91, -55.53, 0.77, 4.79, -27.05, 2.85]
, [der_parcial_3, der_parcial_2, der_parcial_1, der_parcial_0])
```

```
[12-15 21:25:31][INFO] Se ajustarán 4 parámetros.
```

Out[10]:
```
array([ 1.04497792,  0.03132741, -8.88066397,  2.76228992])
```

In [11]:
```
xs = [3.38, 0.35, 2.07, -0.45, -0.56, -1.06, -2.78, 3.08, -3.99, -5.00,
 -3.18, -1.97, 2.37, -1.67, 4.09, -4.60, 2.68, 2.78, -3.79, -1.87]


ys = [15.65, -11.20, -4.00, 18.03, 7.94, 15.32, -4.40, 1.39, -11.92, -90.24,
 6.92, 28.35, -2.41, -5.47, 35.91, -55.53, 0.77, 4.79, -27.05, 2.85]


a, b, c, d = ajustar_min_cuadrados(xs, ys, [der_parcial_3, der_parcial_2, der_pa
xs0 = np.linspace(-10, 13, 200)
```

```
def s_0(x):
    return a * x**3 + b * x**2 + c * x + d


plt.figure(figsize=(8, 5))
for xi, yi in zip(xs, ys):
    plt.scatter([xi], [yi], color='red')



plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')

plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Mínimos cuadrados')
plt.legend()
plt.xlim(-10, 14)
plt.ylim(-80, 40)
plt.show()
```

```
[12-15 21:25:31][INFO] Se ajustarán 4 parámetros.
```