Tarea N°7

Nombre: Rommel Rivera

1) Spline Cúbico Natural para puntos (0,1), (1,5), (2,3)

$$S_0(x)=1+5.5x-1.5x^3$$

$$S_1(x)=5+1(x-1)-4.5(x-1)^2+1.5(x-1)^3$$

2) Spline Cúbico Sujeto (Clamped) para (-1,1), (1,3) con derivadas

$$S(x)=1+1(x+1)-0.5(x+1)^2+0.25(x+1)^3$$

3) Completar la función cubic_spline (Python)

```
import sympy as sym

def cubic_spline(xs: list[float], ys: list[float]) ->
list[sym.Symbol]:
    """
    Cubic spline interpolation 'S'. Natural Boundary.
    """
    # Ordenar puntos
    points = sorted(zip(xs, ys), key=lambda x: x[0])
    xs = [x for x, _ in points]
    ys = [y for _, y in points]

    n = len(points) - 1
    h = [xs[i+1] - xs[i] for i in range(n)]

    # 1. Calcular alpha (lado derecho del sistema)
    alpha = [0.0] * (n + 1)
    for i in range(1, n):
        alpha[i] = (3 / h[i]) * (ys[i+1] - ys[i]) - (3 / h[i-1]) *
(ys[i] - ys[i-1])

    # 2. Resolver sistema tridiagonal (Algoritmo de
Crout/Factorización LU)
    l = [0.0] * (n + 1)
    mu = [0.0] * (n + 1)
    z = [0.0] * (n + 1)

    l[0] = 1.0
    mu[0] = 0.0
    z[0] = 0.0

    for i in range(1, n):
        l[i] = 2 * (xs[i+1] - xs[i-1]) - h[i-1] * mu[i-1]
        mu[i] = h[i] / l[i]
        z[i] = (alpha[i] - h[i-1] * z[i-1]) / l[i]

    l[n] = 1.0
    z[n] = 0.0

    # 3. Calcular coeficientes c, b, d (Back substitution)
    c = [0.0] * (n + 1)
    b = [0.0] * n
    d = [0.0] * n
    a = ys[:] # a_i es simplemente y_i

    c[n] = 0.0

    splines = []
    x = sym.Symbol('x')

    for j in range(n - 1, -1, -1):
        c[j] = z[j] - mu[j] * c[j+1]
        b[j] = (a[j+1] - a[j]) / h[j] - h[j] * (c[j+1] + 2 * c[j]) / 3
        d[j] = (c[j+1] - c[j]) / (3 * h[j])


        expr = a[j] + b[j]*(x - xs[j]) + c[j]*(x - xs[j])**2 + d[j]*(x
- xs[j])**3
        splines.append(expr)

    return splines[::-1]
```

4) Spline Cúbico para xs = [1, 2, 3], ys = [2, 3, 5]

$$S_0(x) = 2 + 0.75(x-1) + 0.25(x-1)^3$$

$$S_1(x) = 3 + 1.5(x-2) + 0.75(x-2)^2 - 0.25(x-2)^3$$

5) Spline Cúbico para xs = [0, 1, 2, 3], ys = [-1, 1, 5, 2]

$$S_0(x) = -1 + 1x + 1x^3$$

$$S_1(x) = 1 + 4(x-1) + 3(x-1)^2 - 3(x-1)^3$$

$$S_2(x) = 5 + 1(x-2) - 6(x-2)^2 + 2(x-2)^3$$

6) Gráfica usando cubic_spline_clamped Nota importante: No has proporcionado la "tabla" mencionada en la pregunta. Sin embargo, asumiendo que el objetivo es graficar unos datos genéricos o los del ejercicio 5 usando matplotlib, aquí tienes cómo se haría el código paso a paso.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline


x_data = [0, 1, 2, 3]
y_data = [-1, 1, 5, 2]

def graficar_spline(x_puntos, y_puntos):

    x_smooth = np.linspace(min(x_puntos), max(x_puntos), 100)

    cs = CubicSpline(x_puntos, y_puntos, bc_type='natural')

    plt.figure(figsize=(8, 6))

    plt.plot(x_puntos, y_puntos, 'o', label='Datos', color='red')

    plt.plot(x_smooth, cs(x_smooth), label='Spline Cúbico',
color='blue')

    plt.title('Interpolación Spline Cúbico')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.grid(True)
    plt.show()

graficar_spline(x_data, y_data)
```

link del repositorio: https://github.com/RommelRam/Metodos-Numericos