ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
MÉTODOS NUMÉRICOS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

Nombre: Sebastián Alexander Morales Cedeño

Curso: GR1CC

Fecha: 01/06/2025

[Tarea 07] Unidad 03-B | splines cúbicos

Repositorio:

https://github.com/SebastianMoralesEpn/Github1.0/tree/979366d2db980e07ffe443a23c69d2eaedf8d7d3/Tareas/%5BTarea%2007%5D%20Unidad%2003-B%20%20splines%20c%C3%BAbicos

CONJUNTO DE EJERCICIOS

1. **Dados los puntos (0,1), (1,5), (2,3), determine el spline cúbico.**

Ecuaciones iniciales:

$$s_0(x) = a_0 + b_0 x + c_0 x^2 + d_0 x^3$$
$$s_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3$$

Derivadas:

$$s_0{}'(x) = b_0 + 2c_0 x + 3d_0 x^2$$
$$s_0''(x) = 2c_0 + 6d_0 x$$

$$s_1'(x) = b_1 - 2c_1 + 2c_1 x + 3d_1 x^2 - 6d_1 x + 3d_1$$
$$s_1''(x) = 2c_1 + 6d_1 x - 6d_1$$

Ecuaciones:

1. $s_0(0) = 1$

$$a_0 = 1$$

2. $s_0(1) = 5$

$$a_0 + b_0 + c_0 + d_0 = 5$$

3. $s_1(1) = 5$

$$a_1 = 5$$

4. $s_1(2) = 3$

$$a_1 + b_1 + c_1 + d_1 = 3$$

5. $s_0'(1) = s_1{}'(1)$

$$b_0 + 2c_0 + 3d_0 = b_1$$

6. $s_0''(1) = s_1{}''(1)$

$$2c_0 + 6d_0 = 2c_1 \rightarrow 3d_0 = c_1$$

7. $s_0''(0) = 0$ Frontera natural

$$2c_0 = 0 \rightarrow c_0 = 0$$

8. $s_1''(2) = 0$ Frontera natural

$$2c_1 + 6d_1 = 0$$

Resolviendo:

- $4 - d_0 + 3d_0 + 3d_0 - d_0 = -2 \rightarrow d_0 = -1.5$
- $b_0 - 1.5 = 4 \rightarrow b_0 = 5.5$
- $5.5 - 4.5 = b_1 \rightarrow b_1 = 1$
- $c_1 = -4.5$
- $d_1 = 1.5$

Ecuaciones de los splines:

$$s_0(x) = 1 + 5.5x - 1.5x^3$$

$$s_1(x) = 5 + (x - 1) - 4.5(x - 1)^2 + 1.5(x - 1)^3$$

2. **Dados los puntos (-1,1), (1,3), determine el spline cubico sabiendo que $\mathbf{f'(x_0) = 1}$, $\mathbf{f'(x_n) = 2}$.**

Ecuación inicial:

$$s_0(x) = a_0 + b_0(x + 1) + c_0(x + 1)^2 + d_0(x + 1)^3$$

Derivada:

$$s_0'(x) = b_0 + 2c_0x + 2c_0 + 3d_0x^2 + 6d_0x + 3d_0$$

Ecuaciones:

1. $s_0(-1) = 1$

$$a_0 = 1$$

2. $s_0(1) = 3$

$$a_0 + 2b_0 + 4c_0 + 8d_0 = 3$$

3. $s_0'(-1) = 1$

$$b_0 - 2c_0 + 2c_0 + 3d_0 - 6d_0 + 3d_0 = 1 \rightarrow b_0 = 1$$

4. $s_0'(1) = 2$

$$b_0 + 4c_0 + 12d_0 = 2$$

Resolviendo:

- $1 + 2 + 4c_0 + 8d_0 = 3 \rightarrow c_0 = -2d_0$
- $1 - 8d_0 + 12d_0 = 2 \rightarrow d_0 = 0.25$
- $c_0 = -0.5$

Ecuación de los splines:

$$s_0(x) = 1 + (x + 1) - 0.5(x + 1)^2 + 0.25(x + 1)^3$$

3. **Diríjase al pseudocódigo del spline cúbico con frontera natural provisto en clase, en base a ese pseudocódigo complete la siguiente función:**

```python
##############################################
def cubic_spline(xs: list[float], ys: list[float]) -> list[sym.Symbol]:
    """
    Cubic spline interpolation ``S``. Every two points are interpolated by a cubic polynomial
    ``S_j`` of the form ``S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3.``

    xs must be different  but not necessarily ordered nor equally spaced.

    ## Parameters
    - xs, ys: points to be interpolated

    ## Return
    - List of symbolic expressions for the cubic spline interpolation.
    """

    points = sorted(zip(xs, ys), key=lambda x: x[0])  # sort points by x

    xs = [x for x, _ in points]
    ys = [y for _, y in points]

    n = len(points) - 1  # number of splines

    h = [xs[i + 1] - xs[i] for i in range(n)]  # distances between  contiguous xs

    # alpha = # completar
    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])

    l = [1]
```

https://github.com/ztjona/EPN-numerical-analysis/blob/main/cubic_splines.ipynb

Código completo:

```python
import sympy as sp
from IPython.display import display

def spline_cubica(valores_x: list[float], valores_y: list[float]) ->
list[sp.Expr]:
    """
    Interpolación cúbica por tramos. Cada par de puntos se conecta
mediante un polinomio cúbico
    de la forma:
        P_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3

    Los valores de entrada no requieren estar ordenados ni ser
equiespaciados.

    ## Parámetros
    - valores_x: coordenadas en x de los puntos
    - valores_y: coordenadas en y de los puntos

    ## Retorna
```

```
    - Lista de expresiones simbólicas que representan los polinomios
cúbicos por tramos.
    """

    # Ordenar los puntos por sus coordenadas x
    puntos_ordenados = sorted(zip(valores_x, valores_y), key=lambda par:
par[0])
    x_data = [p[0] for p in puntos_ordenados]
    y_data = [p[1] for p in puntos_ordenados]

    num_intervalos = len(x_data) - 1
    h_deltas = [x_data[i + 1] - x_data[i] for i in range(num_intervalos)]

    # Cálculo de la secuencia alpha
    alpha = [0] * num_intervalos
    for i in range(1, num_intervalos):
        alpha[i] = 3 / h_deltas[i] * (y_data[i + 1] - y_data[i]) - 3 /
h_deltas[i - 1] * (y_data[i] - y_data[i - 1])

    # Inicialización de los arreglos l, u, z
    l = [1]
    u = [0]
    z = [0]

    for i in range(1, num_intervalos):
        l.append(2 * (x_data[i + 1] - x_data[i - 1]) - h_deltas[i - 1] *
u[i - 1])
        u.append(h_deltas[i] / l[i])
        z.append((alpha[i] - h_deltas[i - 1] * z[i - 1]) / l[i])

    l.append(1)
    z.append(0)
    c_coef = [0] * (num_intervalos + 1)

    x = sp.Symbol("x")
    polinomios = []

    for j in range(num_intervalos - 1, -1, -1):
        c_coef[j] = z[j] - u[j] * c_coef[j + 1]
        b_coef = (y_data[j + 1] - y_data[j]) / h_deltas[j] - h_deltas[j]
* (c_coef[j + 1] + 2 * c_coef[j]) / 3
        d_coef = (c_coef[j + 1] - c_coef[j]) / (3 * h_deltas[j])
        a_coef = y_data[j]
```

```
        print(f"Intervalo {j}: a={a_coef}, b={b_coef}, c={c_coef[j]}, d={d_coef}")

        polinomio = a_coef + b_coef * (x - x_data[j]) + c_coef[j] * (x - x_data[j])**2 + d_coef * (x - x_data[j])**3
        polinomios.append(polinomio)

    polinomios.reverse()
    return polinomios
```

4. Usando la función anterior, encuentre el spline cúbico para:

$$xs = [1,2,3]$$
$$ys = [2,3,5]$$

Resultado:

```
Intervalo 1: a=3, b=1.5, c=0.75, d=-0.25
Intervalo 0: a=2, b=0.75, c=0.0, d=0.25
```

$$0.75x + 0.25\left(x-1\right)^3 + 1.25$$

$$1.5x - 0.25\left(x-2\right)^3 + 0.75\left(x-2\right)^2$$

______

$$0.25x^3 - 0.75x^2 + 1.5x + 1.0$$

$$-0.25x^3 + 2.25x^2 - 4.5x + 5.0$$

5. Usando la función anterior, encuentre el spline cubico para:

$$xs = [0,1,2,3]$$
$$ys = [-1,1,5,2]$$

Resultado:

```
Intervalo 2: a=5, b=1.0, c=-6.0, d=2.0
Intervalo 1: a=1, b=4.0, c=3.0, d=-3.0
Intervalo 0: a=-1, b=1.0, c=0.0, d=1.0
```

$$1.0x^3 + 1.0x - 1$$

$$4.0x - 3.0(x-1)^3 + 3.0(x-1)^2 - 3.0$$

$$1.0x + 2.0(x-2)^3 - 6.0(x-2)^2 + 3.0$$

______

$$1.0x^3 + 1.0x - 1$$

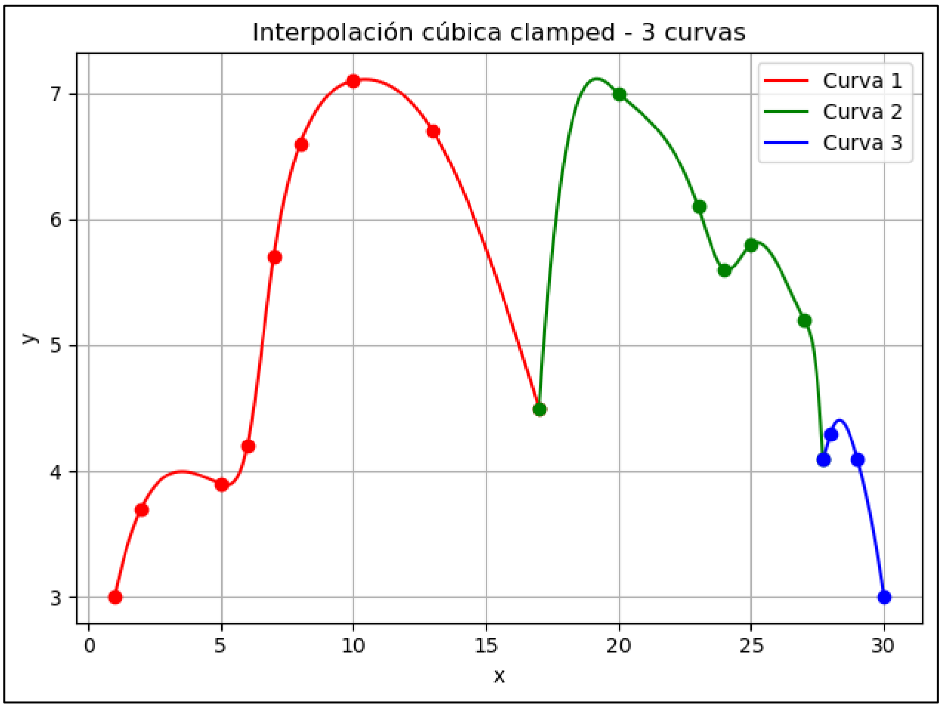
$$-3.0x^3 + 12.0x^2 - 11.0x + 3.0$$

$$2.0x^3 - 18.0x^2 + 49.0x - 37.0$$

6. **Use la función cubic_spline_clamped, provista en el enlace de Github, para graficar los datos de la siguiente tabla.**

Curva 1				Curva 2				Curva 3			
i	x_i	$f(x_i)$	$f'(x_i)$	i	x_i	$f(x_i)$	$f'(x_i)$	i	x_i	$f(x_i)$	$f'(x_i)$
0	1	3.0	1.0	0	17	4.5	3.0	0	27.7	4.1	0.33
1	2	3.7		1	20	7.0		1	28	4.3	
2	5	3.9		2	23	6.1		2	29	4.1	
3	6	4.2		3	24	5.6		3	30	3.0	−1.5
4	7	5.7		4	25	5.8					
5	8	6.6		5	27	5.2					
6	10	7.1		6	27.7	4.1	−4.0				
7	13	6.7									
8	17	4.5	−0.67								

Gráfica:



Conclusiones de la gráfica:

- Las tres curvas siguen un patrón similar, lo cual indica que los conjuntos de datos están relacionados y representan fenómenos con una evolución parecida a lo largo del dominio.

- Aunque las curvas comparten estructura, se evidencian pequeñas variaciones en la altura (valor de y) en ciertos intervalos, especialmente después del punto x = 10, lo que sugiere diferencias sutiles en los datos medidos o simulados.

- La curva 1 muestra valores ligeramente más altos en el intervalo x = 6 a x = 10, lo que podría interpretarse como una mayor respuesta del sistema representado o una variación experimental positiva respecto a las otras dos curvas.

- Hacia el final del intervalo (entre x = 13 y x = 17), la curva 3 desciende más abruptamente que las demás, lo cual puede indicar una caída más pronunciada en la variable representada.

- La curva 2 parece mantenerse entre la Curva 1 y la Curva 3 tanto en las fases de crecimiento como en el descenso, lo cual la convierte en una buena referencia de promedio o tendencia central.