ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
MÉTODOS NUMÉRICOS
INGENIERÍA DE CIENCIAS DE LA COMPUTACIÓN

Nombre: Moisés Pineda

Fecha: 01/06/2025

Curso: GR1CC

Docente: Jonathan A. Zea

Repositorio: Metodos_Numericos_GRCC1/Tareas/[Tarea 07] Unidad 03-B splines cúbicos at main · SantiagoTmg/Metodos_Numericos_GRCC1

CONJUNTO DE EJERCICIOS

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

   Ecuaciones iniciales:

$$s_0(x) = a_0 + b_0x + c_0x^2 + d_0x^3$$
$$s_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3$$

   Derivadas:

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

$$s_1{}'(x) = b_1 - 2c_1 + 2c_1x + 3d_1x^2 - 6d_1x + 3d_1$$
$$s_1{}''(x) = 2c_1 + 6d_1x - 6d_1$$

   Ecuaciones para la resolución:

   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:

   2. $b_0 + d_0 = 4$
   4. $b_1 + c_1 + d_1 = -2$
   5. $b_0 + 3d_0 = b_1$
   8. $6d_0 + 6d_1 = 0 \rightarrow d_0 = -d_1$

Remplazando:

4. $4 - d_0 + 3d_0 + 3d_0 - d_0 = -2 \rightarrow d_0 = -1.5$
2. $b_0 - 1.5 = 4 \rightarrow b_0 = 5.5$
5. $5.5 - 4.5 = b_1 \rightarrow b_1 = 1$
6. $c_1 = -4.5$
8. $d_1 = 1.5$

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 $f'(x_0) = 1$, $f'(x_n) = 2$.**

Ecuacion 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 para la resolución:

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:

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

Remplazando:

2. $c_0 = -0.5$

Ecuaciones de los *splines*:

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

3. **Diríjase al pseudocódigo *spline* cubico 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]
```

Función completa:

```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 = [0] * n
    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]
    u = [0]
    z = [0]

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


    l.append(1)
    z.append(0)
    c = [0] * (n + 1)

    x = sym.Symbol("x")
    splines = []
    for j in range(n - 1, -1, -1):
        c[j] = z[j] - u[j] * c[j + 1]
        b = (ys[j + 1] - ys[j]) / h[j] - h[j] * (c[j + 1] + 2 * c[j]) / 3
        d = (c[j + 1] - c[j]) / (3 * h[j])
        a = ys[j]
        print(j, a, b, c[j], d)
        S  = a + b * (x-xs[j]) + c[j] * (x-xs[j])**2 + d * (x-xs[j])**3

        splines.append(S)
    splines.reverse()
    return splines
```

4. **Usando la función anterior, encuentre el *spline* cubico para:**
   xs = [1, 2, 3]
   ys = [2, 3, 5]

   Respuesta:

```
1 3 1.5 0.75 -0.25
0 2 0.75 0.0 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]

Respuesta:

```
2 5 1.0 -6.0 2.0
1 1 4.0 3.0 -3.0
0 -1 1.0 0.0 1.0
```

$1.0x^3 + 1.0x - 1$

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

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

______

$1.0x^3 + 1.0x - 1$

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

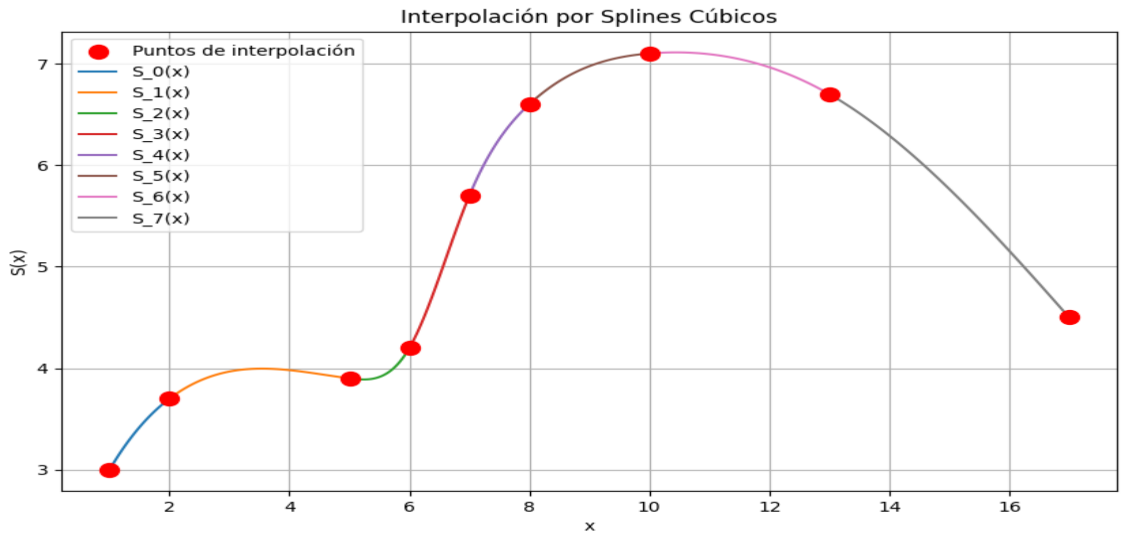
$2.0x^3 - 18.0x^2 + 49.0x - 37.0$

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

Curva 1

i	x_i	$f(x_i)$	$f'(x_i)$
0	1	3.0	1.0
1	2	3.7	
2	5	3.9	
3	6	4.2	
4	7	5.7	
5	8	6.6	
6	10	7.1	
7	13	6.7	
8	17	4.5	-0.67

Grafica:



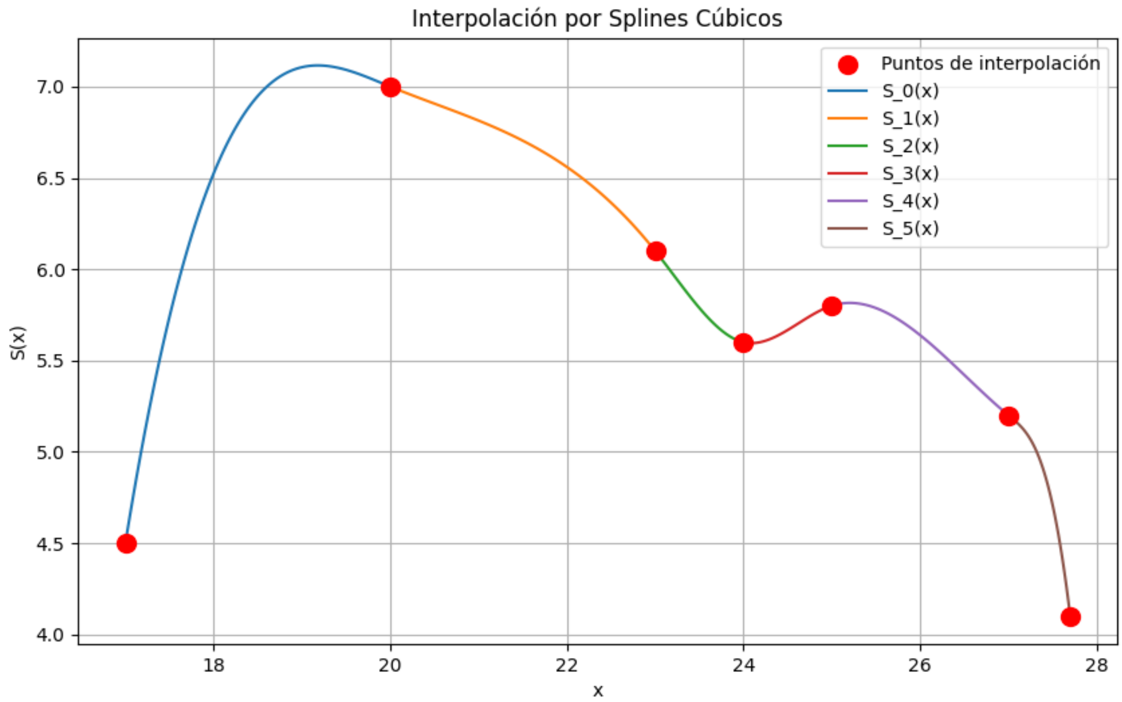
Ecuaciones de los splines:

$$S_0(x) = 0.0468099653460708x^3 - 0.487239861384283x^2 + 1.83404982673035x + 1.60638006930786$$
$$S_1(x) = 0.0265552121382411x^3 - 0.365711342137305x^2 + 1.5909927882364x + 1.7684180949705$$
$$S_2(x) = 0.341862882832256x^3 - 5.09532640254753x^2 + 25.2390680902875x - 37.6450407417814$$
$$S_3(x) = -0.574548094033905x^3 + 11.4000711810434x^2 - 73.7333174112578x + 160.299730261309$$
$$S_4(x) = 0.156329493303363x^3 - 3.94835815303925x^2 + 33.7056879273205x - 90.3912821953733$$
$$S_5(x) = 0.0239201086447503x^3 - 0.770532921232554x^2 + 8.28308607286689x - 22.5976772501638$$
$$S_6(x) = -0.00255606547823463x^3 + 0.023752302456995x^2 + 0.340233835971401x + 3.87849687282113$$
$$S_7(x) = 0.00574178139926946x^3 - 0.299863725765665x^2 + 4.54724220286598x - 14.3518727170554$$

Curva 2

i	x_i	$f(x_i)$	$f'(x_i)$
0	17	4.5	3.0
1	20	7.0	
2	23	6.1	
3	24	5.6	
4	25	5.8	
5	27	5.2	
6	27.7	4.1	-4.0

Grafica:



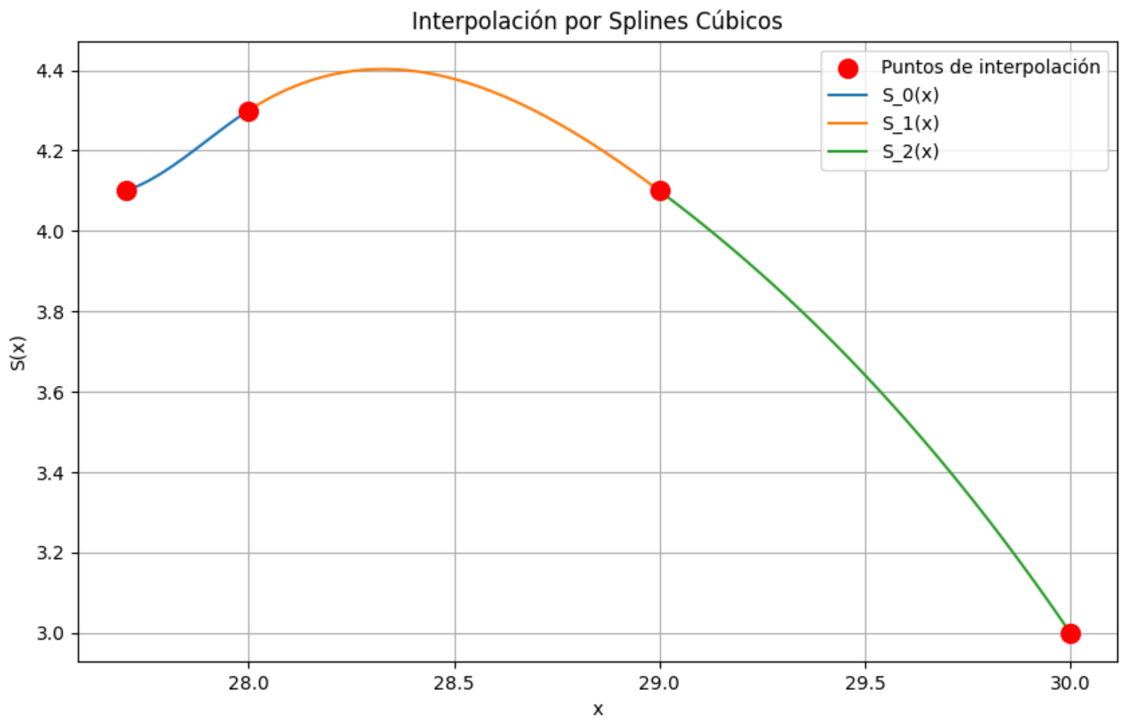
Ecuaciones de los splines:

$$S_0(x) = 0.12616207628025x^3 - 7.53497434135573x^2 + 149.806607471118x - 984.439023122068$$
$$S_1(x) = -0.022930673285195x^3 + 1.41059063257098x^2 - 29.1046920074162x + 208.302973401493$$
$$S_2(x) = 0.280127236863149x^3 - 19.5004051676648x^2 + 451.848211398006x - 3479.00261937341$$
$$S_3(x) = -0.357384536100794x^3 + 26.4004424857391x^2 - 649.772132283688x + 5333.96013008014$$
$$S_4(x) = 0.0882021573401092x^3 - 7.0185595223286x^2 + 185.702917918006x - 1628.33195493397$$
$$S_5(x) = -2.56800212665878x^3 + 208.133987481581x^2 - 5623.41585118756x + 50653.7369670161$$

Curva 3

i	x_i	$f(x_i)$	$f'(x_i)$
0	27.7	4.1	0.33
1	28	4.3	
2	29	4.1	
3	30	3.0	-1.5

Grafica:



Ecuaciones de los splines:

$$S_0(x) = -3.79941327466078x^3 + 317.993289328931x^2 - 8870.74279427938x + 82483.079611294$$
$$S_1(x) = 0.296039603960395x^3 - 26.0247524752475x^2 + 761.762376237622x - 7420.30198019801$$

$$S_2(x) = -0.0653465346534656x^3 + 5.41584158415843x^2 - 150.014851485149x + 1393.54455445545$$