TAREA N° 7

Nombre:

Joel Stalin Tinitana Carrion

Fecha:

02/06/2025

Tema:

Splines cúbicos

Splines Cúbicos

Los *splines cúbicos* son funciones formadas por la unión de varios polinomios de tercer grado que se utilizan para aproximar de forma suave una curva a partir de un conjunto de puntos. Su principal ventaja es que ofrecen una interpolación continua y derivable, evitando las oscilaciones que aparecen en otros métodos como el polinomio de Lagrange de alto grado.

Cada segmento del spline en el intervalo $[x_j, x_{j+1})$ se define como:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Condiciones de continuidad

Para que los polinomios formen un spline cúbico válido, deben cumplirse las siguientes condiciones:

- **Interpolación exacta**:

$$S_j(x_j) = y_j, S_j(x_{j+1}) = y_{j+1}$$

- **Continuidad de la derivada primera**:

$$S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$$

- **Continuidad de la derivada segunda**:

$$S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$$

Condiciones de frontera

Para resolver el sistema de ecuaciones que permite hallar los coeficientes a_j, b_j, c_j, d_j, se aplican condiciones de borde:

- **Frontera natural**: la segunda derivada en los extremos es cero:

$$S''(x_0)=0, S''(x_n)=0$$

- **Frontera condicionada**: se especifican valores de la derivada primera:

$$S'(x_0)=B_0, S'(x_n)=B_n$$

Aplicación

Los splines cúbicos son ampliamente utilizados en:

- Interpolación numérica de datos.
- Diseño asistido por computadora (CAD).
- Animaciones y gráficos por computadora.
- Ajuste de curvas en ingeniería y ciencias aplicadas.

Este método proporciona una curva suave, continua y flexible, ideal para representar datos reales de forma precisa.

Conjunto de Ejercicios – Splines Cúbicos

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

Ejercicio 1 – Spline cúbico con frontera natural

Datos

Puntos dados:

$$x=[0,1,2), y=[1,5,3)$$

Queremos determinar el spline cúbico que interpola estos puntos, bajo condiciones de **frontera natural**:

$$S_0''(x_0)=0, S_1''(x_2)=0$$

El spline cúbico estará formado por dos tramos:

- (S_0(x)) definido en ([0, 1])
- (S_1(x)) definido en ([1, 2])

Cada uno con la forma general:

$$S_j(x)=a_j+b_j(x-x_j)+c_j(x-x_j)^2+d_j(x-x_j)^3$$

Ecuaciones por condiciones

1. Interpolación en los nodos

Para el primer tramo:

$$S_0(0)=y_0 \Rightarrow a_0=1$$

$$S_0(1)=a_0+b_0(1-0)+c_0(1-0)^2+d_0(1-0)^3=1+b_0+c_0+d_0=5$$

Entonces:

$$b_0+c_0+d_0=4$$

Para el segundo tramo:

$$S_1(1)=y_1 \Rightarrow a_1=5$$

$$S_1(2)=a_1+b_1(2-1)+c_1(2-1)^2+d_1(2-1)^3=5+b_1+c_1+d_1=3$$

Entonces:

$$b_1+c_1+d_1=-2$$

---

2. Continuidad de derivadas en (x = 1)

Primera derivada:

$$S_0'(1)=b_0+2c_0(1)+3d_0(1)^2=b_0+2c_0+3d_0$$

$$S_1'(1)=b_1$$

Entonces:

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

Segunda derivada:

$$S_0''(1)=2c_0+6d_0, S_1''(1)=2c_1$$

Entonces:

$$2c_0+6d_0=2c_1$$

---

3. Condiciones de frontera natural

$$S_0''(0)=2c_0=0 \Rightarrow c_0=0$$

$$S_1''(2) = 2c_1 + 6d_1 = 0 \Rightarrow d_1 = -\frac{c_1}{3}$$

Sustitución y resolución

De (5):

$$c_0 = 0$$

Entonces de (1):

$$b_0 + d_0 = 4 \Rightarrow b_0 = 4 - d_0$$

De (4):

$$6d_0 = 2c_1 \Rightarrow c_1 = 3d_0$$

De (3):

$$b_0 + 3d_0 = b_1$$

Sustituyendo (1') en (3'):

$$(4 - d_0) + 3d_0 = b_1 \Rightarrow b_1 = 4 + 2d_0$$

De (6):

$$d_1 = -\frac{c_1}{3} = -\frac{3d_0}{3} = -d_0$$

Sustituyendo todo en (2):

$$b_1 + c_1 + d_1 = -2 \\ (4 + 2d_0) + 3d_0 + (-d_0) = -2 \\ 4 + 4d_0 = -2 \Rightarrow d_0 = -1.5$$

Entonces:

$$b_0 = 4 - (-1.5) = 5.5 \\ c_0 = 0 \\ d_0 = -1.5 \\ c_1 = 3(-1.5) = -4.5 \\ d_1 = -(-1.5) = 1.5 \\ b_1 = 4 + 2(-1.5) = 1$$

Splines resultantes

Primer tramo:

$$0 \leq x \leq 1$$

$$S_0(x) = 1 + 5.5(x - 0) + 0(x - 0)^2 - 1.5(x - 0)^3$$

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

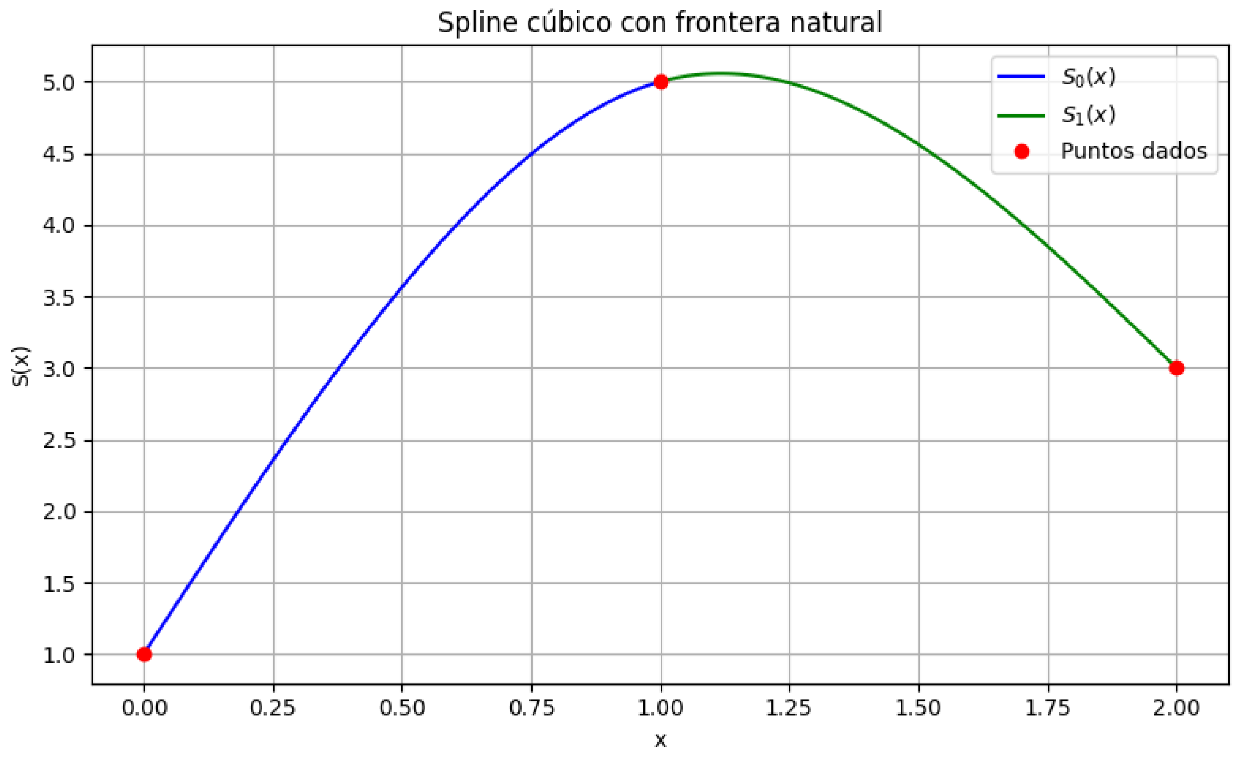
Segundo tramo:

$$1 < x \leq 2$$

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

Código

```python
import numpy as np
import matplotlib.pyplot as plt

# Coeficientes del spline por tramos
# S0(x) = 1 + 5.5x - 1.5x^3  para 0 <= x <= 1
# S1(x) = 5 + 1(x - 1) - 4.5(x - 1)^2 + 1.5(x - 1)^3  para 1 < x <= 2

# Definir funciones de los tramos
def S0(x):
    return 1 + 5.5 * x - 1.5 * x**3

def S1(x):
    return 5 + 1*(x - 1) - 4.5*(x - 1)**2 + 1.5*(x - 1)**3

# Rango de valores para cada tramo
x0 = np.linspace(0, 1, 100)
x1 = np.linspace(1, 2, 100)

# Evaluar funciones
y0 = S0(x0)
y1 = S1(x1)

# Puntos dados
x_data = np.array([0, 1, 2])
y_data = np.array([1, 5, 3])

# Graficar
plt.figure(figsize=(8, 5))
plt.plot(x0, y0, label=r"$S_0(x)$", color="blue")
plt.plot(x1, y1, label=r"$S_1(x)$", color="green")
plt.plot(x_data, y_data, 'ro', label="Puntos dados")
plt.title("Spline cúbico con frontera natural")
plt.xlabel("x")
plt.ylabel("S(x)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



1. Dados los puntos $(-1,1)$, $(1,3)$, determine el spline cúbico sabiendo que:

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

Ejercicio 2 – Spline cúbico con derivadas en los extremos (clamped)

Datos

Puntos dados:

$$x=[-1,1), y=[1,3)$$

Condiciones en las derivadas:

$$f'(-1)=1, f'(1)=2$$

Hay un solo tramo de spline cúbico entre (x = -1) y (x = 1), así que se define:

$$S(x)=a+b(x+1)+c(x+1)^2+d(x+1)^3$$

Se usa (x + 1) ya que el punto de inicio es (x_0 = -1)

Ecuaciones por condiciones

1. Interpolación en los extremos

Para (x = -1):

$$S(-1)=a=1$$

Para (x = 1):

$$ S(1) = a + b(2) + c(2)^2 + d(2)^3 = 3 \\ 1 + 2b + 4c + 8d = 3 \Rightarrow 2b + 4c + 8d = 2 \tag{2} $$

---

2. Derivadas en los extremos

Derivada general:

$$S'(x)=b+2c(x+1)+3d(x+1)^2$$

En (x = -1):

$$S'(-1)=b=1$$

En (x = 1):

$$S'(1)=b+4c+12d=2$$

Sustitución y resolución

De (1) y (3):

$$a=1,b=1$$

Sustituimos en (2):

$$ 2(1) + 4c + 8d = 2 \\ 4c + 8d = 0 \Rightarrow c + 2d = 0 \tag{5} $$

Sustituimos en (4):

$$1+4c+12d=2\Rightarrow 4c+12d=1$$

De (5):

$$c=-2d$$

Sustituyendo en (6):

$$ 4(-2d) + 12d = 1 \\ -8d + 12d = 1 \Rightarrow 4d = 1 \Rightarrow d = \frac{1}{4} $$

Entonces:

$$c = -2 \cdot \frac{1}{4} = -\frac{1}{2}$$

Spline resultante

$$S(x) = 1 + 1(x+1) - \frac{1}{2}(x+1)^2 + \frac{1}{4}(x+1)^3$$

(Opcional: forma simplificada)

$$S(x) = 1 + (x+1) - \frac{1}{2}(x+1)^2 + \frac{1}{4}(x+1)^3$$

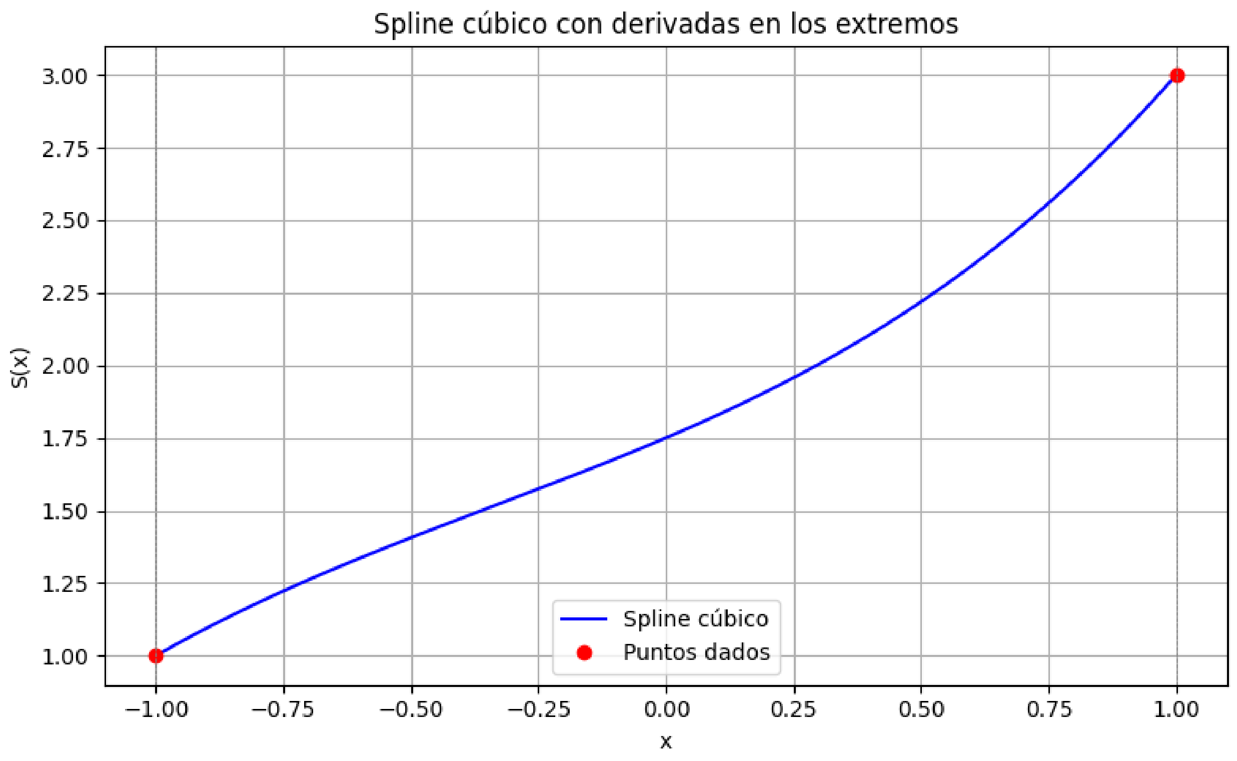
Dominio

$$-1 \leq x \leq 1$$

Código

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp

# Definir la variable simbólica
x = sp.Symbol('x')

# Datos del problema
x0, x1 = -1, 1
y0, y1 = 1, 3
fp0, fp1 = 1, 2  # f'(x0), f'(x1)

# Definir forma general del spline con base en (x + 1)
a, b, c, d = sp.symbols('a b c d')
S = a + b*(x + 1) + c*(x + 1)**2 + d*(x + 1)**3

# Derivada de S
S_deriv = sp.diff(S, x)

# Ecuaciones del sistema
eq1 = S.subs(x, -1) - y0  # S(-1) = 1
eq2 = S.subs(x, 1) - y1   # S(1) = 3
eq3 = S_deriv.subs(x, -1) - fp0  # S'(-1) = 1
eq4 = S_deriv.subs(x, 1) - fp1   # S'(1) = 2

# Resolver el sistema de ecuaciones
sol = sp.solve([eq1, eq2, eq3, eq4], (a, b, c, d))
```

```python
# Construir el spline con los valores encontrados
S_spline = S.subs(sol)
S_func = sp.lambdify(x, S_spline, modules='numpy')

# Generar puntos para graficar
x_vals = np.linspace(-1, 1, 200)
y_vals = S_func(x_vals)

# Puntos originales
x_data = [-1, 1]
y_data = [1, 3]

# Graficar
plt.figure(figsize=(8, 5))
plt.plot(x_vals, y_vals, label='Spline cúbico', color='blue')
plt.plot(x_data, y_data, 'ro', label='Puntos dados')
plt.title('Spline cúbico con derivadas en los extremos')
plt.xlabel('x')
plt.ylabel('S(x)')
plt.axvline(-1, color='gray', linestyle='--', linewidth=0.5)
plt.axvline(1, color='gray', linestyle='--', linewidth=0.5)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



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

Código de clases

import sympy as sym from IPython.display import display

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]
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  # = completar

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  # = completar
    print(j, a, b, c[j], d)
    S  # = completar

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

Código de clases completado

```
import sympy as sym
from IPython.display import display


# ####################################################################
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 += [(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
```

Ejemplo de utilización

```
xs = [0, 1, 2]
ys = [-5, -4, 3]

splines = cubic_spline(xs=xs, ys=ys)
_ = [display(s) for s in splines]
print("_______")
_ = [display(s.expand()) for s in splines]
```

```
# Graficar
xv, yv = evaluar_spline(xs, splines)
plt.plot(xv, yv, label="Spline", color='blue')
plt.plot(xs, ys, 'o', color='blue')
plt.title("Spline natural - Caso 1")
plt.grid(True)
plt.show()
```

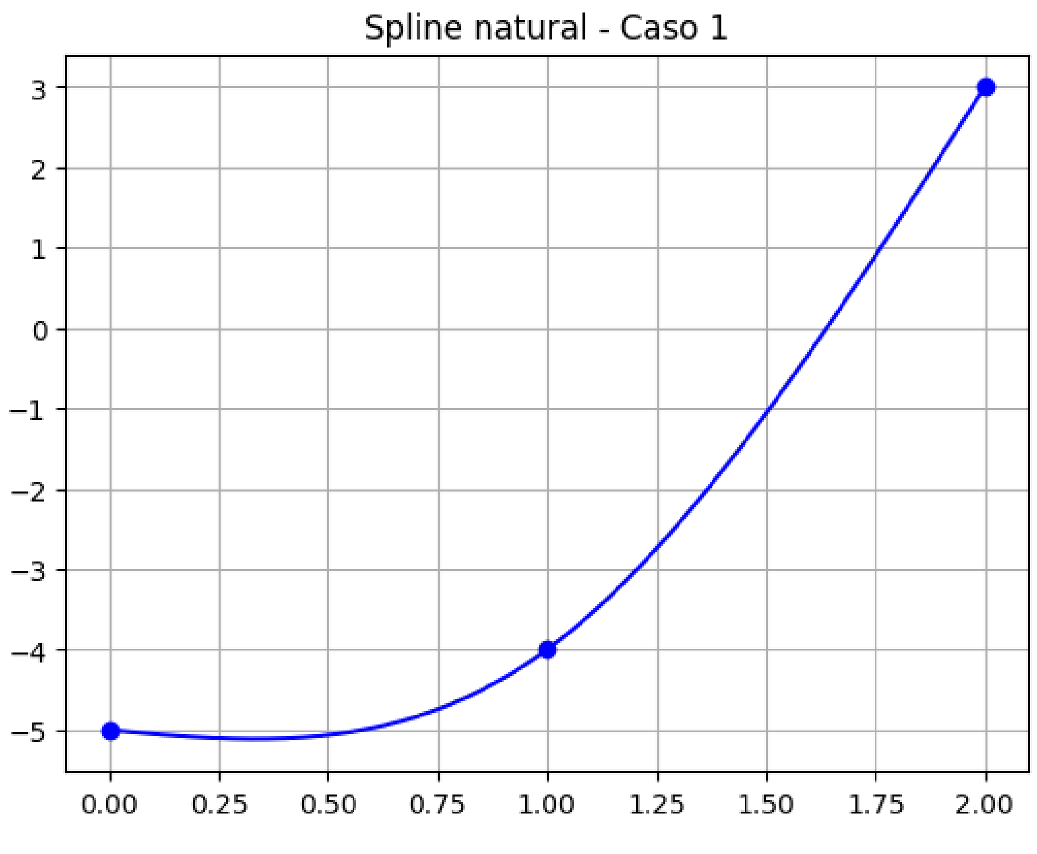
```
1 -4 4.0 4.5 -1.5
0 -5 -0.5 0.0 1.5
```

```
1.5*x**3 - 0.5*x - 5
```

```
4.0*x - 1.5*(x - 1)**3 + 4.5*(x - 1)**2 - 8.0
```

```
1.5*x**3 - 0.5*x - 5
```

```
-1.5*x**3 + 9.0*x**2 - 9.5*x - 2.0
```



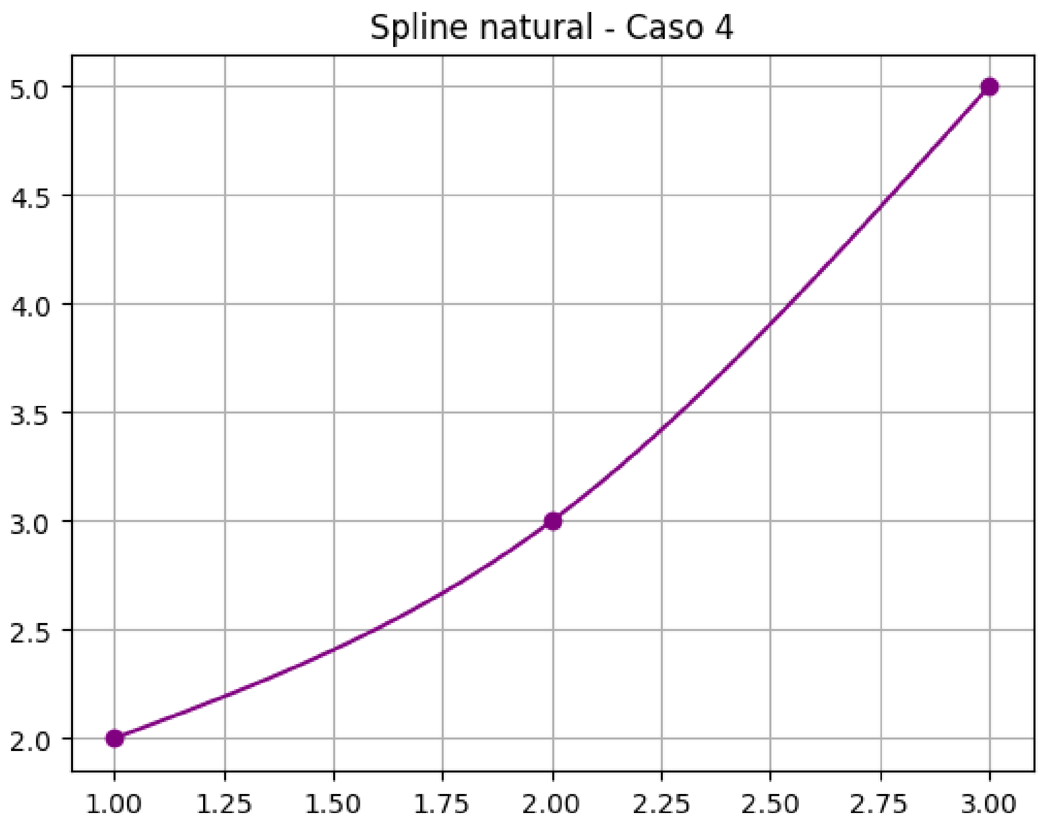
4) Usando la función anterior, encuentre el spline cúbico para:

```
xs = [1, 2, 3]

ys = [2, 3, 5]
```

```
  Cell In[27], line 1
    4) Usando la función anterior, encuentre el spline cúbico para:
     ^
SyntaxError: unmatched ')'
```

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

splines = cubic_spline(xs=xs, ys=ys)
_ = [display(s) for s in splines]
print("______")
_ = [display(s.expand()) for s in splines]

# Graficar
xv, yv = evaluar_spline(xs, splines)
plt.plot(xv, yv, label="Spline", color='purple')
plt.plot(xs, ys, 'o', color='purple')
plt.title("Spline natural - Caso 4")
plt.grid(True)
plt.show()
```

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

0.75*x + 0.25*(x - 1)**3 + 1.25

1.5*x - 0.25*(x - 2)**3 + 0.75*(x - 2)**2

______

0.25*x**3 - 0.75*x**2 + 1.5*x + 1.0

-0.25*x**3 + 2.25*x**2 - 4.5*x + 5.0
```



5) Usando la función anterior, encuentre el spline cúbico para:

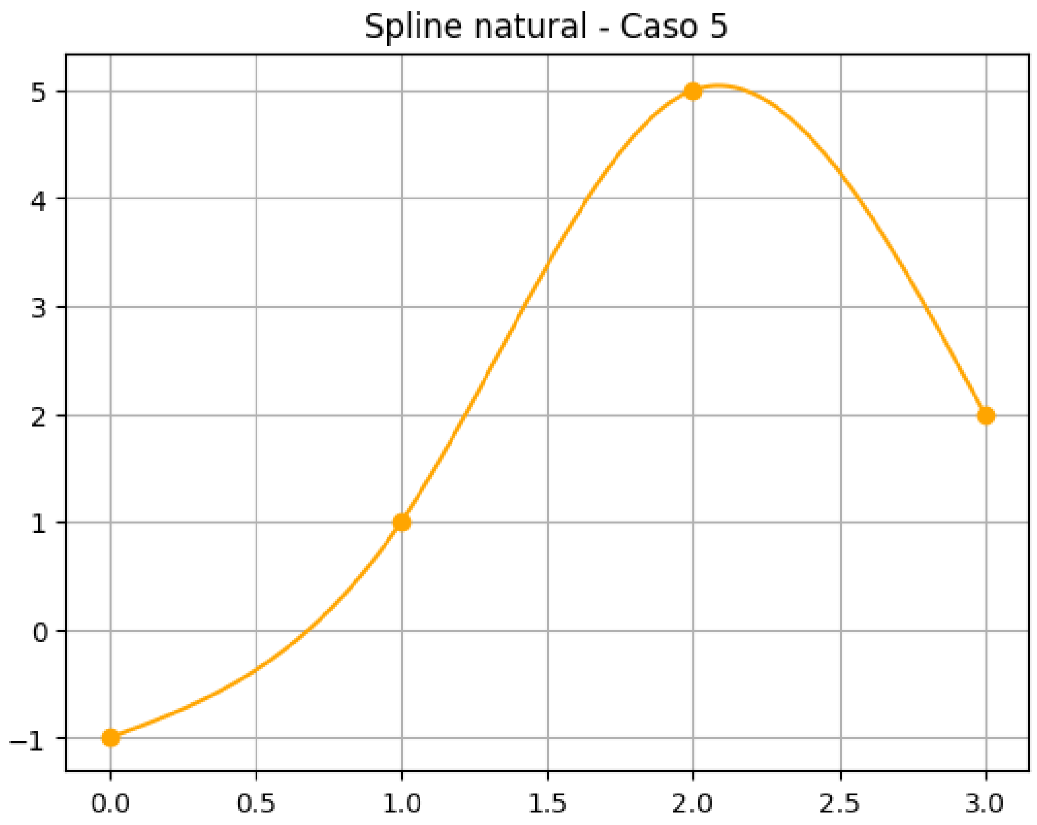
xs = [0, 1, 2, 3]

ys = [-1 ,1, 5, 2]

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

splines = cubic_spline(xs=xs, ys=ys)
_ = [display(s) for s in splines]
print("______")
_ = [display(s.expand()) for s in splines]

# Graficar
xv, yv = evaluar_spline(xs, splines)
plt.plot(xv, yv, label="Spline", color='orange')
plt.plot(xs, ys, 'o', color='orange')
plt.title("Spline natural - Caso 5")
plt.grid(True)
plt.show()
```

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

```
1.0*x**3 + 1.0*x - 1

4.0*x - 3.0*(x - 1)**3 + 3.0*(x - 1)**2 - 3.0

1.0*x + 2.0*(x - 2)**3 - 6.0*(x - 2)**2 + 3.0

______

1.0*x**3 + 1.0*x - 1

-3.0*x**3 + 12.0*x**2 - 11.0*x + 3.0

2.0*x**3 - 18.0*x**2 + 49.0*x - 37.0
```



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

Datos de entrada

Curva 1

i	x_i	$f(x_i)$	$f'(x_i)$
0	1.0	3.0	1.0
1	2.0	3.7	
2	5.0	3.9	

i	x_i	$f(x_i)$	$f'(x_i)$
3	6.0	4.2	
4	7.0	5.7	
5	8.0	6.6	
6	10.0	7.1	
7	13.0	6.7	
8	17.0	4.5	–0.67

Curva 2

i	x_i	$f(x_i)$	$f'(x_i)$
0	17.0	4.5	3.0
1	20.0	7.0	
2	23.0	6.1	
3	24.0	5.6	
4	25.0	5.8	
5	27.0	5.2	
6	27.7	4.1	–4.0

Curva 3

i	x_i	$f(x_i)$	$f'(x_i)$
0	27.7	4.1	0.33
1	28.0	4.3	
2	29.0	4.1	
3	30.0	3.0	–1.5

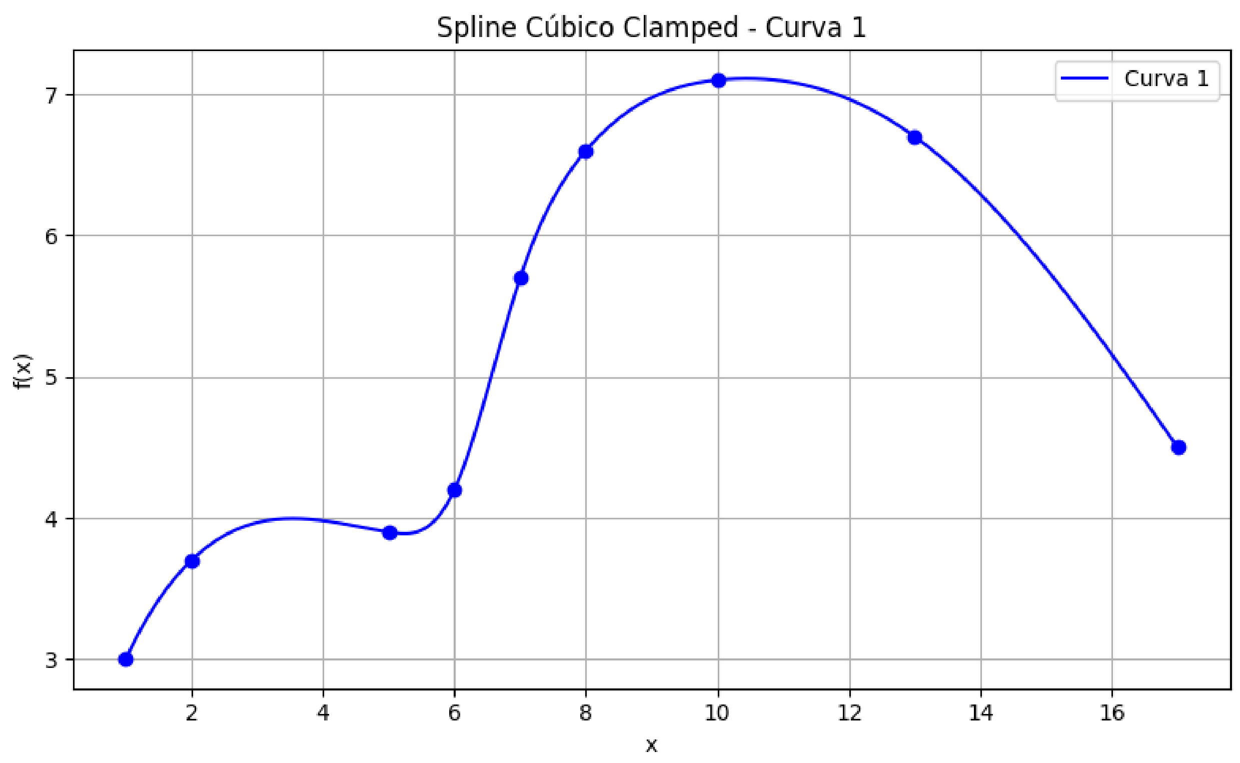
Código con la función

```python
import sympy as sym
import numpy as np
import matplotlib.pyplot as plt

# Función spline cúbico clamped con impresión de coeficientes
def cubic_spline_clamped(xs, ys, B0, B1):
    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 + 1)  # prealloc
    alpha[0] = 3 / h[0] * (ys[1] - ys[0]) - 3 * B0
    alpha[-1] = 3 * B1 - 3 / h[n - 1] * (ys[n] - ys[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])

    l = [2 * h[0]]
    u = [0.5]
    z = [alpha[0] / l[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 += [(alpha[i] - h[i - 1] * z[i - 1]) / l[i]]

    l.append(h[n - 1] * (2 - u[n - 1]))
    z.append((alpha[n] - h[n - 1] * z[n - 1]) / l[n])
    c = [0] * (n + 1)  # prealloc
    c[-1] = z[-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(f"Spline {j}: a={a}, b={b}, c={c[j]}, d={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

# Función para evaluar los splines
def evaluar_spline(xs, splines, puntos_por_tramo=100):
    x_vals = []
    y_vals = []
    x = sym.Symbol("x")
    for i in range(len(splines)):
        xi = np.linspace(xs[i], xs[i+1], puntos_por_tramo)
        yi = [float(splines[i].subs(x, val)) for val in xi]
        x_vals.extend(xi)
        y_vals.extend(yi)
    return x_vals, y_vals
```

Curva 1

```
x1 = [1.0, 2.0, 5.0, 6.0, 7.0, 8.0, 10.0, 13.0, 17.0]
y1 = [3.0, 3.7, 3.9, 4.2, 5.7, 6.6, 7.1, 6.7, 4.5]
B0_1 = 1.0
B1_1 = -0.67
splines1 = cubic_spline_clamped(x1, y1, B0_1, B1_1)
xv1, yv1 = evaluar_spline(x1, splines1)

plt.figure(figsize=(8, 5))
plt.plot(xv1, yv1, label='Curva 1', color='blue')
plt.plot(x1, y1, 'o', color='blue')
plt.title('Spline Cúbico Clamped - Curva 1')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
Spline 7: a=6.7, b=-0.3381314976116886, c=-0.07593425119415571,
d=0.0057417813992694635
Spline 6: a=7.1, b=0.04846024164091059, c=-0.052929661890044014, d=-
0.0025560654782346335
Spline 5: a=6.6, b=0.5472201929380908, c=-0.19645031375854607,
d=0.023920108644750342
Spline 4: a=5.7, b=1.4091093003652708, c=-0.665438793668634,
d=0.15632949330336265
Spline 3: a=4.2, b=1.0163426056008245, c=1.0582054884330803, d=-
0.5745480940339047
Spline 2: a=3.9, b=-0.07447972276856785, c=0.03261683993631198,
d=0.3418628828322561
Spline 1: a=3.7, b=0.4468099653460711, c=-0.20638006930785827,
d=0.02655521213824114
Spline 0: a=3.0, b=1.0, c=-0.3468099653460706, d=0.046809965346070785
```

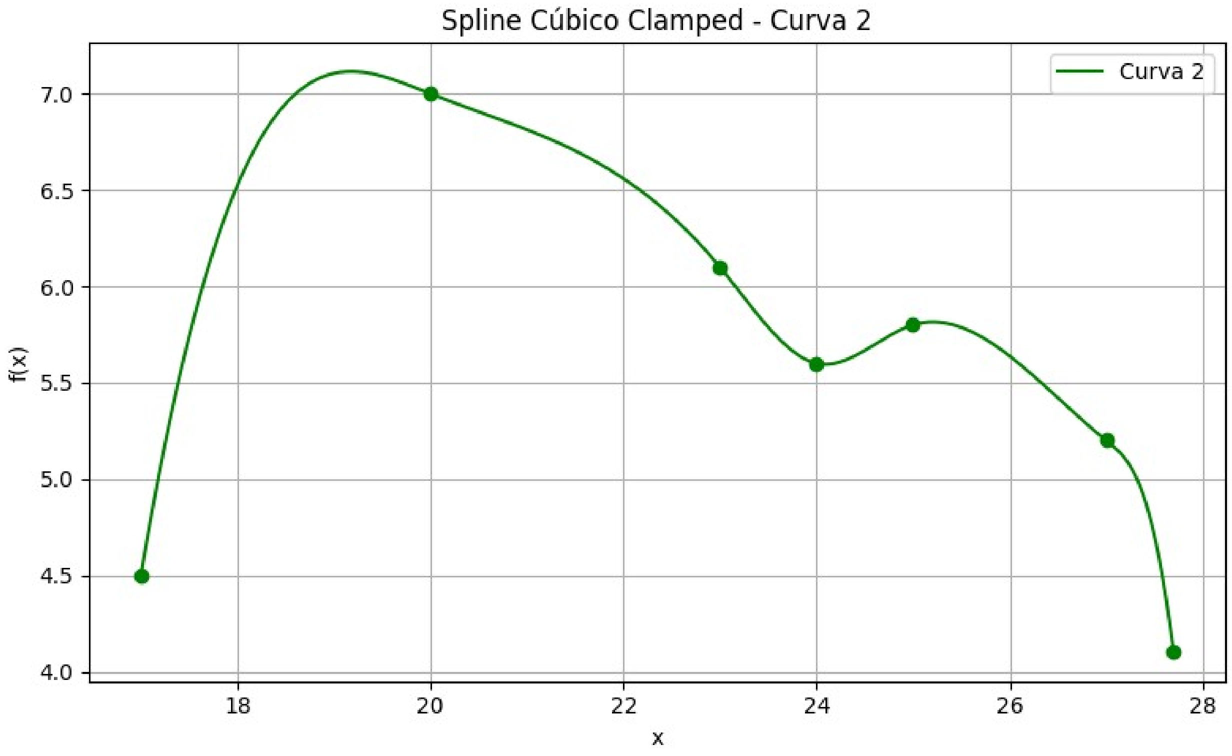


Curva 2

```
x2 = [17.0, 20.0, 23.0, 24.0, 25.0, 27.0, 27.7]
y2 = [4.5, 7.0, 6.1, 5.6, 5.8, 5.2, 4.1]
B0_2 = 3.0
B1_2 = -4.0
splines2 = cubic_spline_clamped(x2, y2, B0_2, B1_2)
xv2, yv2 = evaluar_spline(x2, splines2)

plt.figure(figsize=(8, 5))
plt.plot(xv2, yv2, label='Curva 2', color='green')
plt.plot(x2, y2, 'o', color='green')
plt.title('Spline Cúbico Clamped - Curva 2')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
Spline 5: a=5.2, b=-0.4011781849199465, c=0.1258152222202451, d=-
2.568002126658778
Spline 4: a=5.8, b=0.1539868142803838, c=-0.4033977218204103,
d=0.08820215734010924
Spline 3: a=5.6, b=-0.11137135038117751, c=0.6687558864819717, d=-
0.35738453610079396
```

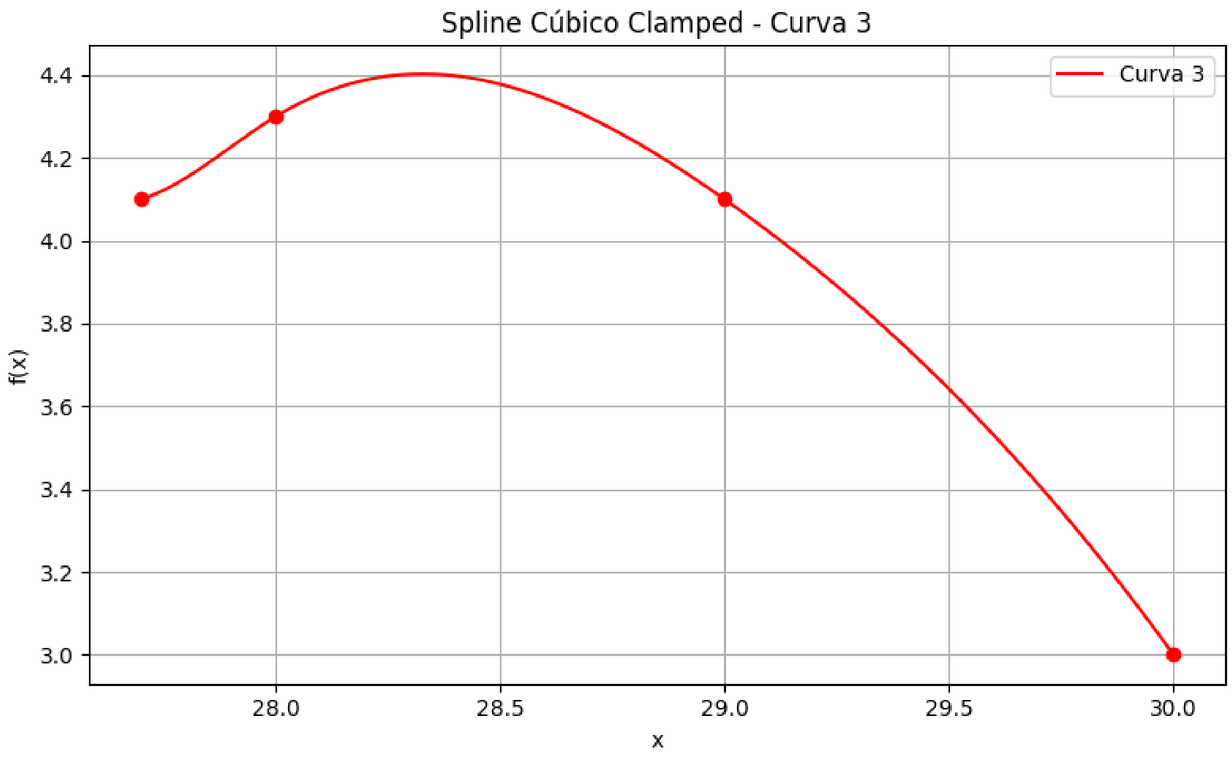
```
Spline 2: a=6.1, b=-0.6085014127556733, c=-0.17162582410747595,
d=0.2801272368631492
Spline 1: a=7.0, b=-0.19787464681108174, c=0.03475023545927881, d=-
0.022930673285194974
Spline 0: a=4.5, b=3.0, c=-1.1007084510629728, d=0.12616207628025017
```



Curva 3

```
x3 = [27.7, 28.0, 29.0, 30.0]
y3 = [4.1, 4.3, 4.1, 3.0]
B0_3 = 0.33
B1_3 = -1.5
splines3 = cubic_spline_clamped(x3, y3, B0_3, B1_3)
xv3, yv3 = evaluar_spline(x3, splines3)

plt.figure(figsize=(8, 5))
plt.plot(xv3, yv3, label='Curva 3', color='red')
plt.plot(x3, y3, 'o', color='red')
plt.title('Spline Cúbico Clamped - Curva 3')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
Spline 2: a=4.1, b=-0.7653465346534649, c=-0.26930693069306927, d=-
0.06534653465346556
Spline 1: a=4.3, b=0.6613861386138599, c=-1.1574257425742556,
d=0.2960396039603954
Spline 0: a=4.1, b=0.32999999999999, c=2.2620462046204524, d=-
3.799413274660778
```


Spline Cúbico Clamped - Curva 3
Curva 3
f(x)
x