Taller 04: splines cúbicos

Métodos Numéricos

Christopher Criollo

2025-11-24

Tabla de Contenidos

1 EJERCICIOS

1.1 Complete el código del siguiente repositorio:

```
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 polyn
    ``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 + 1)  # coefficients for the tridiagonal system
    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
```

1.2 Compruebe gráficamente la solución de los siguientes ejercicios:

1. $$(0,1),(1,5),(2,3)$$

```
import numpy as np

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

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

import matplotlib.pyplot as plt

x_vals = np.linspace(min(xs), max(xs), 200)
y_vals = np.zeros_like(x_vals)

for i, S in enumerate(splines):
    # Each spline is valid in [xs[i], xs[i+1]]
    mask = (x_vals >= xs[i]) & (x_vals <= xs[i+1])
    y_vals[mask] = [float(S.subs('x', x)) for x in x_vals[mask]]

plt.plot(x_vals, y_vals, label='Cubic Spline')
plt.scatter(xs, ys, color='red', label='Data Points')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Cubic Spline Interpolation')
plt.show()
```

```
1 5 1.0 -4.5 1.5
0 1 5.5 0.0 -1.5
```

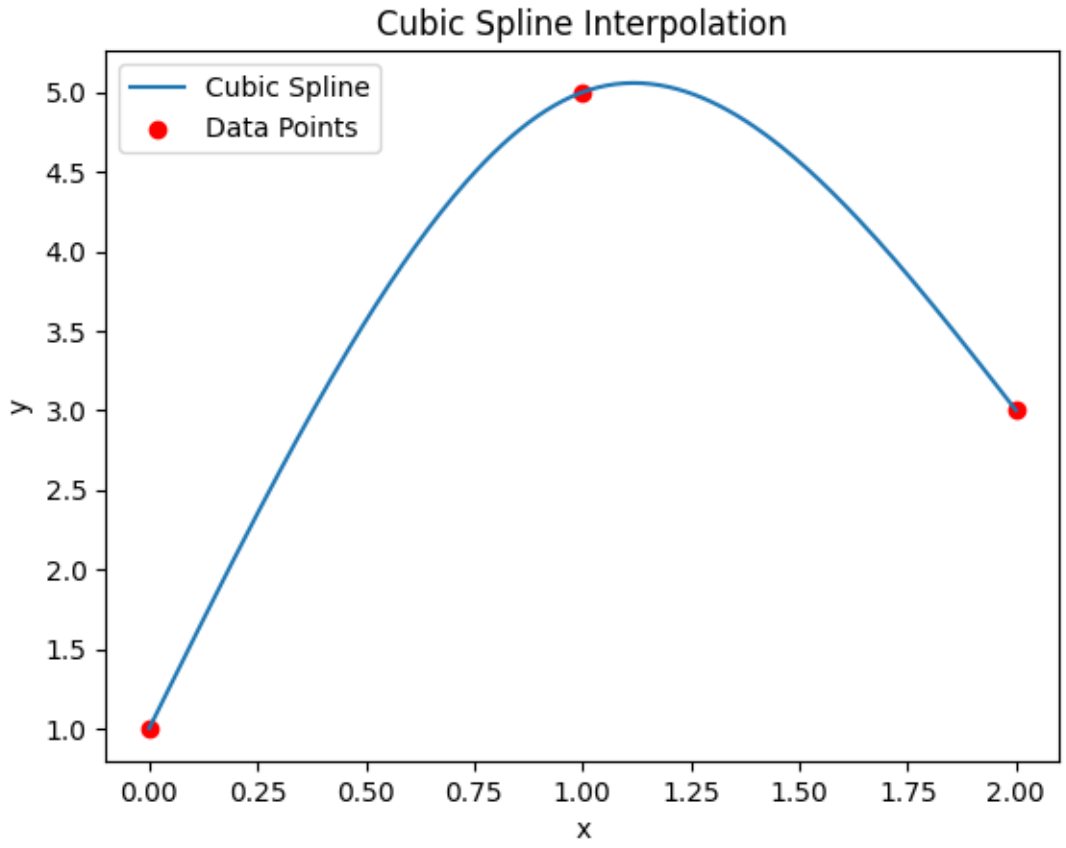
$$-1.5x^3 + 5.5x + 1$$

$$1.0x + 1.5\,(x-1)^3 - 4.5\,(x-1)^2 + 4.0$$

```
------
```

$-1.5x^3 + 5.5x + 1$

$1.5x^3 - 9.0x^2 + 14.5x - 2.0$



2.

$$(0, -5), (1, -4), (2, 3)$$

```
import numpy as np

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]

import matplotlib.pyplot as plt

x_vals = np.linspace(min(xs), max(xs), 200)
y_vals = np.zeros_like(x_vals)
```

```
for i, S in enumerate(splines):
    # Each spline is valid in [xs[i], xs[i+1]]
    mask = (x_vals >= xs[i]) & (x_vals <= xs[i+1])
    y_vals[mask] = [float(S.subs('x', x)) for x in x_vals[mask]]

plt.plot(x_vals, y_vals, label='Cubic Spline')
plt.scatter(xs, ys, color='red', label='Data Points')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Cubic Spline Interpolation')
plt.show()
```

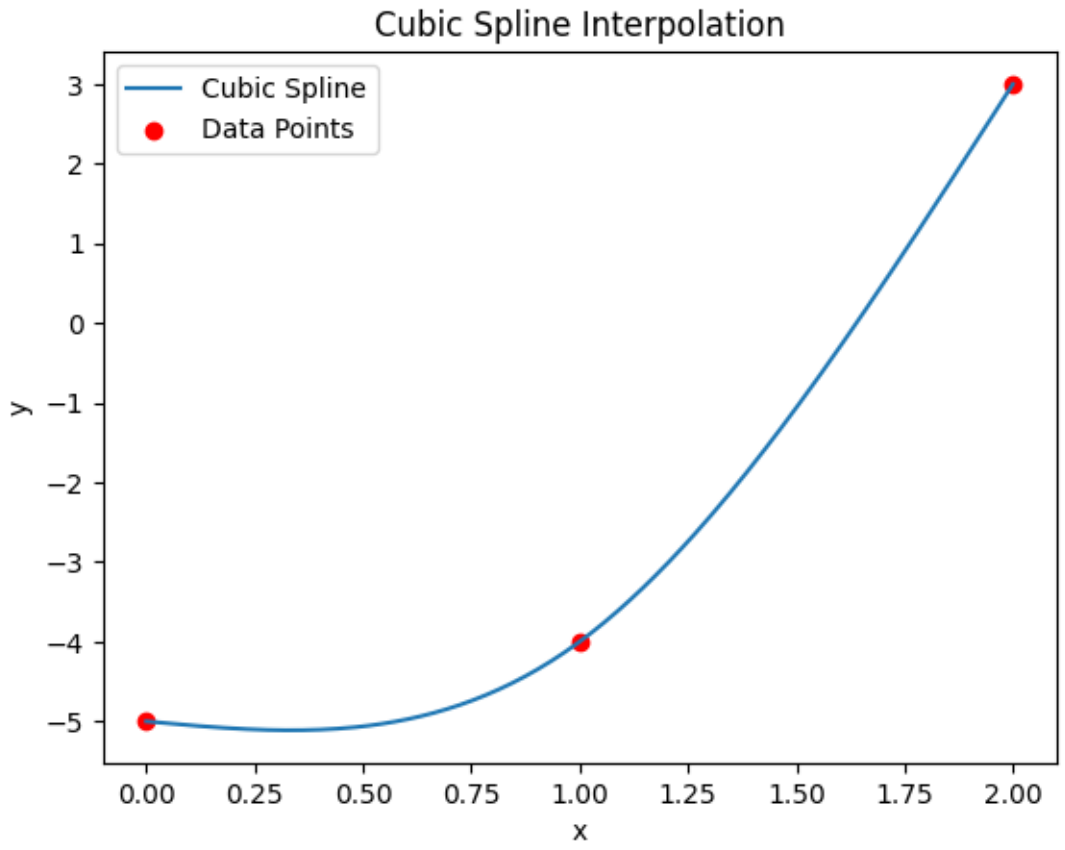
```
1 -4 4.0 4.5 -1.5
0 -5 -0.5 0.0 1.5
```

$$1.5x^3 - 0.5x - 5$$

$$4.0x - 1.5\,(x-1)^3 + 4.5\,(x-1)^2 - 8.0$$

$$1.5x^3 - 0.5x - 5$$

$$-1.5x^3 + 9.0x^2 - 9.5x - 2.0$$



3.

$$(0,-1),(1,1),(2,5),(3,2)$$

```
import numpy as np

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]

import matplotlib.pyplot as plt

x_vals = np.linspace(min(xs), max(xs), 200)
y_vals = np.zeros_like(x_vals)

for i, S in enumerate(splines):
    # Each spline is valid in [xs[i], xs[i+1]]
    mask = (x_vals >= xs[i]) & (x_vals <= xs[i+1])
```

```
    y_vals[mask] = [float(S.subs('x', x)) for x in x_vals[mask]]

plt.plot(x_vals, y_vals, label='Cubic Spline')
plt.scatter(xs, ys, color='red', label='Data Points')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Cubic Spline Interpolation')
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
```

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

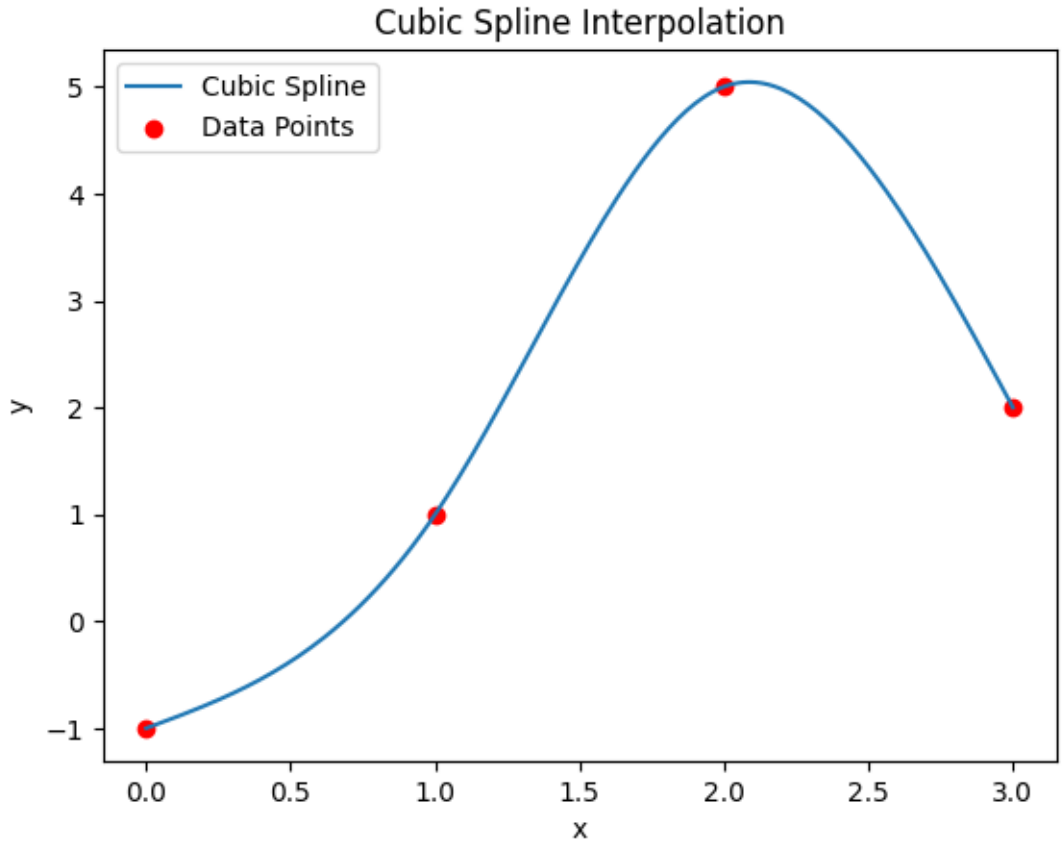
$$4.0x - 3.0\,(x-1)^3 + 3.0\,(x-1)^2 - 3.0$$

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

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

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

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



1.3 Para cada uno de los ejercicios anteriores, resuelva los splines cúbicos de frontera condicionada con $B_0 = 1$ para todos los valores de $B_1 \in R$.

1.3.1 Frntera Condicionada

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


# ######################################################################
def cubic_spline_clamped(
    xs: list[float], ys: list[float], B0: float, B1: float
) -> list[sym.Symbol]:
    """
    Cubic spline interpolation ``S``. Every two points are interpolated by a cubic polyn
    ``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
    - B0, B1: derivatives at the first and last points

    ## 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 + 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(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
```

1.

$$(0,1),(1,5),(2,3)$$

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython.display import HTML
import sympy as sym

# Configuración para el primer conjunto de puntos (0,1), (1,5), (2,3)
xs_points = [0, 1, 2]
ys_points = [1, 5, 3]
B0 = 1  # Derivada fija en el primer punto
B1_values = np.linspace(-10, 10, 50)  # Rango de valores para B1

# Crear la figura
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(-5, 10)
ax.grid(True)
ax.set_title('Splines cúbicos con tangentes en extremos')
line, = ax.plot([], [], 'b-', lw=2, label='Spline cúbico')
points = ax.plot(xs_points, ys_points, 'ro', ms=8, label='Puntos dados')

# Líneas tangentes
tangent0, = ax.plot([], [], 'g--', lw=1.5, label='Tangente en x=0')
tangent1, = ax.plot([], [], 'm--', lw=1.5, label=f'Tangente en x=2 (B1)')
ax.legend()

# Función para evaluar los splines
def evaluate_splines(splines, xs_points, x_range):
    y = []
    for x in x_range:
        for i in range(len(splines)):
            if x >= xs_points[i] and (i == len(splines)-1 or x <= xs_points[i+1]):
                y.append(float(splines[i].subs('x', x)))
                break
```

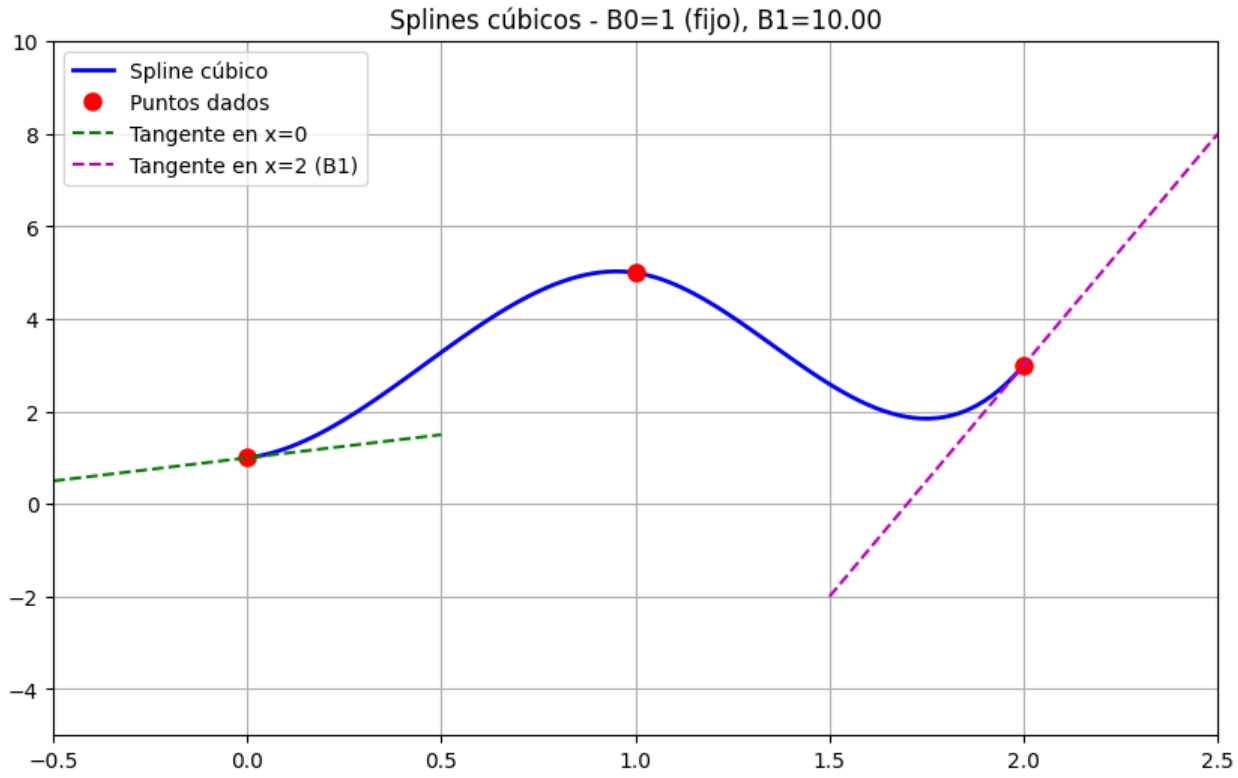
```
    return y

# Función para calcular la tangente en un punto
def get_tangent_line(x0, y0, slope, x_range, length=0.5):
    x_tangent = np.linspace(x0 - length, x0 + length, 2)
    y_tangent = y0 + slope * (x_tangent - x0)
    return x_tangent, y_tangent

# Función de inicialización
def init():
    line.set_data([], [])
    tangent0.set_data([], [])
    tangent1.set_data([], [])
    return line, tangent0, tangent1

# Función de animación
def animate(i):
    B1 = B1_values[i]

    # Calcular splines
    splines = cubic_spline_clamped(xs_points, ys_points, B0, B1)
    x_range = np.linspace(0, 2, 100)
    y_values = evaluate_splines(splines, xs_points, x_range)
    line.set_data(x_range, y_values)

    # Calcular y dibujar tangentes
    # Tangente en x=0 (usando B0)
    x_t0, y_t0 = get_tangent_line(xs_points[0], ys_points[0], B0, x_range)
    tangent0.set_data(x_t0, y_t0)

    # Tangente en x=2 (usando B1 actual)
    x_t1, y_t1 = get_tangent_line(xs_points[-1], ys_points[-1], B1, x_range)
    tangent1.set_data(x_t1, y_t1)

    ax.set_title(f'Splines cúbicos - B0={B0} (fijo), B1={B1:.2f}')
    return line, tangent0, tangent1

# Crear la animación
ani = FuncAnimation(fig, animate, frames=len(B1_values),
                    init_func=init, blit=True, interval=200)

# Mostrar la animación
HTML(ani.to_jshtml())
```

```
1 5 3.75 -3.500000000000001 -2.249999999999999
0 1 1.0 6.25 -3.25
1 5 3.6479591836734686 -3.704081632653062 -1.9438775510204076
0 1 1.0 6.3520408163265305 -3.352040816326531
1 5 3.545918367346939 -3.9081632653061225 -1.637755102040817
0 1 1.0 6.454081632653061 -3.454081632653061
1 5 3.4438775510204076 -4.112244897959185 -1.331632653061223
0 1 1.0 6.556122448979592 -3.556122448979592
1 5 3.341836734693878 -4.316326530612246 -1.025510204081632
0 1 1.0 6.658163265306123 -3.6581632653061233
1 5 3.2397959183673466 -4.520408163265307 -0.7193877551020403
0 1 1.0 6.760204081632653 -3.760204081632653
1 5 3.137755102040817 -4.724489795918368 -0.4132653061224489
0 1 1.0 6.862244897959184 -3.8622448979591844
1 5 3.0357142857142856 -4.92857142857143 -0.10714285714285617
0 1 1.0 6.964285714285715 -3.9642857142857153
1 5 2.933673469387755 -5.132653061224491 0.19897959183673533
0 1 1.0 7.066326530612246 -4.066326530612245
1 5 2.8316326530612246 -5.336734693877551 0.5051020408163264
0 1 1.0 7.168367346938775 -4.168367346938775
1 5 2.729591836734694 -5.540816326530613 0.811224489795919
0 1 1.0 7.270408163265307 -4.270408163265306
1 5 2.6275510204081636 -5.744897959183674 1.1173469387755104
0 1 1.0 7.372448979591837 -4.372448979591837
1 5 2.525510204081633 -5.948979591836735 1.4234693877551023
0 1 1.0 7.474489795918368 -4.474489795918367
1 5 2.4234693877551026 -6.153061224489796 1.7295918367346939
0 1 1.0 7.576530612244898 -4.576530612244898
1 5 2.321428571428572 -6.357142857142858 2.035714285714286
0 1 1.0 7.678571428571429 -4.678571428571429
1 5 2.2193877551020407 -6.561224489795919 2.341836734693878
0 1 1.0 7.780612244897959 -4.780612244897959
1 5 2.11734693877551 -6.76530612244898 2.6479591836734695
0 1 1.0 7.88265306122449 -4.88265306122449
1 5 2.0153061224489797 -6.969387755102041 2.954081632653061
0 1 1.0 7.98469387755102 -4.98469387755102
1 5 1.9132653061224492 -7.173469387755103 3.260204081632653
0 1 1.0 8.08673469387755 -5.086734693877552
1 5 1.8112244897959182 -7.377551020408164 3.5663265306122454
0 1 1.0 8.188775510204081 -5.188775510204081
1 5 1.7091836734693877 -7.581632653061225 3.872448979591837
0 1 1.0 8.290816326530612 -5.290816326530613
1 5 1.6071428571428577 -7.785714285714286 4.178571428571428
0 1 1.0 8.392857142857142 -5.392857142857142
1 5 1.5051020408163267 -7.9897959183673475 4.484693877551021
```

0 1 0.9999999999999996 8.494897959183675 -5.494897959183675
1 5 1.4030612244897953 -8.193877551020408 4.790816326530613
0 1 1.0000000000000004 8.596938775510203 -5.596938775510203
1 5 1.3010204081632657 -8.39795918367347 5.096938775510204
0 1 0.9999999999999996 8.698979591836736 -5.698979591836735
1 5 1.1989795918367343 -8.60204081632653 5.403061224489796
0 1 1.0000000000000004 8.801020408163264 -5.801020408163265
1 5 1.0969387755102047 -8.806122448979592 5.709183673469387
0 1 0.9999999999999996 8.903061224489797 -5.903061224489797
1 5 0.9948979591836742 -9.010204081632654 6.0153061224489806
0 1 1.0 9.005102040816327 -6.005102040816328
1 5 0.8928571428571437 -9.214285714285715 6.321428571428572
0 1 1.0 9.107142857142858 -6.107142857142858
1 5 0.7908163265306132 -9.418367346938776 6.627551020408163
0 1 1.0 9.209183673469388 -6.209183673469387
1 5 0.6887755102040813 -9.622448979591837 6.933673469387756
0 1 1.0 9.311224489795919 -6.311224489795919
1 5 0.5867346938775508 -9.826530612244898 7.239795918367347
0 1 1.0 9.41326530612245 -6.41326530612245
1 5 0.48469387755102034 -10.03061224489796 7.545918367346939
0 1 1.0 9.51530612244898 -6.51530612244898
1 5 0.38265306122448983 -10.23469387755102 7.852040816326531
0 1 1.0 9.61734693877551 -6.617346938775509
1 5 0.28061224489795933 -10.438775510204081 8.158163265306122
0 1 1.0 9.71938775510204 -6.719387755102041
1 5 0.1785714285714275 -10.642857142857142 8.464285714285715
0 1 1.0 9.821428571428571 -6.821428571428572
1 5 0.07653061224489699 -10.846938775510203 8.770408163265307
0 1 1.0 9.923469387755102 -6.923469387755102
1 5 -0.025510204081632848 -11.051020408163266 9.0765306122449
0 1 0.9999999999999996 10.025510204081634 -7.025510204081634
1 5 -0.1275510204081638 -11.255102040816325 9.38265306122449
0 1 1.0 10.127551020408163 -7.127551020408163
1 5 -0.22959183673469385 -11.459183673469388 9.688775510204081
0 1 0.9999999999999996 10.229591836734695 -7.229591836734694
1 5 -0.3316326530612248 -11.663265306122447 9.994897959183673
0 1 1.0 10.331632653061224 -7.331632653061224
1 5 -0.43367346938775486 -11.86734693877551 10.301020408163266
0 1 0.9999999999999996 10.433673469387756 -7.433673469387756
1 5 -0.5357142857142858 -12.07142857142857 10.607142857142856
0 1 1.0 10.535714285714285 -7.535714285714285
1 5 -0.6377551020408159 -12.275510204081632 10.913265306122447
0 1 0.9999999999999996 10.637755102040817 -7.637755102040816
1 5 -0.7397959183673468 -12.479591836734695 11.219387755102042
0 1 1.0 10.739795918367347 -7.739795918367347

```
1 5 -0.841836734693878 -12.683673469387756 11.525510204081634
0 1 1.0 10.841836734693878 -7.841836734693878
1 5 -0.9438775510204078 -12.887755102040817 11.831632653061225
0 1 1.0 10.943877551020408 -7.943877551020409
1 5 -1.045918367346939 -13.091836734693878 12.137755102040819
0 1 1.0 11.045918367346939 -8.045918367346939
1 5 -1.147959183673469 -13.295918367346939 12.443877551020407
0 1 1.0 11.14795918367347 -8.14795918367347
1 5 -1.25 -13.5 12.75
0 1 1.0 11.25 -8.25
```

```
<IPython.core.display.HTML object>
```



2.

$$(0,-5),(1,-4),(2,3)$$

```python
# Configuración para el primer conjunto de puntos (0,1), (1,5), (2,3)
xs_points = [0, 1, 2]
ys_points = [-5, -4, 3]
B0 = 1  # Derivada fija en el primer punto
B1_values = np.linspace(-10, 10, 50)  # Rango de valores para B1
```

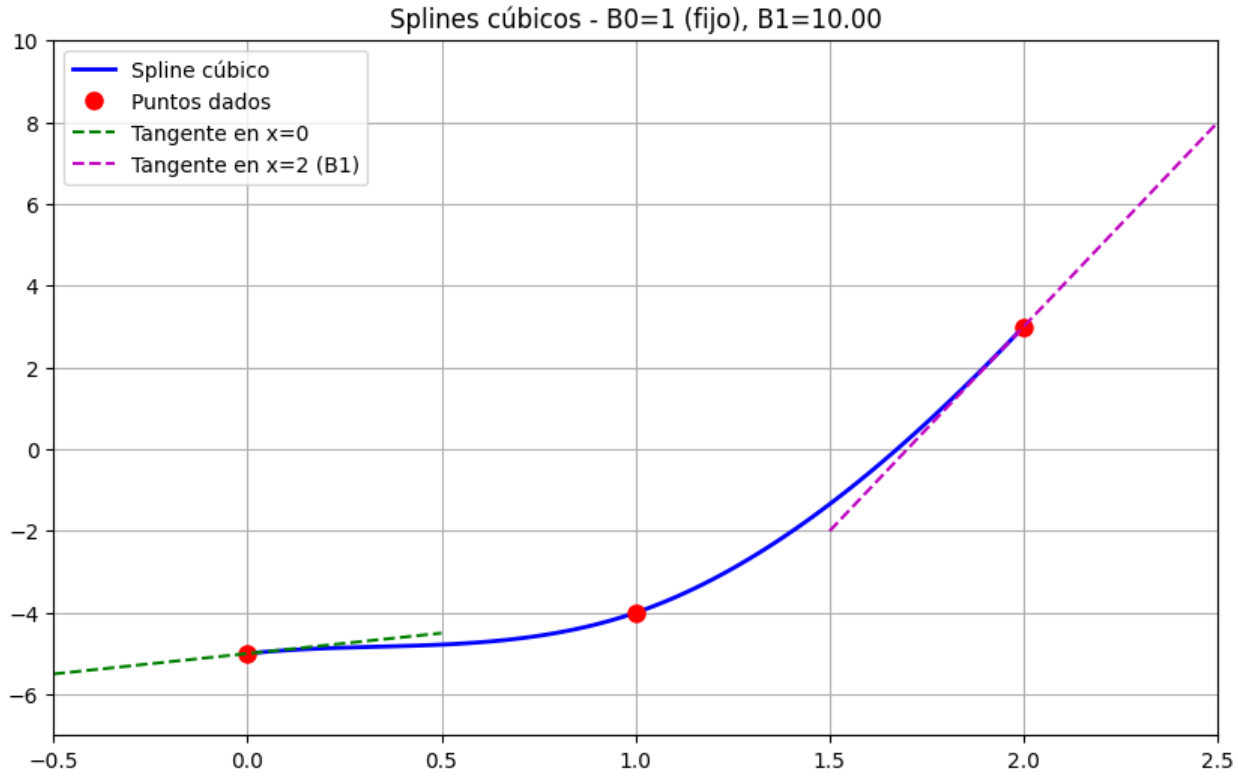
```
# Crear la figura
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(-7, 10)
ax.grid(True)
ax.set_title('Splines cúbicos con tangentes en extremos')
line, = ax.plot([], [], 'b-', lw=2, label='Spline cúbico')
points = ax.plot(xs_points, ys_points, 'ro', ms=8, label='Puntos dados')

# Líneas tangentes
tangent0, = ax.plot([], [], 'g--', lw=1.5, label='Tangente en x=0')
tangent1, = ax.plot([], [], 'm--', lw=1.5, label=f'Tangente en x=2 (B1)')
ax.legend()

# Función para evaluar los splines
def evaluate_splines(splines, xs_points, x_range):
    y = []
    for x in x_range:
        for i in range(len(splines)):
            if x >= xs_points[i] and (i == len(splines)-1 or x <= xs_points[i+1]):
                y.append(float(splines[i].subs('x', x)))
                break
    return y

# Función para calcular la tangente en un punto
def get_tangent_line(x0, y0, slope, x_range, length=0.5):
    x_tangent = np.linspace(x0 - length, x0 + length, 2)
    y_tangent = y0 + slope * (x_tangent - x0)
    return x_tangent, y_tangent

# Función de inicialización
def init():
    line.set_data([], [])
    tangent0.set_data([], [])
    tangent1.set_data([], [])
    return line, tangent0, tangent1

# Función de animación
def animate(i):
    B1 = B1_values[i]

    # Calcular splines
    splines = cubic_spline_clamped(xs_points, ys_points, B0, B1)
    x_range = np.linspace(0, 2, 100)
```

```
    y_values = evaluate_splines(splines, xs_points, x_range)
    line.set_data(x_range, y_values)

    # Calcular y dibujar tangentes
    # Tangente en x=0 (usando B0)
    x_t0, y_t0 = get_tangent_line(xs_points[0], ys_points[0], B0, x_range)
    tangent0.set_data(x_t0, y_t0)

    # Tangente en x=2 (usando B1 actual)
    x_t1, y_t1 = get_tangent_line(xs_points[-1], ys_points[-1], B1, x_range)
    tangent1.set_data(x_t1, y_t1)

    ax.set_title(f'Splines cúbicos - B0={B0} (fijo), B1={B1:.2f}')
    return line, tangent0, tangent1

# Crear la animación
ani2 = FuncAnimation(fig, animate, frames=len(B1_values),
                     init_func=init, blit=True, interval=200)

# Mostrar la animación
HTML(ani2.to_jshtml())
```

```
1 -4 8.25 14.5 -15.75
0 -5 1.0 -7.25 7.25
1 -4 8.14795918367347 14.295918367346939 -15.443877551020407
0 -5 1.0 -7.1479591836734695 7.147959183673469
1 -4 8.045918367346939 14.091836734693878 -15.137755102040819
0 -5 1.0 -7.045918367346939 7.045918367346939
1 -4 7.943877551020407 13.887755102040817 -14.831632653061225
0 -5 1.0 -6.9438775510204085 6.943877551020409
1 -4 7.841836734693877 13.683673469387756 -14.525510204081632
0 -5 1.0 -6.841836734693878 6.841836734693878
1 -4 7.739795918367346 13.479591836734695 -14.21938775510204
0 -5 1.0 -6.7397959183673475 6.739795918367347
1 -4 7.637755102040816 13.275510204081634 -13.91326530612245
0 -5 1.0 -6.637755102040817 6.637755102040817
1 -4 7.535714285714286 13.071428571428571 -13.607142857142856
0 -5 1.0 -6.535714285714286 6.535714285714286
1 -4 7.433673469387755 12.86734693877551 -13.301020408163266
0 -5 1.0 -6.433673469387755 6.433673469387755
1 -4 7.331632653061225 12.66326530612245 -12.994897959183675
0 -5 1.0 -6.331632653061225 6.331632653061225
1 -4 7.229591836734694 12.459183673469388 -12.688775510204081
0 -5 1.0 -6.229591836734694 6.229591836734694
```

```
1 -4 7.127551020408164 12.255102040816325 -12.382653061224488
0 -5 1.0 -6.127551020408163 6.127551020408163
1 -4 7.025510204081633 12.051020408163266 -12.0765306122449
0 -5 1.0 -6.025510204081633 6.025510204081633
1 -4 6.923469387755103 11.846938775510203 -11.770408163265307
0 -5 1.0 -5.923469387755102 5.923469387755102
1 -4 6.821428571428572 11.642857142857144 -11.464285714285715
0 -5 1.0 -5.821428571428572 5.821428571428572
1 -4 6.719387755102041 11.438775510204081 -11.158163265306122
0 -5 1.0 -5.719387755102041 5.719387755102041
1 -4 6.617346938775511 11.23469387755102 -10.85204081632653
0 -5 1.0 -5.61734693877551 5.617346938775509
1 -4 6.51530612244898 11.03061224489796 -10.545918367346939
0 -5 1.0 -5.51530612244898 5.51530612244898
1 -4 6.413265306122448 10.826530612244897 -10.239795918367344
0 -5 1.0 -5.413265306122448 5.413265306122448
1 -4 6.311224489795918 10.622448979591837 -9.933673469387754
0 -5 1.0 -5.311224489795919 5.311224489795919
1 -4 6.209183673469387 10.418367346938775 -9.627551020408163
0 -5 1.0 -5.209183673469387 5.209183673469387
1 -4 6.107142857142857 10.214285714285714 -9.32142857142857
0 -5 1.0 -5.107142857142857 5.107142857142857
1 -4 6.005102040816326 10.01020408163265 -9.015306122448978
0 -5 1.0 -5.005102040816325 5.005102040816325
1 -4 5.903061224489796 9.806122448979592 -8.709183673469388
0 -5 1.0 -4.903061224489796 4.903061224489796
1 -4 5.801020408163265 9.60204081632653 -8.403061224489795
0 -5 1.0 -4.801020408163265 4.801020408163265
1 -4 5.698979591836735 9.39795918367347 -8.096938775510203
0 -5 1.0 -4.698979591836735 4.698979591836735
1 -4 5.596938775510204 9.193877551020408 -7.790816326530613
0 -5 1.0 -4.596938775510204 4.596938775510204
1 -4 5.494897959183673 8.989795918367347 -7.48469387755102
0 -5 1.0 -4.494897959183674 4.494897959183674
1 -4 5.392857142857142 8.785714285714286 -7.178571428571428
0 -5 1.0 -4.392857142857143 4.392857142857143
1 -4 5.290816326530612 8.581632653061225 -6.872448979591837
0 -5 1.0 -4.290816326530613 4.290816326530613
1 -4 5.1887755102040805 8.377551020408163 -6.566326530612244
0 -5 1.0 -4.188775510204081 4.188775510204081
1 -4 5.08673469387755 8.173469387755102 -6.260204081632652
0 -5 1.0 -4.086734693877551 4.086734693877551
1 -4 4.98469387755102 7.969387755102041 -5.95408163265306
0 -5 1.0 -3.9846938775510203 3.9846938775510203
1 -4 4.88265306122449 7.76530612244898 -5.647959183673469
```

```
0 -5 1.0 -3.88265306122449 3.88265306122449
1 -4 4.780612244897959 7.561224489795919 -5.341836734693878
0 -5 1.0 -3.7806122448979593 3.7806122448979593
1 -4 4.678571428571428 7.357142857142858 -5.035714285714286
0 -5 1.0 -3.678571428571429 3.678571428571429
1 -4 4.576530612244897 7.153061224489797 -4.729591836734694
0 -5 1.0 -3.5765306122448983 3.5765306122448983
1 -4 4.474489795918367 6.948979591836736 -4.423469387755103
0 -5 1.0 -3.474489795918368 3.474489795918368
1 -4 4.3724489795918355 6.744897959183673 -4.117346938775509
0 -5 1.0 -3.3724489795918364 3.372448979591836
1 -4 4.270408163265305 6.540816326530612 -3.8112244897959173
0 -5 1.0 -3.270408163265306 3.2704081632653064
1 -4 4.1683673469387745 6.336734693877551 -3.5051020408163267
0 -5 1.0 -3.1683673469387754 3.168367346938775
1 -4 4.066326530612246 6.13265306122449 -3.1989795918367343
0 -5 1.0 -3.066326530612245 3.0663265306122454
1 -4 3.964285714285714 5.928571428571429 -2.8928571428571437
0 -5 1.0 -2.9642857142857144 2.964285714285714
1 -4 3.8622448979591844 5.724489795918368 -2.5867346938775517
0 -5 1.0 -2.862244897959184 2.8622448979591844
1 -4 3.7602040816326525 5.520408163265305 -2.2806122448979576
0 -5 1.0 -2.7602040816326525 2.7602040816326525
1 -4 3.658163265306122 5.316326530612244 -1.974489795918366
0 -5 1.0 -2.658163265306122 2.658163265306122
1 -4 3.556122448979591 5.112244897959184 -1.6683673469387748
0 -5 1.0 -2.556122448979592 2.556122448979592
1 -4 3.4540816326530606 4.908163265306123 -1.3622448979591832
0 -5 1.0 -2.4540816326530615 2.4540816326530615
1 -4 3.35204081632653 4.704081632653062 -1.056122448979592
0 -5 1.0 -2.352040816326531 2.352040816326531
1 -4 3.2499999999999996 4.500000000000001 -0.7500000000000004
0 -5 1.0 -2.2500000000000004 2.2500000000000004


<IPython.core.display.HTML object>
```



3.

$$(0,-1),(1,1),(2,5),(3,2)$$

```
# Configuración para el primer conjunto de puntos
xs_points = [0, 1, 2, 3]
ys_points = [-1, 1, 5, 2]
B0 = 1  # Derivada fija en el primer punto
B1_values = np.linspace(-10, 10, 50)  # Rango de valores para B1

# Crear la figura
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(-5, 10)
ax.grid(True)
ax.set_title('Splines cúbicos con tangentes en extremos')
line, = ax.plot([], [], 'b-', lw=2, label='Spline cúbico')
points = ax.plot(xs_points, ys_points, 'ro', ms=8, label='Puntos dados')

# Líneas tangentes
tangent0, = ax.plot([], [], 'g--', lw=1.5, label='Tangente en x=0')
tangent1, = ax.plot([], [], 'm--', lw=1.5, label=f'Tangente en x=2 (B1)')
ax.legend()
```

```
# Función para evaluar los splines
def evaluate_splines(splines, xs_points, x_range):
    y = []
    for x in x_range:
        for i in range(len(splines)):
            if x >= xs_points[i] and (i == len(splines)-1 or x <= xs_points[i+1]):
                y.append(float(splines[i].subs('x', x)))
                break
    return y

# Función para calcular la tangente en un punto
def get_tangent_line(x0, y0, slope, x_range, length=0.5):
    x_tangent = np.linspace(x0 - length, x0 + length, 2)
    y_tangent = y0 + slope * (x_tangent - x0)
    return x_tangent, y_tangent

# Función de inicialización
def init():
    line.set_data([], [])
    tangent0.set_data([], [])
    tangent1.set_data([], [])
    return line, tangent0, tangent1

# Función de animación
def animate(i):
    B1 = B1_values[i]

    # Calcular splines
    splines = cubic_spline_clamped(xs_points, ys_points, B0, B1)
    x_range = np.linspace(0, 2, 100)
    y_values = evaluate_splines(splines, xs_points, x_range)
    line.set_data(x_range, y_values)

    # Calcular y dibujar tangentes
    # Tangente en x=0 (usando B0)
    x_t0, y_t0 = get_tangent_line(xs_points[0], ys_points[0], B0, x_range)
    tangent0.set_data(x_t0, y_t0)

    # Tangente en x=2 (usando B1 actual)
    x_t1, y_t1 = get_tangent_line(xs_points[-1], ys_points[-1], B1, x_range)
    tangent1.set_data(x_t1, y_t1)

    ax.set_title(f'Splines cúbicos - B0={B0} (fijo), B1={B1:.2f}')
    return line, tangent0, tangent1
```

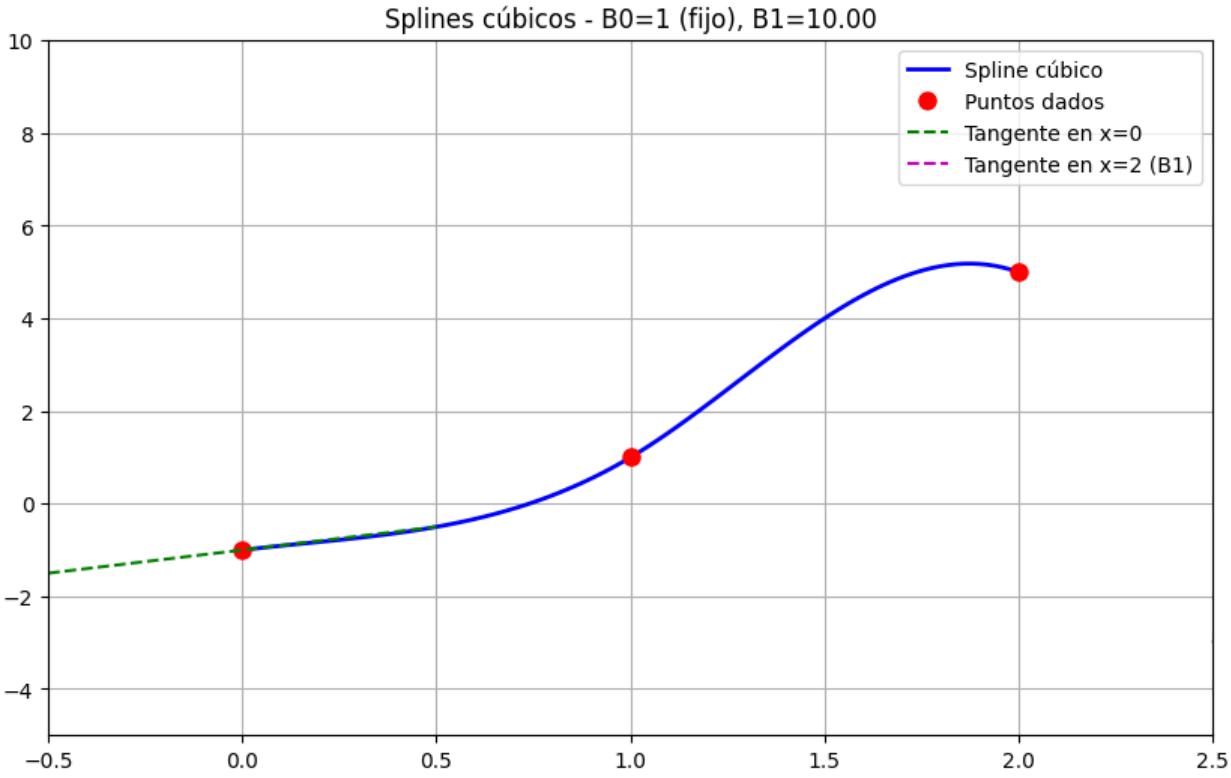
```
# Crear la animación
ani3 = FuncAnimation(fig, animate, frames=len(B1_values),
                     init_func=init, blit=True, interval=200)

# Mostrar la animación
HTML(ani3.to_jshtml())
```

```
2 5 2.333333333333333 -3.666666666666666 -1.6666666666666667
1 1 3.6666666666666665 2.333333333333333 -1.9999999999999998
0 -1 1.0 0.3333333333333335 0.6666666666666665
2 5 2.224489795918367 -3.8571428571428563 -1.3673469387755108
1 1 3.693877551020408 2.387755102040816 -2.081632653061224
0 -1 1.0 0.30612244897959195 0.693877551020408
2 5 2.115646258503401 -4.047619047619047 -1.0680272108843545
1 1 3.7210884353741496 2.442176870748299 -2.1632653061224487
0 -1 1.0 0.2789115646258504 0.7210884353741496
2 5 2.0068027210884347 -4.238095238095238 -0.7687074829931966
1 1 3.748299319727891 2.496598639455782 -2.2448979591836733
0 -1 1.0 0.2517006802721089 0.7482993197278911
2 5 1.8979591836734686 -4.428571428571428 -0.46938775510204067
1 1 3.7755102040816326 2.5510204081632653 -2.326530612244898
0 -1 1.0 0.22448979591836737 0.7755102040816326
2 5 1.7891156462585025 -4.619047619047619 -0.17006802721088418
1 1 3.802721088435374 2.6054421768707483 -2.4081632653061225
0 -1 1.0 0.19727891156462585 0.8027210884353742
2 5 1.6802721088435364 -4.809523809523808 0.12925170068027145
1 1 3.8299319727891157 2.659863945578231 -2.489795918367346
0 -1 1.0 0.17006802721088454 0.8299319727891156
2 5 1.5714285714285703 -4.999999999999999 0.4285714285714284
1 1 3.857142857142857 2.714285714285714 -2.571428571428571
0 -1 1.0 0.14285714285714302 0.8571428571428571
2 5 1.462585034013605 -5.19047619047619 0.7278911564625847
1 1 3.8843537414965987 2.768707482993197 -2.6530612244897953
0 -1 1.0 0.11564625850340149 0.8843537414965986
2 5 1.353741496598639 -5.38095238095238 1.0272108843537406
1 1 3.9115646258503403 2.8231292517006796 -2.73469387755102
0 -1 1.0 0.08843537414966018 0.9115646258503398
2 5 1.2448979591836729 -5.57142857142857 1.3265306122448977
1 1 3.938775510204082 2.877551020408162 -2.816326530612244
0 -1 1.0 0.06122448979591866 0.9387755102040813
2 5 1.1360544217687076 -5.761904761904761 1.625850340136054
1 1 3.9659863945578233 2.9319727891156457 -2.897959183673469
0 -1 1.0 0.03401360544217713 0.9659863945578229
```

```
2 5 1.0272108843537415 -5.952380952380952 1.9251700680272104
1 1 3.993197278911565 2.986394557823129 -2.9795918367346936
0 -1 1.0 0.006802721088435604 0.9931972789115644
2 5 0.918367346938775 -6.1428571428571415 2.2244897959183665
1 1 4.020408163265306 3.040816326530612 -3.0612244897959173
0 -1 1.0 -0.020408163265305923 1.020408163265306
2 5 0.8095238095238089 -6.333333333333332 2.5238095238095233
1 1 4.0476190476190474 3.095238095238095 -3.1428571428571423
0 -1 1.0 -0.04761904761904745 1.0476190476190474
2 5 0.7006802721088428 -6.523809523809523 2.82312925170068
1 1 4.074829931972789 3.149659863945578 -3.2244897959183674
0 -1 1.0 -0.07482993197278898 1.074829931972789
2 5 0.5918367346938767 -6.7142857142857135 3.122448979591836
1 1 4.1020408163265305 3.204081632653061 -3.3061224489795915
0 -1 1.0 -0.1020408163265305 1.1020408163265305
2 5 0.48299319727891055 -6.904761904761903 3.421768707482992
1 1 4.129251700680272 3.258503401360544 -3.3877551020408156
0 -1 1.0 -0.12925170068027203 1.129251700680272
2 5 0.37414965986394444 -7.095238095238094 3.721088435374149
1 1 4.156462585034014 3.312925170068027 -3.4693877551020407
0 -1 1.0 -0.15646258503401356 1.1564625850340136
2 5 0.2653061224489792 -7.285714285714285 4.020408163265306
1 1 4.183673469387755 3.36734693877551 -3.551020408163265
0 -1 1.0 -0.18367346938775508 1.183673469387755
2 5 0.15646258503401356 -7.476190476190475 4.319727891156462
1 1 4.210884353741497 3.421768707482993 -3.6326530612244894
0 -1 1.0 -0.2108843537414966 1.2108843537414966
2 5 0.04761904761904745 -7.666666666666665 4.619047619047618
1 1 4.238095238095238 3.476190476190454 -3.7142857142857135
0 -1 1.0 -0.2380952380952377 1.2380952380952377
2 5 -0.0612244897959191 -7.857142857142857 4.918367346938776
1 1 4.26530612244898 3.5306122448979593 -3.795918367346939
0 -1 1.0 -0.26530612244897966 1.2653061224489797
2 5 -0.17006802721088476 -8.047619047619047 5.217687074829932
1 1 4.292517006802721 3.5850340136054424 -3.877551020408163
0 -1 1.0 -0.2925170068027212 1.2925170068027212
2 5 -0.27891156462585087 -8.238095238095237 5.517006802721088
1 1 4.319727891156463 3.6394557823129245 -3.9591836734693877
0 -1 1.0 -0.3197278911564623 1.3197278911564623
2 5 -0.387755102040817 -8.428571428571427 5.816326530612244
1 1 4.346938775510204 3.6938775510204076 -4.040816326530611
0 -1 1.0 -0.3469387755102038 1.346938775510204
2 5 -0.4965986394557831 -8.619047619047617 6.1156462585034
1 1 4.374149659863945 3.7482993197278907 -4.122448979591836
0 -1 1.0 -0.3741496598639453 1.374149659863945
```

```
2 5 -0.6054421768707492 -8.809523809523808 6.4149659863945585
1 1 4.401360544217687 3.8027210884353737 -4.204081632653061
0 -1 1.0 -0.40136054421768685 1.401360544217687
2 5 -0.7142857142857153 -8.999999999999998 6.714285714285713
1 1 4.428571428571428 3.8571428571428568 -4.285714285714285
0 -1 1.0 -0.4285714285714284 1.4285714285714282
2 5 -0.8231292517006801 -9.19047619047619 7.01360544217687
1 1 4.45578231292517 3.91156462585034 -4.367346938775509
0 -1 1.0 -0.4557823129251699 1.4557823129251701
2 5 -0.9319727891156475 -9.38095238095238 7.312925170068027
1 1 4.482993197278911 3.965986394557823 -4.448979591836735
0 -1 1.0 -0.48299319727891143 1.4829931972789112
2 5 -1.0408163265306136 -9.57142857142857 7.612244897959183
1 1 4.5102040816326525 4.020408163265306 -4.530612244897958
0 -1 1.0 -0.510204081632653 1.5102040816326532
2 5 -1.1496598639455786 -9.761904761904761 7.911564625850339
1 1 4.537414965986394 4.074829931972789 -4.612244897959183
0 -1 1.0 -0.5374149659863945 1.5374149659863943
2 5 -1.2585034013605452 -9.95238095238095 8.210884353741497
1 1 4.564625850340136 4.129251700680272 -4.693877551020408
0 -1 1.0 -0.564625850340136 1.5646258503401362
2 5 -1.3673469387755102 -10.142857142857142 8.510204081632653
1 1 4.591836734693877 4.183673469387755 -4.775510204081633
0 -1 1.0 -0.5918367346938775 1.5918367346938773
2 5 -1.476190476190478 -10.333333333333332 8.80952380952381
1 1 4.6190476190476195 4.238095238095237 -4.857142857142857
0 -1 1.0 -0.6190476190476186 1.6190476190476186
2 5 -1.585034013605443 -10.523809523809524 9.108843537414968
1 1 4.64625850340136 4.292517006802721 -4.938775510204081
0 -1 1.0 -0.6462585034013606 1.6462585034013604
2 5 -1.6938775510204092 -10.714285714285714 9.408163265306122
1 1 4.673469387755103 4.346938775510203 -5.020408163265306
0 -1 1.0 -0.6734693877551017 1.6734693877551017
2 5 -1.8027210884353753 -10.904761904761903 9.70748299319728
1 1 4.700680272108843 4.401360544217686 -5.10204081632653
0 -1 1.0 -0.7006802721088432 1.7006802721088432
2 5 -1.91156462585034 -11.095238095238095 10.006802721088436
1 1 4.727891156462586 4.4557823129251695 -5.183673469387755
0 -1 1.0 -0.7278911564625847 1.7278911564625847
2 5 -2.0204081632653064 -11.285714285714285 10.306122448979592
1 1 4.755102040816326 4.5102040816326525 -5.265306122448979
0 -1 1.0 -0.7551020408163263 1.7551020408163263
2 5 -2.1292517006802725 -11.476190476190474 10.605442176870747
1 1 4.782312925170068 4.564625850340136 -5.346938775510203
0 -1 1.0 -0.7823129251700678 1.7823129251700678
```

```
2 5 -2.2380952380952386 -11.666666666666664 10.904761904761903
1 1 4.809523809523809 4.619047619047619 -5.428571428571428
0 -1 1.0 -0.8095238095238093 1.8095238095238093
2 5 -2.3469387755102047 -11.857142857142854 11.204081632653057
1 1 4.836734693877551 4.673469387755101 -5.510204081632652
0 -1 1.0 -0.8367346938775504 1.8367346938775502
2 5 -2.455782312925171 -12.047619047619047 11.503401360544217
1 1 4.863945578231292 4.727891156462585 -5.591836734693878
0 -1 1.0 -0.8639455782312924 1.8639455782312924
2 5 -2.564625850340137 -12.238095238095237 11.802721088435375
1 1 4.891156462585034 4.782312925170068 -5.673469387755102
0 -1 1.0 -0.8911564625850339 1.891156462585034
2 5 -2.673469387755104 -12.428571428571427 12.10204081632653
1 1 4.918367346938775 4.836734693877551 -5.755102040816325
0 -1 1.0 -0.9183673469387754 1.9183673469387754
2 5 -2.782312925170069 -12.619047619047619 12.401360544217688
1 1 4.945578231292517 4.891156462585034 -5.83673469387755
0 -1 1.0 -0.945578231292517 1.945578231292517
2 5 -2.8911564625850352 -12.809523809523808 12.700680272108842
1 1 4.9727891156462585 4.945578231292517 -5.918367346938775
0 -1 1.0 -0.9727891156462585 1.9727891156462585
2 5 -3.0 -13.0 13.0
1 1 5.0 5.0 -6.0
0 -1 1.0 -1.0 2.0

<IPython.core.display.HTML object>
```


Splines cúbicos - B0=1 (fijo), B1=10.00
Spline cúbico
Puntos dados
Tangente en x=0
Tangente en x=2 (B1)
10
8
6
4
2
0
−2
−4
−0.5
0.0
0.5
1.0
1.5
2.0
2.5