Taller 04

Nombre: David Alejandro Díaz Pineda

In [1]:
```python
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
    `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_

    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

    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]
```

```
        S = a + b*(x - xs[j]) + c[j]*(x - xs[j])**2 + d*(x - xs[j])**3

        print(j, a, b, c[j], d)
        splines.append(S)

    splines.reverse()
    return splines
```

In [2]:
```
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]
```

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

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

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

```
______
```

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

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

Compruebe gráficamente la solución de los siguientes ejercicios:

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

In [3]:
```
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]
```

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

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

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

```
______
```

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

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

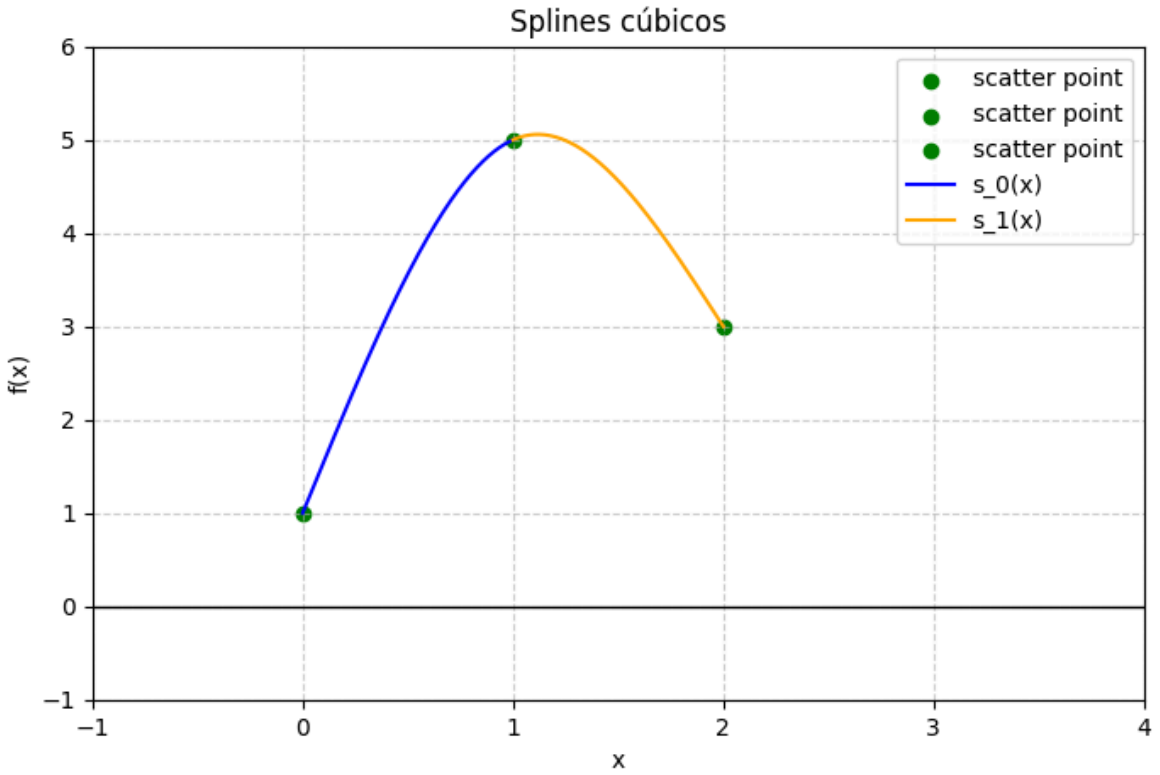
In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
x0,y0 = 0,1
x1, y1= 1,5
x2, y2= 2,3

xs0 = np.linspace(0, 1, 200)
xs1 = np.linspace(1, 2, 200)
```

```
def s_0(x):
    return 1 + 5.5 *x - 1.5 * x **3

def s_1(x):
    return -2 + 14.5*x - 9*x**2 + 1.5*x**3


plt.figure(figsize=(8, 5))
plt.scatter([0.0], [1.0], color='green', label='scatter point')
plt.scatter([1.0], [5.0], color='green', label='scatter point')
plt.scatter([2.0], [3.0], color='green', label='scatter point')
plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')
plt.plot(xs1, s_1(xs1), label='s_1(x)', color='orange')
plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Splines cúbicos')
plt.legend()
plt.xlim(-1, 4)
plt.ylim(-1, 6)
plt.show()
```



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

In [5]:
```
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]
```

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

$1.5x^3 - 0.5x - 5$

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

$1.5x^3 - 0.5x - 5$

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

In [6]:

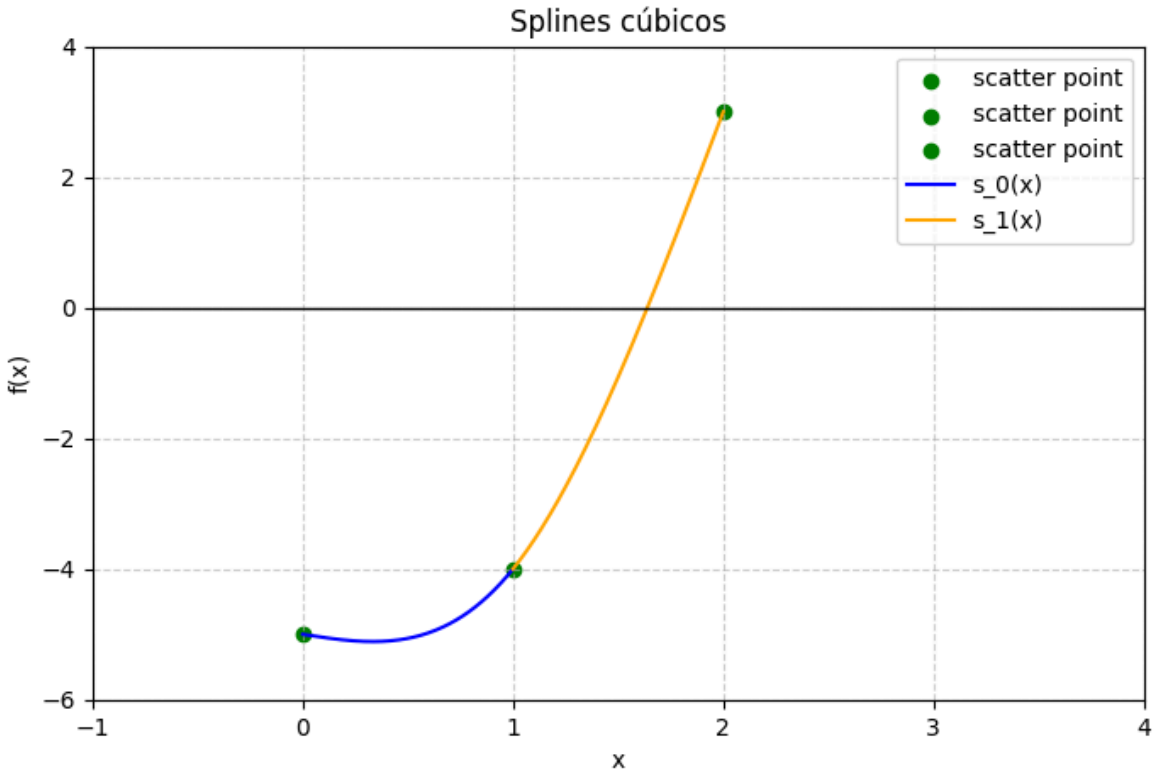
```python
import numpy as np
import matplotlib.pyplot as plt

xs0 = np.linspace(0, 1, 200)
xs1 = np.linspace(1, 2, 200)


def s_0(x):
    return -5-0.5*x + 1.5*x**3

def s_1(x):
    return -2-9.5*x + 9*x**2 -1.5*x**3


plt.figure(figsize=(8, 5))
plt.scatter([0.0], [-5.0], color='green', label='scatter point')
plt.scatter([1.0], [-4.0], color='green', label='scatter point')
plt.scatter([2.0], [3.0], color='green', label='scatter point')
plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')
plt.plot(xs1, s_1(xs1), label='s_1(x)', color='orange')
plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Splines cúbicos')
plt.legend()
plt.xlim(-1, 4)
plt.ylim(-6, 4)
plt.show()
```



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

In [7]:
```
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]
```

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

$1.0x^3 + 1.0x - 1$

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

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

_____

$1.0x^3 + 1.0x - 1$

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

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

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

xs0 = np.linspace(0, 1, 200)
xs1 = np.linspace(1, 2, 200)
xs3 = np.linspace(2, 3, 200)

def s_0(x):
```

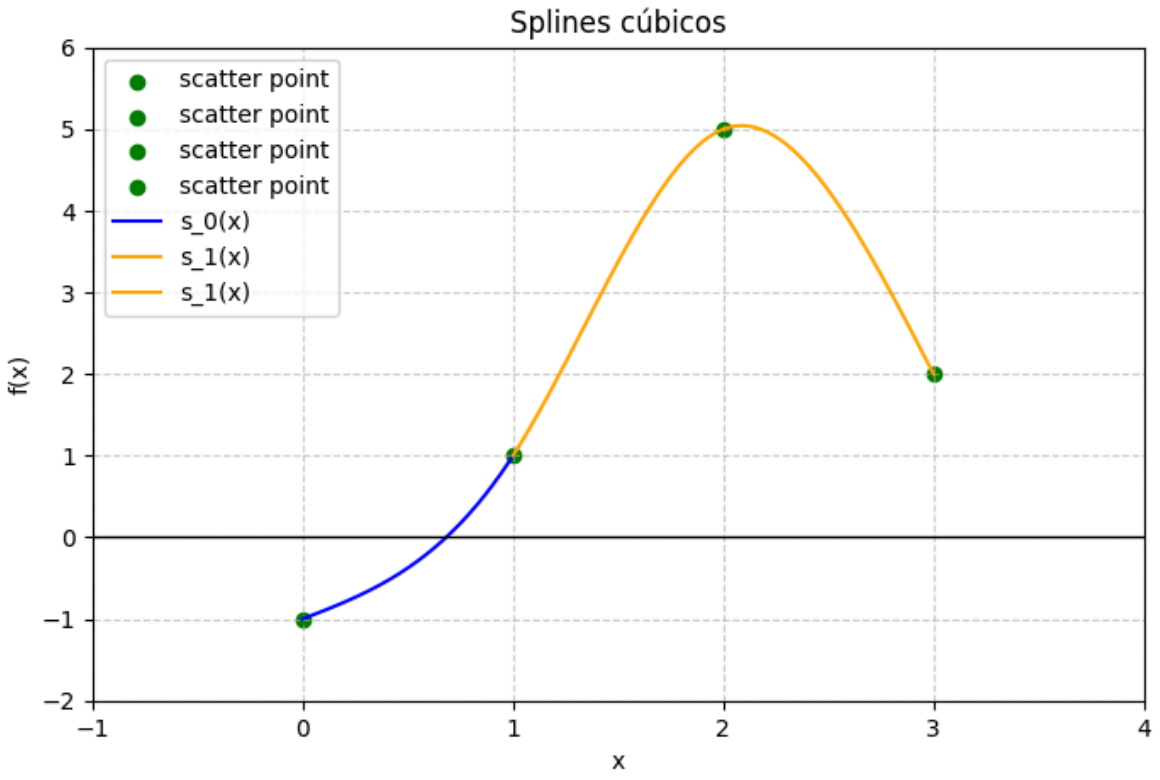
```
    return 1*x**3+1*x-1

def s_1(x):
    return -3*x**3+12*x**2-11*x+3

def s_2(x):
    return 2*x**3 - 18*x**2+ 49*x -37

plt.figure(figsize=(8, 5))
plt.scatter([0.0], [-1.0], color='green', label='scatter point')
plt.scatter([1.0], [1.0], color='green', label='scatter point')
plt.scatter([2.0], [5.0], color='green', label='scatter point')
plt.scatter([3.0], [2.0], color='green', label='scatter point')

plt.plot(xs0, s_0(xs0), label='s_0(x)', color='blue')
plt.plot(xs1, s_1(xs1), label='s_1(x)', color='orange')
plt.plot(xs3, s_2(xs3), label='s_1(x)', color='orange')
plt.axhline(0, color='black', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Splines cúbicos')
plt.legend()
plt.xlim(-1, 4)
plt.ylim(-2, 6)
plt.show()
```



Para cada uno de los ejercicios anteriores, resuelva los splines cúbicos de frontera condicionada con B0=1 para todos los valores de B1∈R

In [9]:
```
def cubic_spline_clamped(xs: list[float], ys: list[float], B0: float, B1: float)
    """
    Cubic spline interpolation with clamped boundary conditions.
    S'_0(x0) = B0
    S'_{n-1}(xn) = B1
```

```
    Returns symbolic expressions for each cubic polynomial.
    """

    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  # number of splines
    h = [xs[i + 1] - xs[i] for i in range(n)]


    alpha = [0] * (n + 1)


    alpha[0] = 3 * ((ys[1] - ys[0]) / h[0] - B0)

    # Interior points
    for i in range(1, n):
        alpha[i] = 3 * ((ys[i + 1] - ys[i]) / h[i] - (ys[i] - ys[i - 1]) / h[i -

    # Last boundary
    alpha[n] = 3 * (B1 - (ys[n] - ys[n - 1]) / h[n - 1])


    l = [2 * h[0]]
    u = [h[0] / l[0]]
    z = [alpha[0] / l[0]]

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

    l.append(2 * h[n - 1] - h[n - 1] * u[n - 1])
    z.append((alpha[n] - h[n - 1] * z[n - 1]) / l[n])

    c = [0] * (n + 1)
    b = [0] * n
    d = [0] * n


    c[n] = z[n]

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


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

    for j in range(n):
        a = ys[j]
        S = a + b[j] * (x - xs[j]) + c[j] * (x - xs[j])**2 + d[j] * (x - xs[j])*
        print(f"S_{j}(x): a={a}, b={b[j]}, c={c[j]}, d={d[j]}")
        splines.append(S)
```

```
    return splines
```

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

In [10]:

```
%matplotlib notebook
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation



xs = [0, 1, 2]
ys = [1, 5, 3]
B0 = 1
B1_values = list(range(20))   # 0,1,2,3,4

all_splines = []

for B1 in B1_values:
    spl = cubic_spline_clamped(xs, ys, B0=B0, B1=B1)
    all_splines.append(spl)

# ============================
# 2. Gráfico inicial
# ============================

x = sym.Symbol("x")
xplot = np.linspace(min(xs), max(xs), 200)

fig, ax = plt.subplots()
ax.set_xlim(min(xs), max(xs))
ax.set_ylim(min(ys) - 2, max(ys) + 2)

# puntos originales
ax.scatter(xs, ys, color="black", s=50)

# líneas que se actualizarán
spline_line, = ax.plot([], [], lw=3)
tangent_line, = ax.plot([], [], '--', lw=2)
b1_text = ax.text(0.05, 0.95, "", transform=ax.transAxes)

ax.set_title("Animación de Splines ")



def update(frame):
    splines = all_splines[frame]
    B1 = B1_values[frame]

    xtotal = []
    ytotal = []


    for i, S in enumerate(splines):
        xi = xs[i]
        xf = xs[i+1]
```

```
        xp = np.linspace(xi, xf, 80)
        yp = [float(S.subs(x, val)) for val in xp]

        xtotal += list(xp)
        ytotal += list(yp)

    spline_line.set_data(xtotal, ytotal)


    x0 = xs[-1]
    y0 = ys[-1]

    tx = np.linspace(x0 - 1, x0 + 1, 40)
    ty = y0 + B1 * (tx - x0)

    tangent_line.set_data(tx, ty)
    b1_text.set_text(f"B1 = {B1}")

    return spline_line, tangent_line, b1_text


# ============================
# 4. Ejecutar animación
# ============================
anim = FuncAnimation(
    fig,
    update,
    frames=len(all_splines),
    interval=1000,
    repeat=True,
    blit=False
)
from matplotlib.animation import PillowWriter

writer = PillowWriter(fps=1)
anim.save("(0,1),(1,5),(2,3)splines.gif", writer=writer)

print("GIF guardado como splines.gif")

plt.show()
```

```
S_0(x): a=1, b=1.0, c=8.75, d=-5.75
S_1(x): a=5, b=1.25, c=-8.5, d=5.25
S_0(x): a=1, b=1.0, c=9.0, d=-6.0
S_1(x): a=5, b=1.0, c=-9.0, d=6.0
S_0(x): a=1, b=1.0, c=9.25, d=-6.25
S_1(x): a=5, b=0.75, c=-9.5, d=6.75
S_0(x): a=1, b=1.0, c=9.5, d=-6.5
S_1(x): a=5, b=0.5, c=-10.0, d=7.5
S_0(x): a=1, b=1.0, c=9.75, d=-6.75
S_1(x): a=5, b=0.25, c=-10.5, d=8.25
S_0(x): a=1, b=1.0, c=10.0, d=-7.0
S_1(x): a=5, b=0.0, c=-11.0, d=9.0
S_0(x): a=1, b=1.0, c=10.25, d=-7.25
S_1(x): a=5, b=-0.25, c=-11.5, d=9.75
S_0(x): a=1, b=1.0, c=10.5, d=-7.5
S_1(x): a=5, b=-0.5, c=-12.0, d=10.5
S_0(x): a=1, b=1.0, c=10.75, d=-7.75
S_1(x): a=5, b=-0.75, c=-12.5, d=11.25
S_0(x): a=1, b=1.0, c=11.0, d=-8.0
S_1(x): a=5, b=-1.0, c=-13.0, d=12.0
S_0(x): a=1, b=1.0, c=11.25, d=-8.25
S_1(x): a=5, b=-1.25, c=-13.5, d=12.75
S_0(x): a=1, b=1.0, c=11.5, d=-8.5
S_1(x): a=5, b=-1.5, c=-14.0, d=13.5
S_0(x): a=1, b=1.0, c=11.75, d=-8.75
S_1(x): a=5, b=-1.75, c=-14.5, d=14.25
S_0(x): a=1, b=1.0, c=12.0, d=-9.0
S_1(x): a=5, b=-2.0, c=-15.0, d=15.0
S_0(x): a=1, b=1.0, c=12.25, d=-9.25
S_1(x): a=5, b=-2.25, c=-15.5, d=15.75
S_0(x): a=1, b=1.0, c=12.5, d=-9.5
S_1(x): a=5, b=-2.5, c=-16.0, d=16.5
S_0(x): a=1, b=1.0, c=12.75, d=-9.75
S_1(x): a=5, b=-2.75, c=-16.5, d=17.25
S_0(x): a=1, b=1.0, c=13.0, d=-10.0
S_1(x): a=5, b=-3.0, c=-17.0, d=18.0
S_0(x): a=1, b=1.0, c=13.25, d=-10.25
S_1(x): a=5, b=-3.25, c=-17.5, d=18.75
S_0(x): a=1, b=1.0, c=13.5, d=-10.5
S_1(x): a=5, b=-3.5, c=-18.0, d=19.5
GIF guardado como splines.gif
```

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

In [11]:

```
%matplotlib notebook
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation



xs = [0, 1, 2]
ys = [-5, -4, 3]
B0 = 1
B1_values = list(range(20))   # 0,1,2,3,4

all_splines = []
```

```python
for B1 in B1_values:
    spl = cubic_spline_clamped(xs, ys, B0=B0, B1=B1)
    all_splines.append(spl)

# ============================
# 2. Gráfico inicial
# ============================

x = sym.Symbol("x")
xplot = np.linspace(min(xs), max(xs), 200)

fig, ax = plt.subplots()
ax.set_xlim(min(xs), max(xs))
ax.set_ylim(min(ys) - 2, max(ys) + 2)

# puntos originales
ax.scatter(xs, ys, color="black", s=50)

# líneas que se actualizarán
spline_line, = ax.plot([], [], lw=3)
tangent_line, = ax.plot([], [], '--', lw=2)
b1_text = ax.text(0.05, 0.95, "", transform=ax.transAxes)

ax.set_title("Animación de Splines ")



def update(frame):
    splines = all_splines[frame]
    B1 = B1_values[frame]

    xtotal = []
    ytotal = []


    for i, S in enumerate(splines):
        xi = xs[i]
        xf = xs[i+1]


        xp = np.linspace(xi, xf, 80)
        yp = [float(S.subs(x, val)) for val in xp]

        xtotal += list(xp)
        ytotal += list(yp)

    spline_line.set_data(xtotal, ytotal)


    x0 = xs[-1]
    y0 = ys[-1]

    tx = np.linspace(x0 - 1, x0 + 1, 40)
    ty = y0 + B1 * (tx - x0)

    tangent_line.set_data(tx, ty)
    b1_text.set_text(f"B1 = {B1}")

    return spline_line, tangent_line, b1_text
```

```python
# ============================
# 4. Ejecutar animación
# ============================
anim = FuncAnimation(
    fig,
    update,
    frames=len(all_splines),
    interval=1000,
    repeat=True,
    blit=False
)
from matplotlib.animation import PillowWriter

writer = PillowWriter(fps=1)
anim.save("(0,-5),(1,-4),(2,3)splines.gif", writer=writer)

print("GIF guardado como splines.gif")

plt.show()
```

```
S_0(x): a=-5, b=1.0, c=-4.75, d=4.75
S_1(x): a=-4, b=5.749999999999999, c=9.5, d=-8.25
S_0(x): a=-5, b=1.0, c=-4.5, d=4.5
S_1(x): a=-4, b=5.499999999999999, c=9.0, d=-7.5
S_0(x): a=-5, b=1.0, c=-4.25, d=4.25
S_1(x): a=-4, b=5.249999999999999, c=8.5, d=-6.75
S_0(x): a=-5, b=1.0, c=-4.0, d=4.0
S_1(x): a=-4, b=5.0, c=8.0, d=-6.0
S_0(x): a=-5, b=1.0, c=-3.75, d=3.75
S_1(x): a=-4, b=4.75, c=7.5, d=-5.249999999999999
S_0(x): a=-5, b=1.0, c=-3.5, d=3.5
S_1(x): a=-4, b=4.5, c=7.0, d=-4.5
S_0(x): a=-5, b=1.0, c=-3.25, d=3.25
S_1(x): a=-4, b=4.25, c=6.5, d=-3.75
S_0(x): a=-5, b=1.0, c=-3.0, d=3.0
S_1(x): a=-4, b=4.0, c=6.0, d=-3.0
S_0(x): a=-5, b=1.0, c=-2.75, d=2.75
S_1(x): a=-4, b=3.75, c=5.5, d=-2.25
S_0(x): a=-5, b=1.0, c=-2.5000000000000004, d=2.5000000000000004
S_1(x): a=-4, b=3.4999999999999996, c=5.000000000000001, d=-1.5000000000000002
S_0(x): a=-5, b=1.0, c=-2.2500000000000004, d=2.2500000000000004
S_1(x): a=-4, b=3.2499999999999996, c=4.500000000000001, d=-0.7500000000000004
S_0(x): a=-5, b=1.0, c=-2.0000000000000004, d=2.0000000000000004
S_1(x): a=-4, b=2.999999999999999, c=4.000000000000001, d=-4.440892098500626e-16
S_0(x): a=-5, b=1.0, c=-1.7500000000000002, d=1.7500000000000002
S_1(x): a=-4, b=2.75, c=3.5000000000000004, d=0.7499999999999999
S_0(x): a=-5, b=1.0, c=-1.5000000000000002, d=1.5000000000000002
S_1(x): a=-4, b=2.5, c=3.0000000000000004, d=1.5
S_0(x): a=-5, b=1.0, c=-1.2500000000000002, d=1.2500000000000002
S_1(x): a=-4, b=2.25, c=2.5000000000000004, d=2.25
S_0(x): a=-5, b=1.0, c=-1.0000000000000002, d=1.0000000000000002
S_1(x): a=-4, b=2.0, c=2.0000000000000004, d=3.0
S_0(x): a=-5, b=1.0, c=-0.7500000000000002, d=0.7500000000000003
S_1(x): a=-4, b=1.75, c=1.5000000000000004, d=3.75
S_0(x): a=-5, b=1.0, c=-0.5000000000000004, d=0.5000000000000004
S_1(x): a=-4, b=1.5, c=1.0000000000000009, d=4.5
S_0(x): a=-5, b=1.0, c=-0.25000000000000044, d=0.25000000000000044
S_1(x): a=-4, b=1.25, c=0.5000000000000009, d=5.25
S_0(x): a=-5, b=1.0, c=-4.440892098500626e-16, d=4.440892098500626e-16
S_1(x): a=-4, b=1.0, c=8.881784197001252e-16, d=6.0

GIF guardado como splines.gif
```

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

In [12]:

```
%matplotlib notebook
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation



xs = [0, 1, 2,3]
ys = [-1, 1, 5,2]
B0 = 1
B1_values = list(range(20))   # 0,1,2,3,4

all_splines = []
```

```python
for B1 in B1_values:
    spl = cubic_spline_clamped(xs, ys, B0=B0, B1=B1)
    all_splines.append(spl)

# ============================
# 2. Gráfico inicial
# ============================

x = sym.Symbol("x")
xplot = np.linspace(min(xs), max(xs), 200)

fig, ax = plt.subplots()
ax.set_xlim(min(xs), max(xs))
ax.set_ylim(min(ys) - 2, max(ys) + 2)

# puntos originales
ax.scatter(xs, ys, color="black", s=50)

# líneas que se actualizarán
spline_line, = ax.plot([], [], lw=3)
tangent_line, = ax.plot([], [], '--', lw=2)
b1_text = ax.text(0.05, 0.95, "", transform=ax.transAxes)

ax.set_title("Animación de Splines ")



def update(frame):
    splines = all_splines[frame]
    B1 = B1_values[frame]

    xtotal = []
    ytotal = []


    for i, S in enumerate(splines):
        xi = xs[i]
        xf = xs[i+1]


        xp = np.linspace(xi, xf, 80)
        yp = [float(S.subs(x, val)) for val in xp]

        xtotal += list(xp)
        ytotal += list(yp)

    spline_line.set_data(xtotal, ytotal)


    x0 = xs[-1]
    y0 = ys[-1]

    tx = np.linspace(x0 - 1, x0 + 1, 40)
    ty = y0 + B1 * (tx - x0)

    tangent_line.set_data(tx, ty)
    b1_text.set_text(f"B1 = {B1}")

    return spline_line, tangent_line, b1_text
```

```python
# ============================
# 4. Ejecutar animación
# ============================
anim = FuncAnimation(
    fig,
    update,
    frames=len(all_splines),
    interval=1000,
    repeat=True,
    blit=False
)
from matplotlib.animation import PillowWriter

writer = PillowWriter(fps=1)
anim.save("(0,-1),(1,1),(2,5),(3,2)splines.gif", writer=writer)

print("GIF guardado como splines.gif")

plt.show()
```

S_0(x): a=-1, b=1.0, c=-0.33333333333333304, d=1.333333333333333
S_1(x): a=1, b=4.333333333333333, c=3.666666666666666, d=-3.9999999999999996
S_2(x): a=5, b=-0.3333333333333339, c=-8.333333333333332, d=5.666666666666667
S_0(x): a=-1, b=1.0, c=-0.3999999999999999, d=1.3999999999999997
S_1(x): a=1, b=4.3999999999999995, c=3.8, d=-4.199999999999999
S_2(x): a=5, b=-0.600000000000001, c=-8.799999999999999, d=6.3999999999999995
S_0(x): a=-1, b=1.0, c=-0.46666666666666634, d=1.4666666666666661
S_1(x): a=1, b=4.466666666666667, c=3.9333333333333327, d=-4.3999999999999995
S_2(x): a=5, b=-0.8666666666666671, c=-9.266666666666666, d=7.133333333333333
S_0(x): a=-1, b=1.0, c=-0.5333333333333332, d=1.5333333333333332
S_1(x): a=1, b=4.533333333333333, c=4.066666666666666, d=-4.6
S_2(x): a=5, b=-1.1333333333333335, c=-9.733333333333333, d=7.866666666666666
S_0(x): a=-1, b=1.0, c=-0.5999999999999996, d=1.5999999999999996
S_1(x): a=1, b=4.6000000000000005, c=4.199999999999999, d=-4.8
S_2(x): a=5, b=-1.4000000000000004, c=-10.2, d=8.6
S_0(x): a=-1, b=1.0, c=-0.6666666666666661, d=1.666666666666666
S_1(x): a=1, b=4.666666666666667, c=4.333333333333332, d=-4.999999999999999
S_2(x): a=5, b=-1.6666666666666679, c=-10.666666666666664, d=9.333333333333332
S_0(x): a=-1, b=1.0, c=-0.7333333333333334, d=1.7333333333333334
S_1(x): a=1, b=4.733333333333333, c=4.466666666666667, d=-5.2
S_2(x): a=5, b=-1.9333333333333336, c=-11.133333333333333, d=10.066666666666666
S_0(x): a=-1, b=1.0, c=-0.7999999999999998, d=1.7999999999999998
S_1(x): a=1, b=4.8, c=4.6, d=-5.399999999999999
S_2(x): a=5, b=-2.2000000000000015, c=-11.599999999999998, d=10.799999999999999
S_0(x): a=-1, b=1.0, c=-0.8666666666666663, d=1.8666666666666663
S_1(x): a=1, b=4.866666666666666, c=4.7333333333333325, d=-5.599999999999999
S_2(x): a=5, b=-2.4666666666666672, c=-12.066666666666665, d=11.533333333333331
S_0(x): a=-1, b=1.0, c=-0.9333333333333331, d=1.933333333333333
S_1(x): a=1, b=4.933333333333333, c=4.866666666666666, d=-5.8
S_2(x): a=5, b=-2.7333333333333343, c=-12.533333333333331, d=12.266666666666666
S_0(x): a=-1, b=1.0, c=-1.0, d=2.0
S_1(x): a=1, b=5.0, c=5.0, d=-6.0
S_2(x): a=5, b=-3.0, c=-13.0, d=13.0
S_0(x): a=-1, b=1.0, c=-1.0666666666666664, d=2.0666666666666664
S_1(x): a=1, b=5.066666666666666, c=5.133333333333333, d=-6.199999999999999
S_2(x): a=5, b=-3.266666666666667, c=-13.466666666666665, d=13.733333333333333
S_0(x): a=-1, b=1.0, c=-1.1333333333333329, d=2.133333333333333
S_1(x): a=1, b=5.133333333333334, c=5.266666666666666, d=-6.399999999999999
S_2(x): a=5, b=-3.5333333333333337, c=-13.933333333333332, d=14.466666666666667
S_0(x): a=-1, b=1.0, c=-1.1999999999999997, d=2.1999999999999997
S_1(x): a=1, b=5.2, c=5.3999999999999995, d=-6.599999999999999
S_2(x): a=5, b=-3.8000000000000007, c=-14.399999999999999, d=15.199999999999998
S_0(x): a=-1, b=1.0, c=-1.2666666666666662, d=2.266666666666666
S_1(x): a=1, b=5.266666666666667, c=5.533333333333332, d=-6.799999999999998
S_2(x): a=5, b=-4.066666666666667, c=-14.866666666666664, d=15.933333333333332
S_0(x): a=-1, b=1.0, c=-1.333333333333333, d=2.333333333333333
S_1(x): a=1, b=5.333333333333333, c=5.666666666666666, d=-7.0
S_2(x): a=5, b=-4.333333333333333, c=-15.333333333333332, d=16.666666666666668
S_0(x): a=-1, b=1.0, c=-1.3999999999999995, d=2.3999999999999995
S_1(x): a=1, b=5.3999999999999995, c=5.799999999999999, d=-7.199999999999998
S_2(x): a=5, b=-4.600000000000001, c=-15.799999999999997, d=17.4
S_0(x): a=-1, b=1.0, c=-1.4666666666666663, d=2.4666666666666663
S_1(x): a=1, b=5.466666666666667, c=5.933333333333333, d=-7.3999999999999995
S_2(x): a=5, b=-4.866666666666667, c=-16.266666666666666, d=18.133333333333333
S_0(x): a=-1, b=1.0, c=-1.5333333333333328, d=2.5333333333333328
S_1(x): a=1, b=5.533333333333333, c=6.066666666666655, d=-7.599999999999999
S_2(x): a=5, b=-5.133333333333335, c=-16.73333333333333, d=18.866666666666664
S_0(x): a=-1, b=1.0, c=-1.5999999999999996, d=2.5999999999999996
S_1(x): a=1, b=5.6000000000000005, c=6.199999999999999, d=-7.8
S_2(x): a=5, b=-5.4, c=-17.2, d=19.599999999999998

```
GIF guardado como splines.gif
```

Realice una animación al mover el punto x1,y1

In [ ]:

```python
import sympy as sym
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation, PillowWriter

# ======================================================
# Tu función original
# ======================================================
def cubic_spline_clamped(xs: list[float], ys: list[float], B0: float, B1: float)
    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)]

    alpha = [0] * (n + 1)
    alpha[0] = 3 * ((ys[1] - ys[0]) / h[0] - B0)

    for i in range(1, n):
        alpha[i] = 3 * ((ys[i + 1] - ys[i]) / h[i] -
                        (ys[i] - ys[i - 1]) / h[i - 1])

    alpha[n] = 3 * (B1 - (ys[n] - ys[n - 1]) / h[n - 1])

    l = [2 * h[0]]
    u = [h[0] / l[0]]
    z = [alpha[0] / l[0]]

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

    l.append(2 * h[n - 1] - h[n - 1] * u[n - 1])
    z.append((alpha[n] - h[n - 1] * z[n - 1]) / l[n])

    c = [0] * (n + 1)
    b = [0] * n
    d = [0] * n

    c[n] = z[n]

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

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

    for j in range(n):
        a = ys[j]
        S = a + b[j] * (x - xs[j]) + c[j] * (x - xs[j])**2 + d[j] * (x - xs[j])*
        splines.append(S)
```

```
    return splines


# ======================================================
# Animación modificando (x1,y1)
# ======================================================

# Puntos iniciales
xs = [0, 1, 2]
ys = [1, 5, 3]

# Movimiento del punto 1
frames = 60
x1_values = np.linspace(0.3, 1.7, frames)
y1_values = np.linspace(2, 6, frames)

fig, ax = plt.subplots(figsize=(6, 4))

def update(frame):
    ax.clear()

    # Actualizar el punto del medio
    xs[1] = x1_values[frame]
    ys[1] = y1_values[frame]

    # Recalcular spline usando tu función
    spl = cubic_spline_clamped(xs, ys, B0=1, B1=0)

    # Graficar cada spline
    x = sym.Symbol("x")
    for i, poly in enumerate(spl):
        xp = np.linspace(xs[i], xs[i+1], 200)
        fp = [poly.subs(x, val) for val in xp]
        ax.plot(xp, fp, linewidth=2)

    # Dibujar los puntos
    ax.scatter(xs, ys, s=60, color="red")

    # Dibujar recta tangente en extremo derecho
    B1 = 0
    x_last = xs[-1]
    y_last = ys[-1]
    tangent_x = np.linspace(x_last - 0.8, x_last + 0.8, 20)
    tangent_y = y_last + B1 * (tangent_x - x_last)
    ax.plot(tangent_x, tangent_y, linestyle="--", color="black")

    # Texto con coordenadas
    ax.text(
        0.05, 0.95,
        f"Frame {frame+1}\nx1={xs[1]:.2f}, y1={ys[1]:.2f}",
        transform=ax.transAxes,
        fontsize=10,
        verticalalignment="top"
    )

    ax.set_xlim(-0.5, 2.5)
    ax.set_ylim(-1, 7)
    ax.set_title("Animación del spline moviendo (x1, y1)")
    ax.grid(True)
```

```
anim = FuncAnimation(
    fig,
    update,
    frames=frames,
    interval=150,
    repeat=True
)

# Guardar GIF
anim.save("spline_moviendo_punto.gif", writer=PillowWriter(fps=20))

plt.show()
```