Tabla de Contenidos

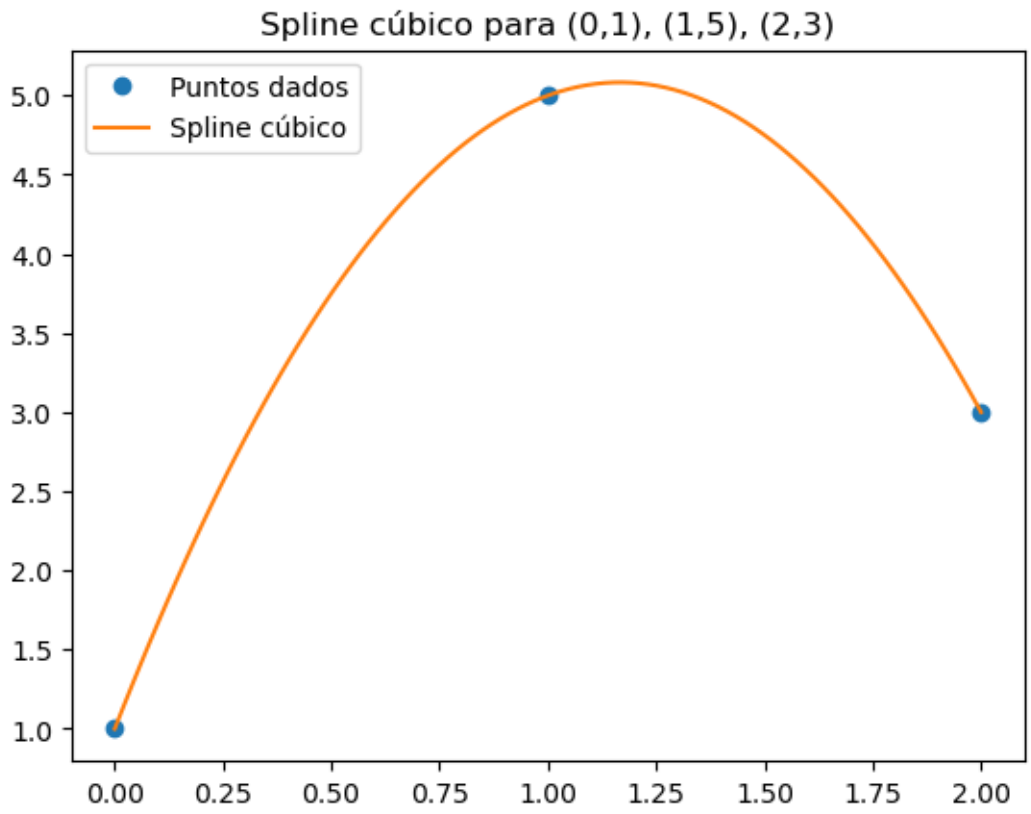
ESCUELA POLITÉCNICA NACIONAL	Tarea No.
Metodos Numericos – Computación	**7**
NOMBRE: Ivonne Carolina Ayala	

[Tarea 07] Unidad 03-A | Splines

Conjunto de Ejercicios

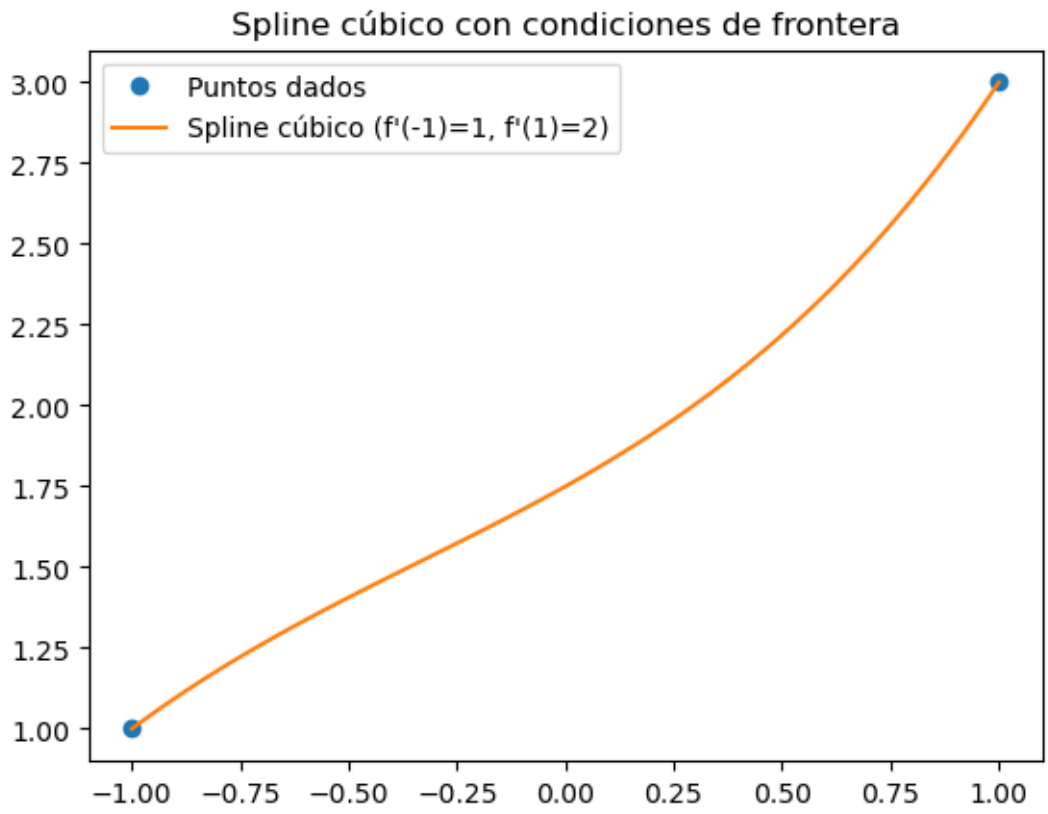
1. Dados los puntos (0,1), (1,5), (2,3), determine el spline cúbico.

```
import numpy as np
from scipy.interpolate import CubicSpline
import matplotlib.pyplot as plt

x1 = np.array([0, 1, 2])
y1 = np.array([1, 5, 3])

spline1 = CubicSpline(x1, y1)

x_dense1 = np.linspace(0, 2, 100)
y_dense1 = spline1(x_dense1)

plt.plot(x1, y1, 'o', label='Puntos dados')
plt.plot(x_dense1, y_dense1, label='Spline cúbico')
plt.legend()
plt.title('Spline cúbico para (0,1), (1,5), (2,3)')
plt.show()
```



2. Dados los puntos (-1,1), (1,3), determine el spline cúbico sabiendo que (f'(x_0) = 1), (f'(x_n) = 2).

```
x2 = np.array([-1, 1])
y2 = np.array([1, 3])

bc_type = ((1, 1), (1, 2))

spline2 = CubicSpline(x2, y2, bc_type=bc_type)

x_dense2 = np.linspace(-1, 1, 100)
y_dense2 = spline2(x_dense2)

plt.plot(x2, y2, 'o', label='Puntos dados')
plt.plot(x_dense2, y_dense2, label='Spline cúbico (f\'(-1)=1, f\'(1)=2)')
plt.legend()
plt.title('Spline cúbico con condiciones de frontera')
plt.show()
```



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

Codigo proporcionado en git completado

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

def cubic_spline(xs: list[float], ys: list[float]) -> list[sym.Expr]:
    """
    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 + 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 = [1] + [0] * n
    u = [0] * n
    z = [0] * (n + 1)

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

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

    x = sym.Symbol("x")
    splines = []
    b = [0] * n
    d = [0] * n
    a = ys

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

```
    splines.reverse()
    return splines
```

Codigo proporcionado en git

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


# ######################################################################
def cubic_spline_clamped(
    xs: list[float], ys: list[float], d0: float, dn: 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
    - d0, dn: 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 * d0
    alpha[-1] = 3 * dn - 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
```

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

```
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]
```

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

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

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

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

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

```
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]
```

$1.0x^3 + 1.0x - 1$

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

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

$1.0x^3 + 1.0x - 1$

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

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

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

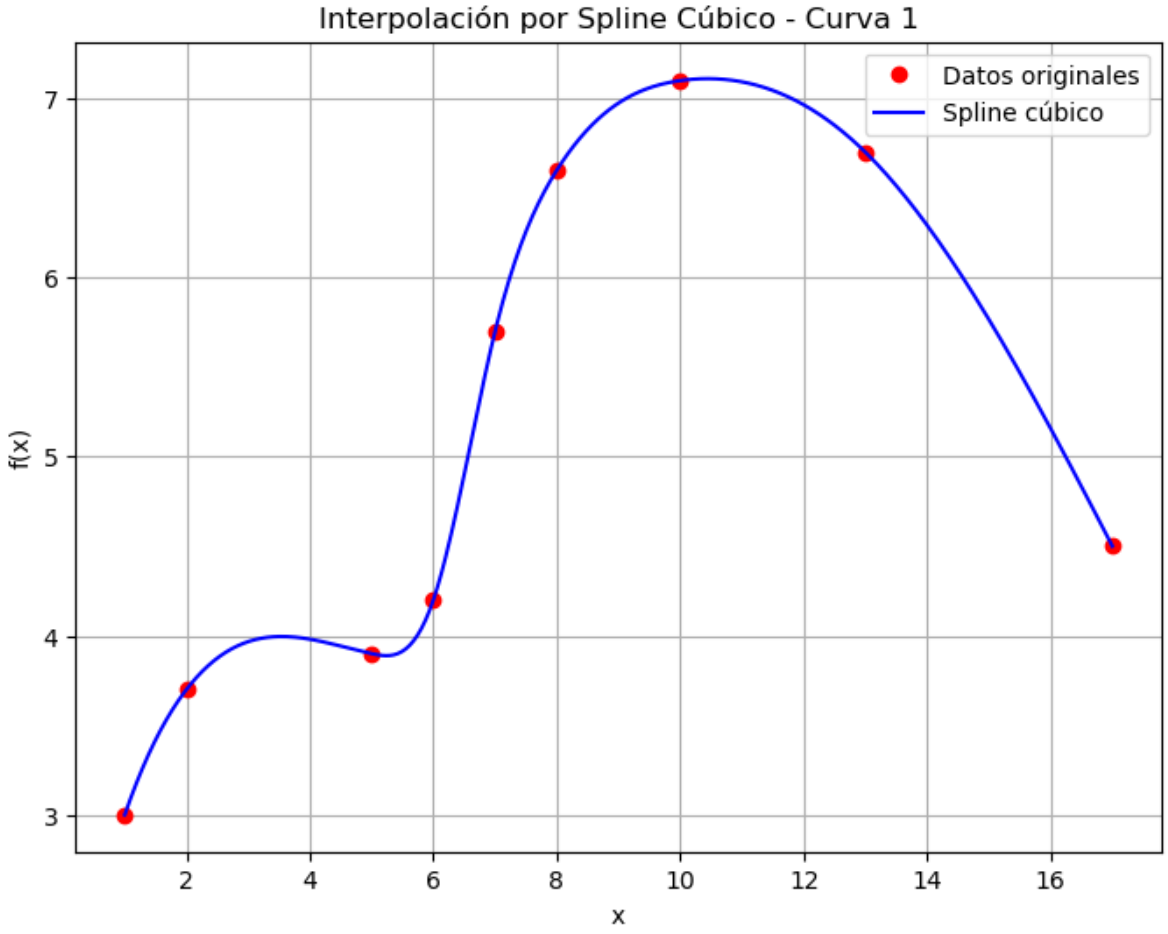
	Curva 1				Curva 2				Curva 3		
i	x_i	$f(x_i)$	$f'(x_i)$	i	x_i	$f(x_i)$	$f'(x_i)$	i	x_i	$f(x_i)$	$f'(x_i)$
0	1	3.0	1.0	0	17	4.5	3.0	0	27.7	4.1	0.33
1	2	3.7		1	20	7.0		1	28	4.3	
2	5	3.9		2	23	6.1		2	29	4.1	
3	6	4.2		3	24	5.6		3	30	3.0	−1.5
4	7	5.7		4	25	5.8					
5	8	6.6		5	27	5.2					
6	10	7.1		6	27.7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

Figura 1: TablaEjercicio6

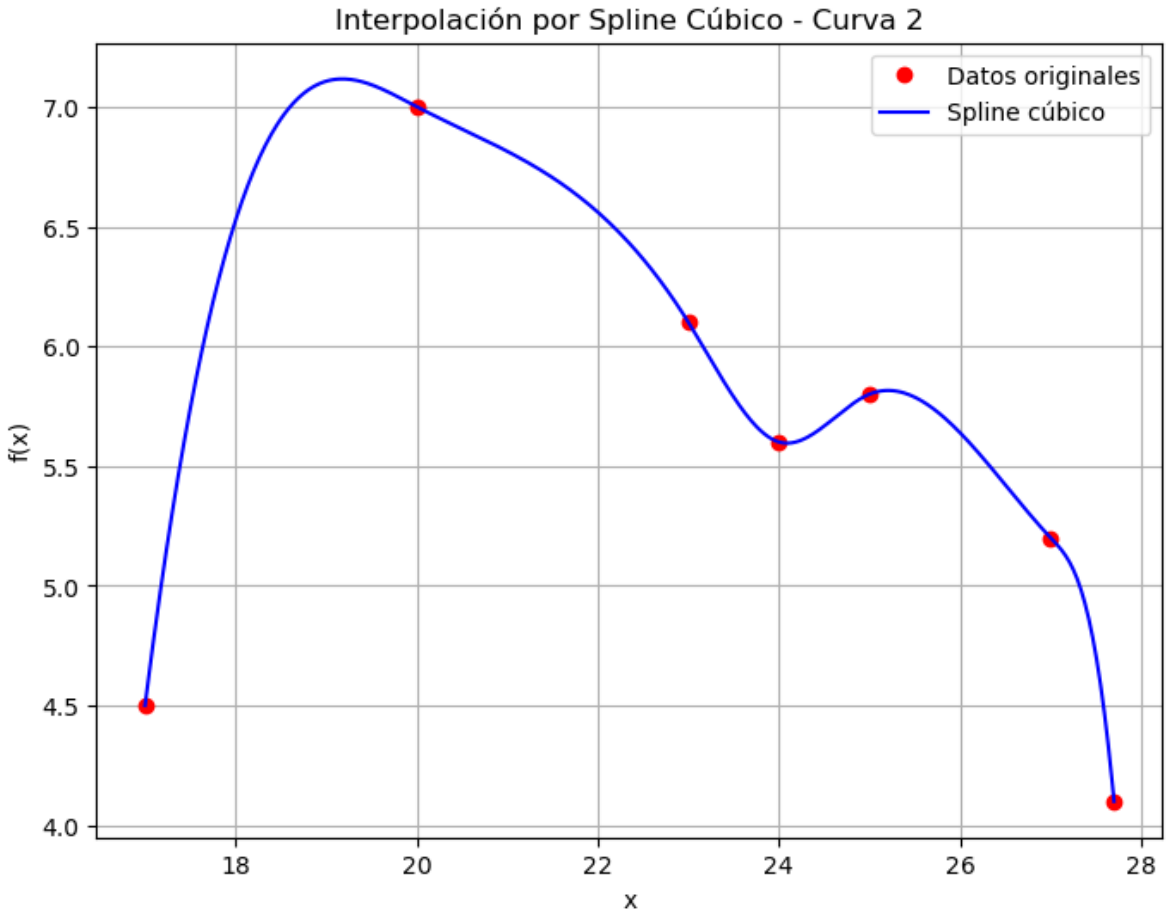
```
curves = {
    "Curva 1": {
        "xs": [1, 2, 5, 6, 7, 8, 10, 13, 17],
        "ys": [3.0, 3.7, 3.9, 4.2, 5.7, 6.6, 7.1, 6.7, 4.5],
        "d0": 1.0,
        "dn": -0.67,
    },
    "Curva 2": {
        "xs": [17, 20, 23, 24, 25, 27, 27.7],
        "ys": [4.5, 7.0, 6.1, 5.6, 5.8, 5.2, 4.1],
        "d0": 3.0,
        "dn": -4.0,
    },
    "Curva 3": {
        "xs": [27.7, 28, 29, 30],
        "ys": [4.1, 4.3, 4.1, 3.0],
        "d0": 0.33,
        "dn": -1.5,
    },
}

for curve_name, data in curves.items():
    xs = data["xs"]
    ys = data["ys"]
    d0 = data["d0"]
    dn = data["dn"]
    splines = cubic_spline_clamped(xs, ys, d0, dn)

    x_vals = np.linspace(min(xs), max(xs), 500)
    y_vals = []

    for x_val in x_vals:
        for i in range(len(xs) - 1):
            if xs[i] <= x_val <= xs[i + 1]:
                y_vals.append(sym.lambdify(sym.Symbol("x"), splines[i])(x_val))
                break

    plt.figure(figsize=(8, 6))
    plt.plot(xs, ys, "o", label="Datos originales", color="red")
    plt.plot(x_vals, y_vals, label="Spline cúbico", color="blue")
    plt.title(f"Interpolación por Spline Cúbico - {curve_name}")
    plt.xlabel("x")
```

```
    plt.ylabel("f(x)")
    plt.legend()
    plt.grid(True)
    plt.show()
```

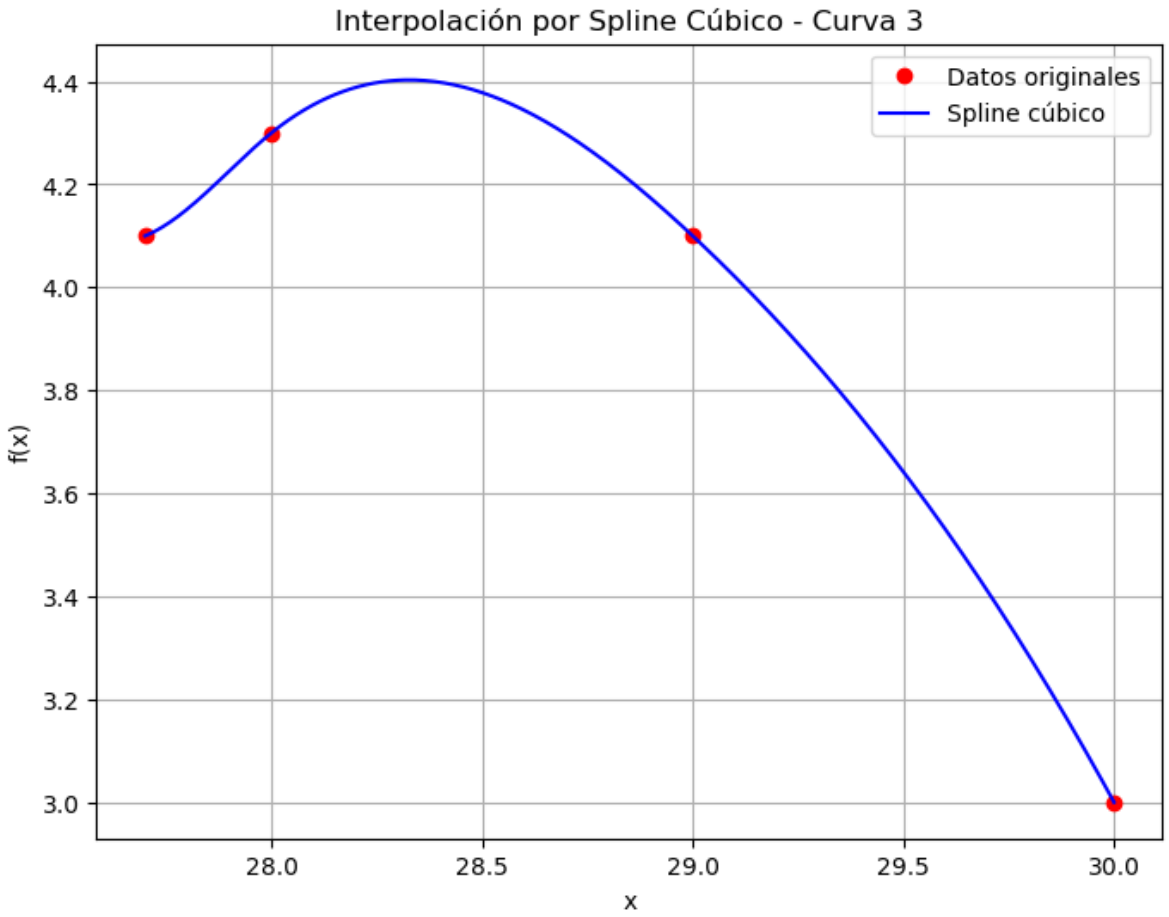
```
7 6.7 -0.3381314976116886 -0.07593425119415571 0.0057417813992694635
6 7.1 0.04846024164091059 -0.052929661890044014 -0.0025560654782346335
5 6.6 0.5472201929380908 -0.19645031375854607 0.023920108644750342
4 5.7 1.4091093003652708 -0.665438793668634 0.15632949330336265
3 4.2 1.0163426056008245 1.0582054884330803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



```
5 5.2 -0.4011781849199465 0.1258152222202451 -2.568002126658778
4 5.8 0.1539868142803838 -0.4033977218204103 0.08820215734010924
3 5.6 -0.11137135038117751 0.6687558864819717 -0.35738453610079396
2 6.1 -0.6085014127556733 -0.17162582410747595 0.2801272368631492
1 7.0 -0.19787464681108174 0.03475023545927881 -0.022930673285194974
0 4.5 3.0 -1.1007084510629728 0.12616207628025017
```




```
2 4.1 -0.7653465346534649 -0.26930693069306927 -0.06534653465346556
1 4.3 0.6613861386138599 -1.1574257425742556 0.2960396039603954
0 4.1 0.3299999999999999 2.2620462046204524 -3.799413274660778
```


Interpolación por Spline Cúbico - Curva 3
Datos originales
Spline cúbico
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
f(x)
28.0
28.5
29.0
29.5
30.0
x