Tarea 6

Danny Iñaguazo

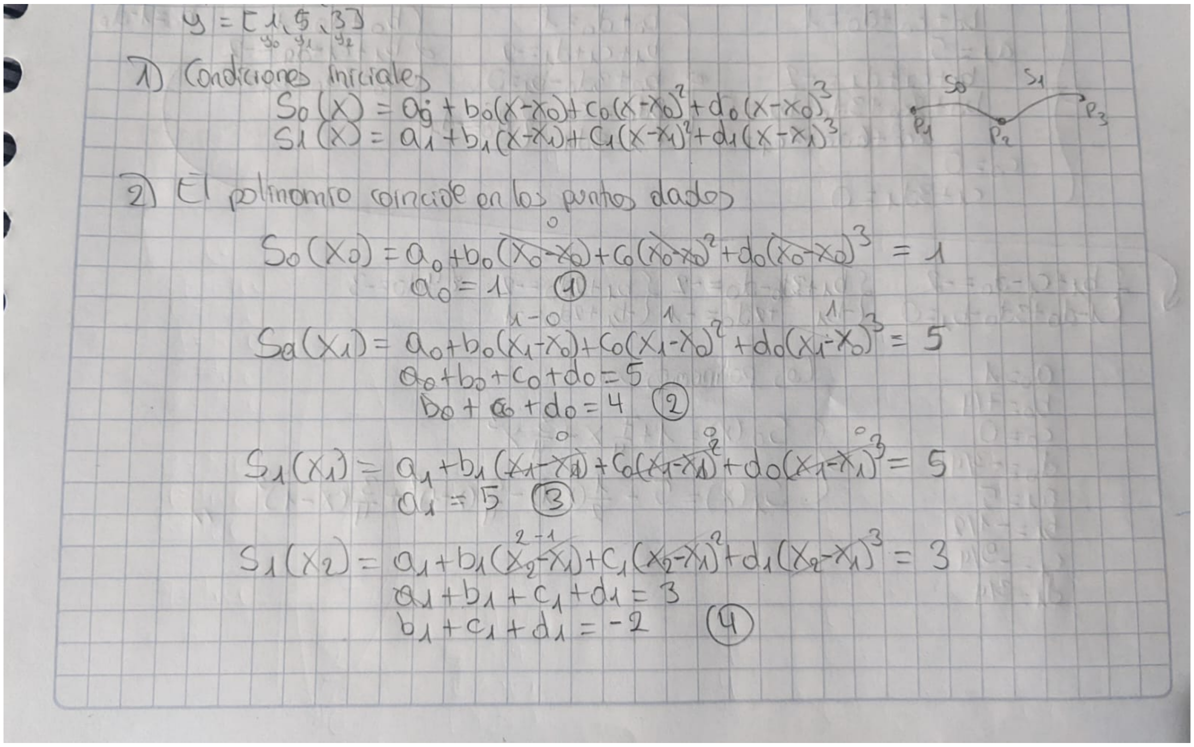
Tabla de Contenidos

Conjunto de ejercicios

Ejercicio 1

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

Las siguientes imágenes muestran la solución y el procedimiento del ejercicio planteado:



$y = [1, 5, 3]$ (y_0, y_1, y_2)

1) Condiciones iniciales

$$S_0(x) = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3$$
$$S_1(x) = a_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3$$

2) El polinomio coincide en los puntos dados

$$S_0(x_0) = a_0 + b_0\overset{0}{(\cancel{x_0-x_0})} + c_0(\cancel{x_0-x_0})^2 + d_0(\cancel{x_0-x_0})^3 = 1$$
$$a_0 = 1 \quad (1)$$

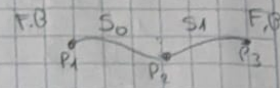
$$S_0(x_1) = a_0 + b_0\overset{1-0}{(x_1-x_0)} + c_0\overset{1}{(x_1-x_0)}^2 + d_0\overset{1}{(x_1-x_0)}^3 = 5$$
$$a_0 + b_0 + c_0 + d_0 = 5$$
$$b_0 + c_0 + d_0 = 4 \quad (2)$$

$$S_1(x_1) = a_1 + b_1\overset{0}{(\cancel{x_1-x_1})} + c_0(\cancel{x_1-x_1})^2 + d_0(\cancel{x_1-x_1})^3 = 5$$
$$a_1 = 5 \quad (3)$$

$$S_1(x_2) = a_1 + b_1\overset{2-1}{(x_2-x_1)} + c_1(x_2-x_1)^2 + d_1(x_2-x_1)^3 = 3$$
$$a_1 + b_1 + c_1 + d_1 = 3$$
$$b_1 + c_1 + d_1 = -2 \quad (4)$$

Figura 1: Ejercicio 1: Resolución

3) E1/Los polinomio/s son iguales en su f' y f''



$f' \to \quad S'_j(x) = b_j + 2c_j(x-x_j) + 3(x-x_j)^2 d_j$

$f'' \to \quad S''_j(x) = 2c_j + 6(x-x_j)d_j$

$$S'_0(x_1) = S'_1(x_1)$$
$$b_0 + 2c_0(x_1-x_0) + 3d_0(x_1-x_0)^2 = b_1 + 2c_1\cancel{(x_1-x_1)} + 3d_1\cancel{(x_1-x_1)}^2$$
$$b_0+2c_0+3d_0 = b_1$$
$$b_0+2c_0+3d_0-b_1 = 0 \quad (5)$$

$$S''_0(x_1) = S''_1(x_1)$$
$$2c_0+6d_0(x_1-x_0) = 2c_1+6d_1(x_1-x_1)$$
$$2c_0+6d_0 = 2c_1$$
$$2c_0+6d_0-2c_1 = 0 \quad (6)$$

4) Frontera de banera: La segunda derivada es igual a cero

$$S''_0(x_0) = 0$$
$$2c_0+6d_0\cancel{(x_0-x_0)} = 0$$
$$c_0 = 0 \quad (7)$$

$$S''_1(x_2) = 0$$
$$2c_1+6d_1\cancel{(x_2-x_1)} = 0$$
$$c_1+3d_1 = 0 \quad (8)$$

$$\left\{\begin{aligned} &a_0 = 1\\ &b_0+c_0+d_0 = 1\\ &a_1 = 5\\ &b_1+c_1+d_1 = -2\\ &b_0+2c_0+3d_0-b_1 = 0\\ &2c_0+6d_0-2c_1 = 0\\ &c_0 = 0\\ &c_1+3d_1 = 0 \end{aligned}\right. \quad \left\{\begin{aligned} &b_0+d_0 = 1\\ &b_1+c_1+d_1 = -2\\ &b_0+3d_0-b_1 = 0\\ &3d_0 = c_1\\ &c_1 = 3d_1 \end{aligned}\right. \quad \{d_0 = -d_1 \quad \left\{\begin{aligned} &b_0+d_0 = 1\\ &b_1+c_1-d_0 = -2\\ &b_0+3d_0-b_1 = 0 \end{aligned}\right. \quad b_0 = 1-d_0$$

$$\left\{\begin{aligned} &b_1+c_1-d_0 = -2\\ &1-d_0+3d_0-b_1 = 0 \end{aligned}\right. \quad \left\{\begin{aligned} &b_1+3d_0-d_0 = -2\\ &-b_1+2d_0 = -1 \end{aligned}\right. \quad \left\{\begin{aligned} &b_1+2d_0 = -2\\ &-b_1+2d_0 = -1 \end{aligned}\right. \quad \{4d_0 = -3$$

$a_0 = 1$
$b_0 = 7/4$
$c_0 = 0$
$d_0 = -3/4$
$a_1 = 5$
$b_1 = -1/2$
$c_1 = -9/4$
$d_1 = 3/4$

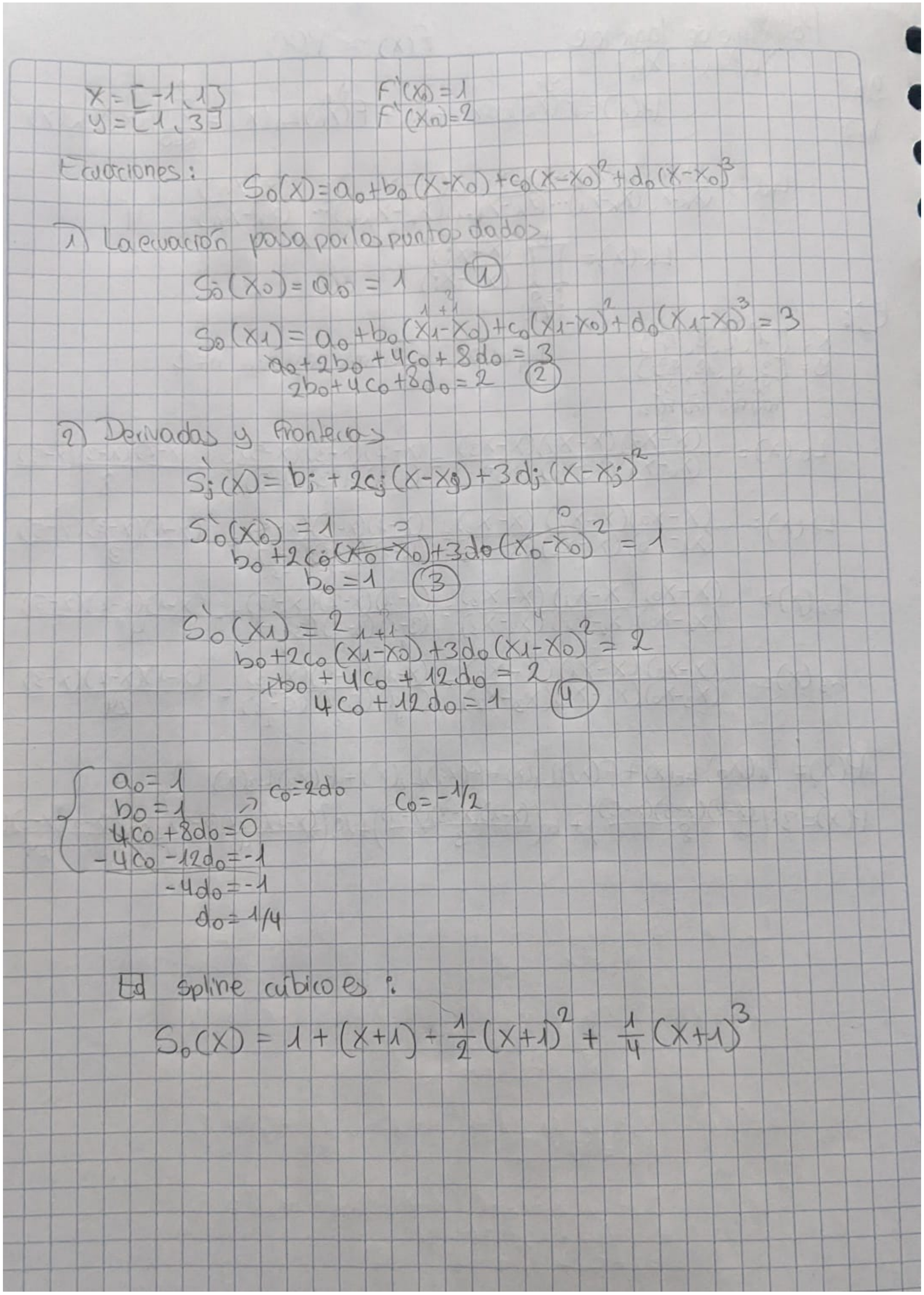
Los polinomios son:

$$\left\{\begin{aligned} &S_0(x) = 1+\frac{7}{4}x - \frac{3}{4}x^3\\ &S_1(x) = 5 - \frac{1}{2}(x-1) - \frac{9}{4}(x-1)^2 + \frac{3}{4}(x-1)^3 \end{aligned}\right. \quad //$$

Figura 2: Ejercicio 1: Resolución

Ejercicio 2

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



$X = [-1, 1]$ $\quad$ $F'(x_0) = 1$

$y = [1, 3]$ $\quad$ $F'(x_n) = 2$

Ecuaciones: $S_0(x) = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3$

1) La ecuación pasa por los puntos dados

$$S_0(x_0) = a_0 = 1 \quad (1)$$

$$S_0(x_1) = a_0 + b_0(x_1-x_0) + c_0(x_1-x_0)^2 + d_0(x_1-x_0)^3 = 3$$

$$a_0 + 2b_0 + 4c_0 + 8d_0 = 3$$

$$2b_0 + 4c_0 + 8d_0 = 2 \quad (2)$$

2) Derivadas y fronteras

$$S'_j(x) = b_j + 2c_j(x-x_j) + 3d_j(x-x_j)^2$$

$$S'_0(x_0) = 1$$

$$b_0 + 2c_0\cancel{(x_0-x_0)} + 3d_0\cancel{(x_0-x_0)^2} = 1$$

$$b_0 = 1 \quad (3)$$

$$S'_0(x_1) = 2$$

$$b_0 + 2c_0(x_1-x_0) + 3d_0(x_1-x_0)^2 = 2$$

$$\cancel{b_0} + 4c_0 + 12d_0 = 2$$

$$4c_0 + 12d_0 = 1 \quad (4)$$

$$\begin{cases} a_0 = 1 \\ b_0 = 1 \\ 4c_0 + 8d_0 = 0 \\ -4c_0 - 12d_0 = -1 \end{cases} \qquad c_0 = 2d_0 \qquad c_0 = -1/2$$

$$-4d_0 = -1$$

$$d_0 = 1/4$$

El spline cúbico es:

$$S_0(x) = 1 + (x+1) - \frac{1}{2}(x+1)^2 + \frac{1}{4}(x+1)^3$$

Figura 3: Ejercicio 2: Resolución

Ejercicio 3

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

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

```python
    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
```

```python
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
------
```

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

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

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

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

Ejercicio 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]
```

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

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

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

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

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

Ejercicio 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]
```

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

$1.0x^3 + 1.0x - 1$

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

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

$1.0x^3 + 1.0x - 1$

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

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

Ejercicio 6

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

```python
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
```

```python
    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
```

Literal a)

La resolución es la siguiente:

```python
xs = [1, 2, 5, 6, 7, 8, 10, 13, 17]
ys = [3, 3.7, 3.9, 4.2, 5.7, 6.6, 7.1, 6.7, 4.5]

splines = cubic_spline_clamped(xs=xs, ys=ys, d0=1, dn=-0.67)
_ = [display(s) for s in splines]
print("______")
_ = [display(s.expand()) for s in splines]
```

```
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 1.0 -0.3468099653460706 0.046809965346070785
______
```

$1.0x + 0.0468099653460708\,(x-1)^3 - 0.346809965346071\,(x-1)^2 + 2.0$

$0.446809965346071x + 0.0265552121382411\,(x-2)^3 - 0.206380069307858\,(x-2)^2 + 2.80638006930786$

$-0.0744797227685678x + 0.341862882832256\,(x-5)^3 + 0.032616839936312\,(x-5)^2 + 4.27239861384284$

$1.01634260560082x - 0.574548094033905\,(x-6)^3 + 1.05820548843308\,(x-6)^2 - 1.89805563360495$

$1.40910930036527x + 0.156329493303363\,(x-7)^3 - 0.665438793668634\,(x-7)^2 - 4.1637651025569$

$0.547220192938091x + 0.0239201086447503\,(x-8)^3 - 0.196450313758546\,(x-8)^2 + 2.22223845649527$

$0.0484602416409106x - 0.00255606547823463\,(x-10)^3 - 0.052929661890044\,(x-10)^2 + 6.61539758359089$

$-0.338131497611689x + 0.00574178139926946\,(x-13)^3 - 0.0759342511941557\,(x-13)^2 + 11.095709468952$

$0.0468099653460708x^3 - 0.487239861384283x^2 + 1.83404982673035x + 1.60638006930786$

$0.0265552121382411x^3 - 0.365711342137305x^2 + 1.5909927882364x + 1.7684180949705$

$0.341862882832256x^3 - 5.09532640254753x^2 + 25.2390680902875x - 37.6450407417814$

$-0.574548094033905x^3 + 11.4000711810434x^2 - 73.7333174112578x + 160.299730261309$

$0.156329493303363x^3 - 3.94835815303925x^2 + 33.7056879273205x - 90.3912821953733$

$0.0239201086447503x^3 - 0.770532921232554x^2 + 8.28308607286689x - 22.5976772501638$

$-0.00255606547823463x^3 + 0.023752302456995x^2 + 0.340233835971401x + 3.87849687282113$

$0.00574178139926946x^3 - 0.299863725765665x^2 + 4.54724220286598x - 14.3518727170554$

Literal b)

La resolución es la siguiente:

```
xs = [17, 20, 23, 24, 25, 27, 27.7]
ys = [4.5, 7, 6.1, 5.6, 5.8, 5.2, 4.1]

splines = cubic_spline_clamped(xs=xs, ys=ys, d0=3, dn=-4)
_ = [display(s) for s in splines]
print("______")
_ = [display(s.expand()) for s in splines]
```

```
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 -0.19787464681108174 0.03475023545927881 -0.022930673285194974
0 4.5 3.0 -1.1007084510629728 0.12616207628025017
______
```

$3.0x + 0.12616207628025\left(x - 17\right)^3 - 1.10070845106297\left(x - 17\right)^2 - 46.5$

$-0.197874646811082x - 0.022930673285195\left(x - 20\right)^3 + 0.0347502354592788\left(x - 20\right)^2 + 10.9574929362216$

$-0.608501412755673x + 0.280127236863149\left(x - 23\right)^3 - 0.171625824107476\left(x - 23\right)^2 + 20.0955324933805$

$-0.111371350381178x - 0.357384536100794\left(x - 24\right)^3 + 0.668755886481972\left(x - 24\right)^2 + 8.27291240914826$

$0.153986814280384x + 0.0882021573401092\left(x - 25\right)^3 - 0.40339772182041\left(x - 25\right)^2 + 1.9503296429904$

$-0.401178184919947x - 2.56800212665878\left(x - 27\right)^3 + 0.125815222220245\left(x - 27\right)^2 + 16.0318109928386$

$0.12616207628025x^3 - 7.53497434135573x^2 + 149.806607471118x - 984.439023122068$

$-0.022930673285195x^3 + 1.41059063257098x^2 - 29.1046920074162x + 208.302973401493$

$0.280127236863149x^3 - 19.5004051676648x^2 + 451.848211398006x - 3479.00261937341$

$-0.357384536100794x^3 + 26.4004424857391x^2 - 649.772132283688x + 5333.96013008014$

$0.0882021573401092x^3 - 7.0185595223286x^2 + 185.702917918006x - 1628.33195493397$

$-2.56800212665878x^3 + 208.133987481581x^2 - 5623.41585118756x + 50653.7369670161$

Literal c)

La resolución es la siguiente:

```
xs = [27.7, 28, 29, 30]
ys = [4.1, 4.3, 4.1, 3]

splines = cubic_spline_clamped(xs=xs, ys=ys, d0=0.33, dn=-1.5)
_ = [display(s) for s in splines]
print("______")
_ = [display(s.expand()) for s in splines]
```

```
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
______
```

$0.33x - 80752.4751789508\left(0.036101083032491x - 1\right)^3 + 1735.64543234323\left(0.036101083032491x - 1\right)^2 - 5.041$

$0.66138613861386x + 0.296039603960395\left(x - 28\right)^3 - 1.15742574257426\left(x - 28\right)^2 - 14.2188118811881$

$-0.765346534653465x - 0.0653465346534656\left(x - 29\right)^3 - 0.269306930693069\left(x - 29\right)^2 + 26.2950495049505$

$-3.79941327466078x^3 + 317.993289328931x^2 - 8870.74279427938x + 82483.079611294$

$0.296039603960395x^3 - 26.0247524752475x^2 + 761.762376237622x - 7420.30198019801$

$-0.0653465346534656x^3 + 5.41584158415843x^2 - 150.014851485149x + 1393.54455445545$