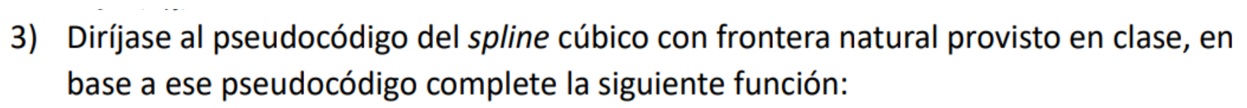
ESCUELA POLITÉCNICA NACIONAL
MÉTODOS NUMÉRICOS

Nombre: David Guachamín
Tema: Tarea 7
Gr1CC



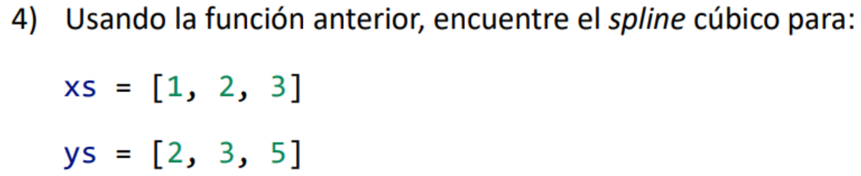
3) Diríjase al pseudocódigo del *spline* cúbico con frontera natural provisto en clase, en base a ese pseudocódigo complete la siguiente función:

```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 calculation
    alpha = [0] * (n + 1)  # Initialize alpha with zeros
    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])

    # Initialize l, u, z
    l = [1]
    u = [0]
    z = [0]

    # Forward substitution
    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(1)
    z.append(0)
    c = [0] * (n + 1)

    # Back substitution and spline coefficients
    x = sym.Symbol("x")
```

ESCUELA POLITÉCNICA NACIONAL
MÉTODOS NUMÉRICOS

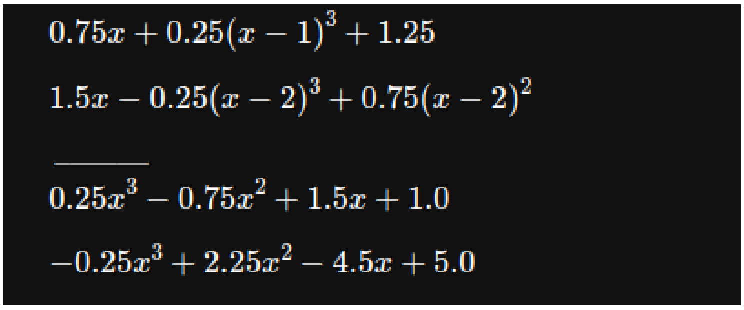
Nombre: David Guachamín
Tema: Tarea 7
Gr1CC

```
    splines = []
    b = [0] * n
    d = [0] * n
    a = [0] * 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])
        a[j] = ys[j]
        # Spline expression
        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
```



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

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

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



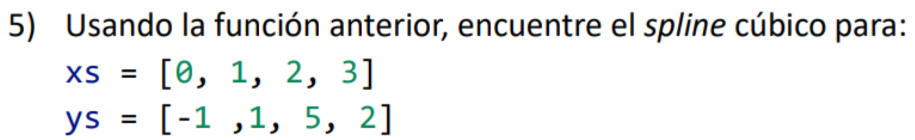
$0.75x + 0.25(x-1)^3 + 1.25$

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

______

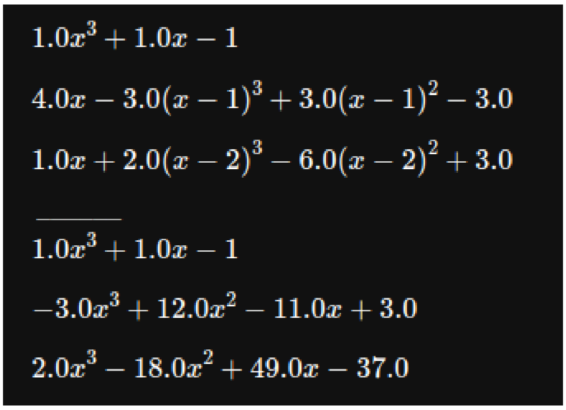
$0.25x^3 - 0.75x^2 + 1.5x + 1.0$

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



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

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



$1.0x^3 + 1.0x - 1$

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

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

______

$1.0x^3 + 1.0x - 1$

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

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

ESCUELA POLITÉCNICA NACIONAL
MÉTODOS NUMÉRICOS

Nombre: David Guachamín
Tema: Tarea 7
Gr1CC

LINK GITHUB: https://github.com/Davandres/Deberes-MN/tree/main/TAREA%207