Tarea 7 - Splines cúbicos

Nombre: Alexis Bautista
Fecha de entrega: 27 de noviembre de 2024
Paralelo: GR1CC
Enlace de GitHub: https://github.com/alexis-bautista/Tarea07-MN.git

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

In [10]:
```
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 cub
    ``S_j`` of the form ``S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(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

    # Calculate alpha
    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])
        )

    # Tridiagonal system
    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)
```

```
    # Back substitution
    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
```

In [11]:

```
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
```

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

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

______

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

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

4. Usando la función anterior, encuentre el spline cúbico para:
   $xs = [1, 2, 3]$
   $ys = [2, 3, 5]$

In [12]:

```
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
```

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

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

______

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

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