Métodos Numéricos - DEBER 07 - Splines Cúbicos

Alicia Pereira

Tabla de Contenidos

0.1 Conjunto de Ejercicios

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

```
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

    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]
    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]]
        # Completar z
        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)
        #Completar S
```

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

        splines.append(S)
    splines.reverse()
    return splines
```

Análisis de las líneas completadas en el código

`alpha = [0] * n` - **`alpha`**: Una lista de tamaño (n) (el número de intervalos) inicializada con ceros. - La expresión `[0] * n` crea una lista de (n) elementos, todos inicializados a (0). - `alpha` almacena los términos necesarios para construir el sistema tridiagonal de ecuaciones en el cálculo de los splines cúbicos. - Cada valor de ([i]) se calcula usando:

$$\alpha[i] = \frac{3}{h[i]}(y_{i+1} - y_i) - \frac{3}{h[i-1]}(y_i - y_{i-1})$$

- Estos valores representan cambios relativos en las derivadas entre puntos adyacentes.

`z += [(alpha[i] - h[i - 1] * z[i - 1]) / l[i]]` - **`z`**: Lista que almacena soluciones intermedias del sistema tridiagonal. - Esta línea añade un nuevo valor calculado a la lista `z`. - Calcula los valores intermedios necesarios para determinar las derivadas segundas ((c)) en el spline cúbico. - Se deriva de la resolución del sistema tridiagonal que surge en el cálculo de los splines cúbicos.

- Cada valor de (z[i]) se calcula usando:

$$z[i] = \frac{\alpha[i] - h[i-1] \cdot z[i-1]}{l[i]}$$

- Donde:
  - ([i]): Cambios relativos calculados previamente.
  - (h[i-1]): Distancia entre puntos consecutivos.
  - (z[i-1]): Valor calculado en la iteración previa.
  - (l[i]): Diagonal principal del sistema tridiagonal.

`a = ys[j]` - **`a`**: Constante (a_j) en el spline cúbico. - En cada intervalo, (a_j) es simplemente el valor (y_j) correspondiente al punto inicial del intervalo.

- Representa el término constante del spline cúbico (S_j(x)), asegurando que pase exactamente por el punto ((x_j, y_j)).
- Este valor se utiliza directamente en la construcción de la fórmula del spline.

`S = a + b * (x - xs[j]) + c[j] * (x - xs[j])**2 + d * (x - xs[j])**3` - `S`: Ecuación simbólica que representa el spline cúbico en un intervalo. - Se calcula como:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

- Donde: - (a_j, b_j, c_j, d_j) son los coeficientes calculados para ese intervalo. - (x) es el valor simbólico donde se evalúa la interpolación. - Define el spline cúbico (S_j(x)) en cada intervalo ([x_j, x_{j+1}]). - Este spline garantiza suavidad (continuidad en valores, derivadas primeras y segundas) entre los puntos.

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

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

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

cubic_spline(xs,ys)
```

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

```
[1.0*x**3 + 1.0*x - 1,
 4.0*x - 3.0*(x - 1)**3 + 3.0*(x - 1)**2 - 3.0,
 1.0*x + 2.0*(x - 2)**3 - 6.0*(x - 2)**2 + 3.0]
```

Spline cúbico para el intervalo [0, 1]:

$$S_0(x) = 1.0 \cdot x^3 + 1.0 \cdot x - 1$$

Spline cúbico para el intervalo [1, 2]:

$$S_1(x) = -3.0 \cdot (x - 1)^3 + 3.0 \cdot (x - 1)^2 + 4.0 \cdot x - 3.0$$

Spline cúbico para el intervalo [2, 3]:

$$S_2(x) = 2.0 \cdot (x - 2)^3 - 6.0 \cdot (x - 2)^2 + 1.0 \cdot x + 3.0$$

GitHub: git@github.com: dayapt04

GitHub Métodos Númericos - Repositorio