

## Objetivos Gerais:

O trabalho teve como objetivo principal implementar um programa em Python para realizar interpolação cúbica por splines naturais. Utilizando este método, buscamos criar polinômios cúbicos suaves que passam por um conjunto de pontos fornecidos.

## O Que Foi Feito:

Desenvolvimento de um programa em Python que implementa a interpolação cúbica por splines naturais. Este programa realiza a interpolação cúbica natural de um conjunto de pontos de entrada. Ele solicita ao usuário a quantidade de pontos de interpolação, os valores de  $x$  e  $y$  para esses pontos, um valor  $z$  para calcular a spline cúbica nesse ponto e a quantidade de pontos  $m$  para os quais as imagens das splines serão calculadas. O programa verifica se a entrada está correta, calcula os coeficientes dos polinômios cúbicos naturais e, em seguida, imprime e plota as splines resultantes. O gráfico gerado mostra os pontos de interpolação em vermelho e as splines em um intervalo específico.

## Execuções

### 1. Problema 1

Obter as splines interpoladoras dos seguintes dados:

5  
1.0      1.3      2.0      3.0      3.5  
0.5      0.2      0.8      1.7      1.3  
1.1  
26

Fazer o gráfico das splines em  $D = [a = 1.0, b = 3.5]$  usando  $m = 26$  pontos (contando com  $x_0 = 1.0$  e  $b = 3.5$ )

### Saida gerada pelo programa:

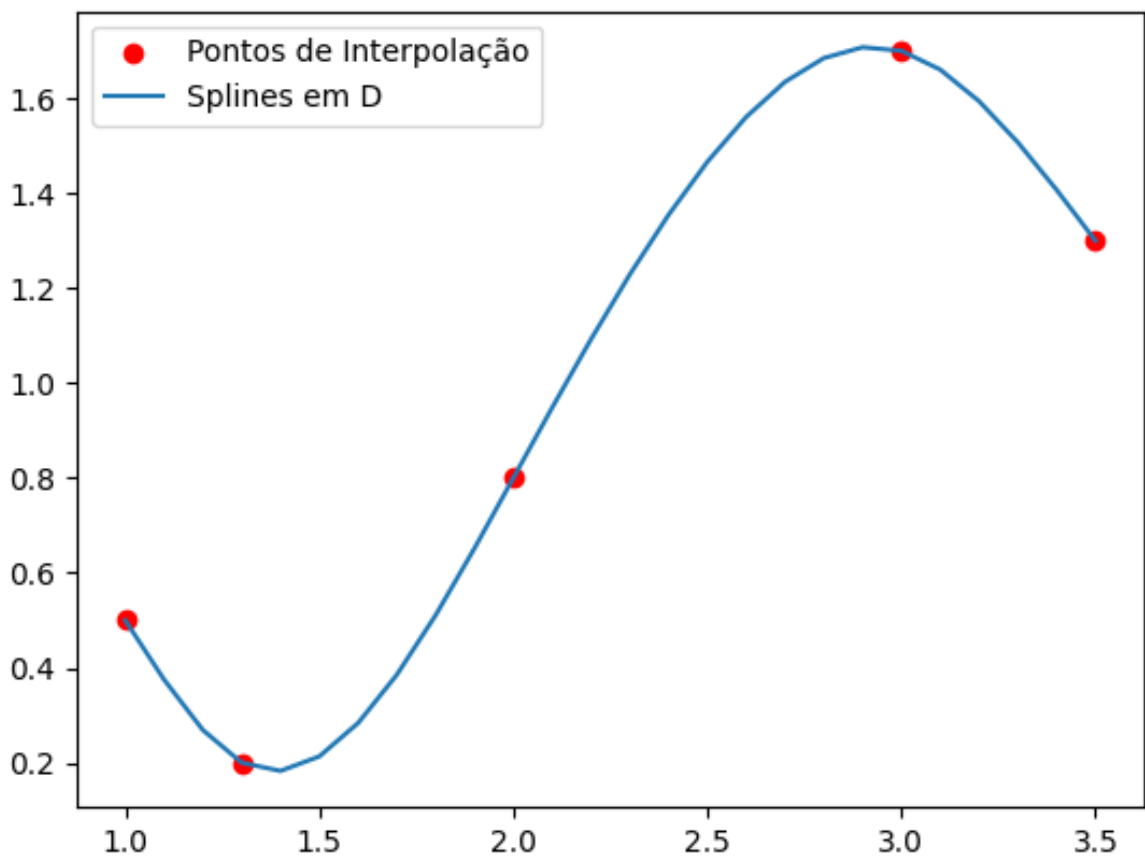
Para  $z = 1.1$ ,  $si(z) = 0.3751$

Valores de  $x_i$  e suas respectivas imagens  $si(x_i)$  para o conjunto de  $m$  pontos em  $D$ :

$x_0 = 1.0000$ ,  $si(x_0) = 0.5000$   
 $x_1 = 1.1000$ ,  $si(x_1) = 0.3751$   
 $x_2 = 1.2000$ ,  $si(x_2) = 0.2689$   
 $x_3 = 1.3000$ ,  $si(x_3) = 0.2000$   
 $x_4 = 1.4000$ ,  $si(x_4) = 0.1827$   
 $x_5 = 1.5000$ ,  $si(x_5) = 0.2132$   
 $x_6 = 1.6000$ ,  $si(x_6) = 0.2836$   
 $x_7 = 1.7000$ ,  $si(x_7) = 0.3855$   
 $x_8 = 1.8000$ ,  $si(x_8) = 0.5110$   
 $x_9 = 1.9000$ ,  $si(x_9) = 0.6519$   
 $x_{10} = 2.0000$ ,  $si(x_{10}) = 0.8000$   
 $x_{11} = 2.1000$ ,  $si(x_{11}) = 0.9481$   
 $x_{12} = 2.2000$ ,  $si(x_{12}) = 1.0920$   
 $x_{13} = 2.3000$ ,  $si(x_{13}) = 1.2285$

$x_{14} = 2.4000$ ,  $si(x_{14}) = 1.3543$   
 $x_{15} = 2.5000$ ,  $si(x_{15}) = 1.4661$   
 $x_{16} = 2.6000$ ,  $si(x_{16}) = 1.5606$   
 $x_{17} = 2.7000$ ,  $si(x_{17}) = 1.6345$   
 $x_{18} = 2.8000$ ,  $si(x_{18}) = 1.6846$   
 $x_{19} = 2.9000$ ,  $si(x_{19}) = 1.7075$   
 $x_{20} = 3.0000$ ,  $si(x_{20}) = 1.7000$   
 $x_{21} = 3.1000$ ,  $si(x_{21}) = 1.6605$   
 $x_{22} = 3.2000$ ,  $si(x_{22}) = 1.5939$   
 $x_{23} = 3.3000$ ,  $si(x_{23}) = 1.5072$   
 $x_{24} = 3.4000$ ,  $si(x_{24}) = 1.4070$   
 $x_{25} = 3.5000$ ,  $si(x_{25}) = 1.3000$

### Gráfico



## 2. Problema 2

Obter as splines interpoladoras dos seguintes dados:

9

0.0	1.25	2.5	3.75	5.0	6.25	7.5	8.75	10.0
2.0	3.0	4.3	4.5	4.0	3.4	3.1	3.6	2.4
3.0								
51								

Fazer o gráfico das splines em  $D = [a = 0.0, b = 10.0]$  usando  $m = 51$ .

### Saida gerada pelo programa:

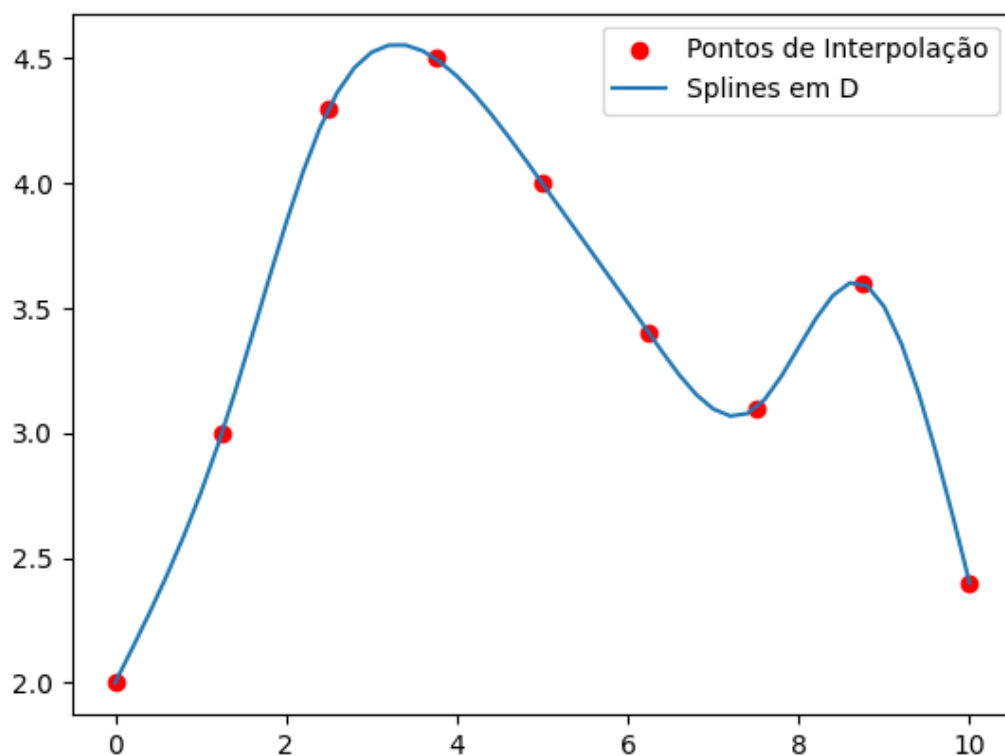
Para  $z = 3.0$ ,  $si(z) = 4.5247$

Valores de  $x_i$  e suas respectivas imagens  $si(x_i)$  para o conjunto de  $m$  pontos em  $D$ :

$x_0 = 0.0000$ ,  $si(x_0) = 2.0000$   
 $x_1 = 0.2000$ ,  $si(x_1) = 2.1372$   
 $x_2 = 0.4000$ ,  $si(x_2) = 2.2780$   
 $x_3 = 0.6000$ ,  $si(x_3) = 2.4259$   
 $x_4 = 0.8000$ ,  $si(x_4) = 2.5847$   
 $x_5 = 1.0000$ ,  $si(x_5) = 2.7578$   
 $x_6 = 1.2000$ ,  $si(x_6) = 2.9490$   
 $x_7 = 1.4000$ ,  $si(x_7) = 3.1607$   
 $x_8 = 1.6000$ ,  $si(x_8) = 3.3869$   
 $x_9 = 1.8000$ ,  $si(x_9) = 3.6170$   
 $x_{10} = 2.0000$ ,  $si(x_{10}) = 3.8404$   
 $x_{11} = 2.2000$ ,  $si(x_{11}) = 4.0465$   
 $x_{12} = 2.4000$ ,  $si(x_{12}) = 4.2247$   
 $x_{13} = 2.6000$ ,  $si(x_{13}) = 4.3646$   
 $x_{14} = 2.8000$ ,  $si(x_{14}) = 4.4633$   
 $x_{15} = 3.0000$ ,  $si(x_{15}) = 4.5247$   
 $x_{16} = 3.2000$ ,  $si(x_{16}) = 4.5534$   
 $x_{17} = 3.4000$ ,  $si(x_{17}) = 4.5539$   
 $x_{18} = 3.6000$ ,  $si(x_{18}) = 4.5305$   
 $x_{19} = 3.8000$ ,  $si(x_{19}) = 4.4877$   
 $x_{20} = 4.0000$ ,  $si(x_{20}) = 4.4295$   
 $x_{21} = 4.2000$ ,  $si(x_{21}) = 4.3585$   
 $x_{22} = 4.4000$ ,  $si(x_{22}) = 4.2774$   
 $x_{23} = 4.6000$ ,  $si(x_{23}) = 4.1889$   
 $x_{24} = 4.8000$ ,  $si(x_{24}) = 4.0955$   
 $x_{25} = 5.0000$ ,  $si(x_{25}) = 4.0000$   
 $x_{26} = 5.2000$ ,  $si(x_{26}) = 3.9045$   
 $x_{27} = 5.4000$ ,  $si(x_{27}) = 3.8092$   
 $x_{28} = 5.6000$ ,  $si(x_{28}) = 3.7138$   
 $x_{29} = 5.8000$ ,  $si(x_{29}) = 3.6180$

$x_{30} = 6.0000$ ,  $si(x_{30}) = 3.5217$   
 $x_{31} = 6.2000$ ,  $si(x_{31}) = 3.4245$   
 $x_{32} = 6.4000$ ,  $si(x_{32}) = 3.3267$   
 $x_{33} = 6.6000$ ,  $si(x_{33}) = 3.2339$   
 $x_{34} = 6.8000$ ,  $si(x_{34}) = 3.1544$   
 $x_{35} = 7.0000$ ,  $si(x_{35}) = 3.0964$   
 $x_{36} = 7.2000$ ,  $si(x_{36}) = 3.0683$   
 $x_{37} = 7.4000$ ,  $si(x_{37}) = 3.0782$   
 $x_{38} = 7.6000$ ,  $si(x_{38}) = 3.1339$   
 $x_{39} = 7.8000$ ,  $si(x_{39}) = 3.2292$   
 $x_{40} = 8.0000$ ,  $si(x_{40}) = 3.3442$   
 $x_{41} = 8.2000$ ,  $si(x_{41}) = 3.4583$   
 $x_{42} = 8.4000$ ,  $si(x_{42}) = 3.5510$   
 $x_{43} = 8.6000$ ,  $si(x_{43}) = 3.6018$   
 $x_{44} = 8.8000$ ,  $si(x_{44}) = 3.5902$   
 $x_{45} = 9.0000$ ,  $si(x_{45}) = 3.5061$   
 $x_{46} = 9.2000$ ,  $si(x_{46}) = 3.3596$   
 $x_{47} = 9.4000$ ,  $si(x_{47}) = 3.1634$   
 $x_{48} = 9.6000$ ,  $si(x_{48}) = 2.9297$   
 $x_{49} = 9.8000$ ,  $si(x_{49}) = 2.6711$   
 $x_{50} = 10.0000$ ,  $si(x_{50}) = 2.4000$

**Gráfico**



### Instruções para Rodar o Programa:

1. Execute o script Python em um ambiente que suporte a linguagem Python (por exemplo, um terminal ou IDE).
2. O programa solicitará a entrada dos seguintes dados via teclado:
  - A quantidade de pontos de interpolação (n).
  - Os valores de x separados por espaço.
  - Os valores de y separados por espaço.
  - O valor de z.
  - A quantidade de pontos m para calcular as imagens das splines em D.
3. Após fornecer os dados, o programa calculará  $s_i(z)$  e imprimirá os valores de  $s_i(x)$  para o conjunto de pontos igualmente espaçados em D.
4. O programa também plotará as splines e os pontos de interpolação e salvará esse gráfico em um arquivo chamado "spline.png"

### Observações:

- Certifique-se de fornecer os dados conforme solicitado pelo programa.
- Garanta que o ambiente de execução tenha o Python instalado.
- Pode ser necessário instalar a biblioteca Matplotlib, dependendo do ambiente, usando o comando `pip install matplotlib`.

### Código Utilizado:

```
import matplotlib.pyplot as plt
import numpy as np

def cubic_spline_natural(x, y):
    """
    Calcula os coeficientes dos polinômios cúbicos naturais para interpolação.

    Args:
        x (list): Lista de coordenadas x dos pontos de interpolação.
        y (list): Lista de coordenadas y dos pontos de interpolação.

    Returns:
        tuple: Coeficientes dos polinômios cúbicos naturais (a, b, c, d).
    """
    n = len(x)
    h = {k: x[k+1] - x[k] for k in range(n - 1)}

    # Construção da matriz tridiagonal
    A = np.zeros((n, n))
    for i in range(1, n - 1):
        A[i, i-1] = h[i-1]
        A[i, i] = 2 * (h[i-1] + h[i])
```

```
A[i, i+1] = h[i]

A[0, 0] = 1
A[-1, -1] = 1

# Construção do vetor B
B = np.zeros(n)
for k in range(1, n - 1):
    B[k] = 3 * ((y[k+1] - y[k]) / h[k] - (y[k] - y[k-1]) / h[k-1])

# Resolução do sistema linear para obter os coeficientes c
c = np.linalg.solve(A, B)

# Cálculo dos demais coeficientes a, b, d
a = y
b = np.zeros(n-1)
d = np.zeros(n-1)
for k in range(n-1):
    b[k] = (1/h[k]) * (a[k+1] - a[k]) - (h[k]/3) * (2*c[k] + c[k+1])
    d[k] = (c[k+1] - c[k]) / (3 * h[k])

return a, b, c, d

def evaluate_spline(x, a, b, c, d, xi):
    """
    Avalia o valor da spline cúbica no ponto xi.

    Args:
        x (list): Lista de coordenadas x dos pontos de interpolação.
        a (list): Coeficientes a dos polinômios cúbicos naturais.
        b (list): Coeficientes b dos polinômios cúbicos naturais.
        c (list): Coeficientes c dos polinômios cúbicos naturais.
        d (list): Coeficientes d dos polinômios cúbicos naturais.
        xi (float): Ponto onde a spline cúbica é avaliada.

    Returns:
        float: Valor da spline cúbica no ponto xi.
    """
    k = 0
    while x[k+1] < xi:
        k += 1

    # Distância do ponto xi para o ponto inicial do intervalo
    dx = xi - x[k]

    # Avaliação do polinômio cúbico no ponto xi
```

```
result = a[k] + b[k] * dx + c[k] * dx**2 + d[k] * dx**3
return result

# Entrada dos valores via teclado
n = int(input("Digite a quantidade de pontos de interpolação: "))
x_values = list(map(float, input("Digite os valores de x separados por espaço: ").split()))
y_values = list(map(float, input("Digite os valores de y separados por espaço: ").split()))
z_value = float(input("Digite o valor de z: "))
m = int(input("Digite a quantidade de pontos m para calcular as imagens das splines em D: "))

# Verifica se o número de pontos de interpolação é igual
if len(x_values) != n or len(y_values) != n:
    print("Erro: O número de pontos de interpolação não é igual.")
else:
    a, b, c, d = cubic_spline_natural(x_values, y_values)

    # Calcular si(z)
    si_z = evaluate_spline(x_values, a, b, c, d, z_value)
    print(f"Para z = {z_value}, si(z) = {si_z:.4f}")

    # Imprimir valores xi e suas respectivas imagens si(x) para o conjunto de m pontos em D
    print("Valores de xi e suas respectivas imagens si(x) para o conjunto de m pontos em D:")
    for i, xi in enumerate(np.linspace(x_values[0], x_values[-1], m)):
        si_xi = evaluate_spline(x_values, a, b, c, d, xi)
        #limitar as casas decimais
        print(f"x{i} = {xi:.4f}, si(x{i}) = {si_xi:.4f}")

    # Gerar pontos igualmente espaçados em D para plotagem
    plot_points = np.linspace(x_values[0], x_values[-1], m)

    # Calcular as imagens das splines em D
    spline_images = [evaluate_spline(x_values, a, b, c, d, xi) for xi in plot_points]

    # Plotar as splines em D
    plt.scatter(x_values, y_values, color='red', Label='Pontos de Interpolação')
    plt.plot(plot_points, spline_images, Label='Splines em D')
    plt.legend()
    plt.savefig('spline.png')
```