#### **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 x0 = 1.0 e b = 3.5)

**Saida gerada pelo programa:**

Para z = 1.1, si(z) = 0.3751042106151379

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

x0 = 1.0, si(x0) = 0.5

x1 = 1.1, si(x1) = 0.3751042106151379

x2 = 1.2, si(x2) = 0.2688802632689225

x3 = 1.3, si(x3) = 0.19999999999999996

x4 = 1.4, si(x4) = 0.18266909362681877

x5 = 1.5, si(x5) = 0.21322854008703312

x6 = 1.6, si(x6) = 0.28355316609809916

x7 = 1.7000000000000002, si(x7) = 0.3855177983774729

x8 = 1.8, si(x8) = 0.5109972636426101

x9 = 1.9, si(x9) = 0.651866388610967

x10 = 2.0, si(x10) = 0.8

x11 = 2.1, si(x11) = 0.9480795797822971

x12 = 2.2, si(x12) = 1.0920132309509747

x13 = 2.3, si(x13) = 1.2285157117542822

x14 = 2.4000000000000004, si(x14) = 1.3543017804404696

x15 = 2.5, si(x15) = 1.4660861952577842

x16 = 2.6, si(x16) = 1.5605837144544763

x17 = 2.7, si(x17) = 1.6345090962787951

x18 = 2.8, si(x18) = 1.684577098978989

x19 = 2.9000000000000004, si(x19) = 1.7075024808033072

x20 = 3.0, si(x20) = 1.6999999999999997

x21 = 3.1, si(x21) = 1.6604557252552525

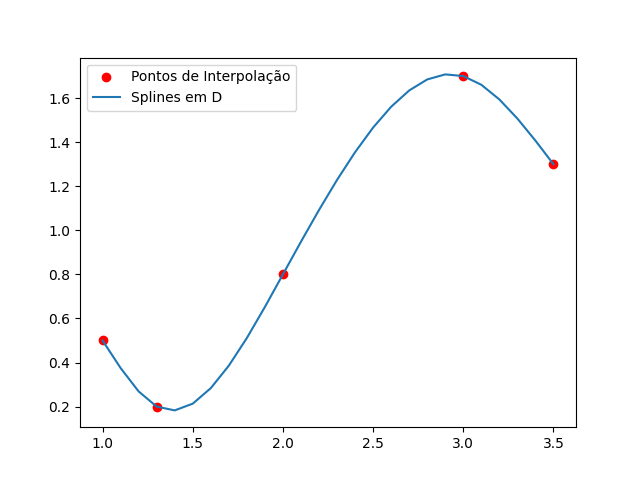
x22 = 3.2, si(x22) = 1.5939409670070035

x23 = 3.3000000000000003, si(x23) = 1.507198346131128

x24 = 3.4000000000000004, si(x24) = 1.4069704835035015

x25 = 3.5, si(x25) = 1.3

**Gráfico**



1. **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) = 4.524707805596465

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

x0 = 0.0, si(x0) = 2.0

x1 = 0.2, si(x1) = 2.137179810309278

x2 = 0.4, si(x2) = 2.2779568918998527

x3 = 0.6000000000000001, si(x3) = 2.4259285160530197

x4 = 0.8, si(x4) = 2.584691954050074

x5 = 1.0, si(x5) = 2.7578444771723123

x6 = 1.2000000000000002, si(x6) = 2.9489833567010315

x7 = 1.4000000000000001, si(x7) = 3.1607066650957294

x8 = 1.6, si(x8) = 3.386915744329897

x9 = 1.8, si(x9) = 3.6170340453608247

x10 = 2.0, si(x10) = 3.840448011782032

x11 = 2.2, si(x11) = 4.046544087187039

x12 = 2.4000000000000004, si(x12) = 4.224708715169367

x13 = 2.6, si(x13) = 4.364642174963182

x14 = 2.8000000000000003, si(x14) = 4.463262965537555

x15 = 3.0, si(x15) = 4.524707805596465

x16 = 3.2, si(x16) = 4.553427249484536

x17 = 3.4000000000000004, si(x17) = 4.5538718515463925

x18 = 3.6, si(x18) = 4.530492166126657

x19 = 3.8000000000000003, si(x19) = 4.487734037113402

x20 = 4.0, si(x20) = 4.4294733431516935

x21 = 4.2, si(x21) = 4.3584790055964655

x22 = 4.4, si(x22) = 4.2773927634756985

x23 = 4.6000000000000005, si(x23) = 4.188856355817379

x24 = 4.800000000000001, si(x24) = 4.095511521649485

x25 = 5.0, si(x25) = 4.0

x26 = 5.2, si(x26) = 3.9044779216494847

x27 = 5.4, si(x27) = 3.809158984388807

x28 = 5.6000000000000005, si(x28) = 3.7137712777614134

x29 = 5.800000000000001, si(x29) = 3.6180428913107505

x30 = 6.0, si(x30) = 3.521701914580265

x31 = 6.2, si(x31) = 3.4244764371134018

x32 = 6.4, si(x32) = 3.326695594698085

x33 = 6.6000000000000005, si(x33) = 3.2339198515463914

x34 = 6.800000000000001, si(x34) = 3.154403249484536

x35 = 7.0, si(x35) = 3.096422091310751

x36 = 7.2, si(x36) = 3.0682526798232694

x37 = 7.4, si(x37) = 3.078171317820324

x38 = 7.6000000000000005, si(x38) = 3.13385385802651

x39 = 7.800000000000001, si(x39) = 3.2291658014727544

x40 = 8.0, si(x40) = 3.344162297496318

x41 = 8.200000000000001, si(x41) = 3.4582980453608254

x42 = 8.4, si(x42) = 3.551027744329897

x43 = 8.6, si(x43) = 3.6018060936671574

x44 = 8.8, si(x44) = 3.590173756701031

x45 = 9.0, si(x45) = 3.5060730486008835

x46 = 9.200000000000001, si(x46) = 3.3596478397643583

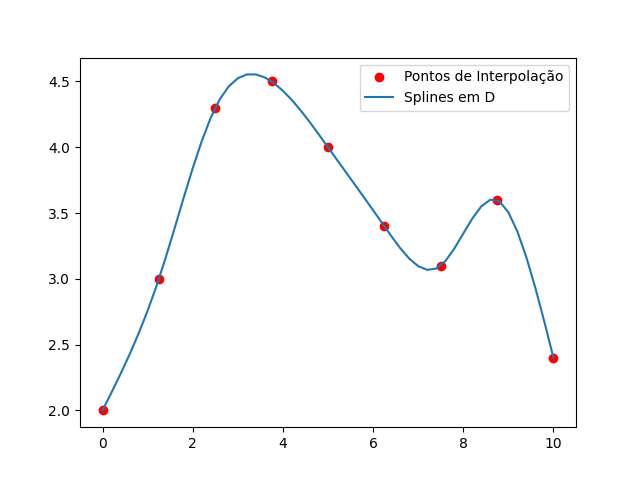
x47 = 9.4, si(x47) = 3.1633630303387332

x48 = 9.600000000000001, si(x48) = 2.9296835204712797

x49 = 9.8, si(x49) = 2.671074210309277

x50 = 10.0, si(x50) = 2.3999999999999995

**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 si(z) e imprimirá os valores de si(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}")
* # 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)
* print(f"x{i} = {xi}, si(x{i}) = {si\_xi}")
* # 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')