Cálculo Numérico

Um guia prático com Python

Prof. Dr. Rogério Vargas[1]

[1]Centro de Estudos do Mar
Universidade Federal do Paraná

2026

rogerio.in

UFPR

Encontro 24: Aplicação prática de interpolação

Da aula anterior para a prática

Na aula anterior, usamos interpolação linear para estimar valores entre dois pontos.

> **Ideia principal**
> Entre dois pontos conhecidos, aproximamos o comportamento por uma reta.

Hoje vamos aplicar essa ideia várias vezes para criar novos pontos entre pontos consecutivos.

$$x(t) = (1 - t)x_i + tx_{i+1}$$

$$y(t) = (1 - t)y_i + ty_{i+1}$$

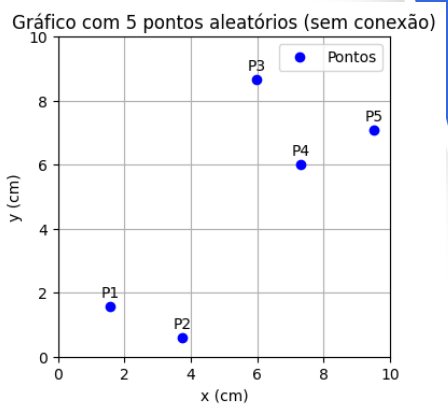
A problemática

Temos um gráfico com poucos pontos isolados.

Pergunta
O que acontece entre os pontos?

Queremos preencher esse “vazio” de forma controlada.

Critério
Criar pontos intermediários para que a distância entre pontos consecutivos seja, no máximo, 2 cm.



Coordenadas iniciais

Ponto	x	y
P_1	1,56	1,56
P_2	3,75	0,58
P_3	5,99	8,66
P_4	7,32	6,01
P_5	9,51	7,08

Objetivo computacional

Percorrer os pontos consecutivos e inserir novos pontos quando a distância entre eles for maior que 2 unidades.

Distância entre dois pontos

Antes de interpolar, precisamos medir a distância entre dois pontos consecutivos.

Dados:

$$A = (x_1, y_1) \qquad B = (x_2, y_2)$$

A distância euclidiana é:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Uso no algoritmo
Se a distância for maior que 2, o segmento será subdividido.

Exemplo: distância entre P_1 e P_2

Pontos considerados:

$$P_1 = (1{,}56,\ 1{,}56)$$

$$P_2 = (3{,}75,\ 0{,}58)$$

Aplicando a fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{(3{,}75 - 1{,}56)^2 + (0{,}58 - 1{,}56)^2}$$

Simplificando:

$$d = \sqrt{(2{,}19)^2 + (-0{,}98)^2}$$

$$d = \sqrt{4{,}80 + 0{,}96}$$

$$d = \sqrt{5{,}76}$$

$$\boxed{d \approx 2{,}40}$$

Conclusão

Como $2{,}40 > 2$, precisamos inserir pelo menos um ponto intermediário.

Quantos segmentos criar?

Queremos que cada segmento tenha comprimento máximo igual a 2.

Se a distância entre dois pontos é d, usamos:

$$n = \left\lceil \frac{d}{2} \right\rceil$$

onde n é o número de segmentos.

Consequência
O número de pontos internos criados será:

$$n - 1$$

Para P_1 e P_2:

$$d \approx 2{,}40$$

$$n = \left\lceil \frac{2{,}40}{2} \right\rceil$$

$$n = \lceil 1{,}20 \rceil = 2$$

Resultado
Criamos:

$$n - 1 = 2 - 1 = 1$$

ponto intermediário.

Interpolação linear no segmento

Para gerar pontos entre:

$$P_i = (x_i, y_i) \qquad P_{i+1} = (x_{i+1}, y_{i+1})$$

usamos:

$$x(t) = (1-t)x_i + tx_{i+1}$$

$$y(t) = (1-t)y_i + ty_{i+1}$$

Parâmetro t

$$0 < t < 1$$

O valor de t indica a posição relativa entre os 2 cm.

Ponto intermediário entre P_1 e P_2

Como $n = 2$, usamos:

$$t = \frac{1}{2}$$

$$P_1 = (1{,}56,\ 1{,}56)$$

$$P_2 = (3{,}75,\ 0{,}58)$$

Fórmulas:

$$x(t) = (1 - t)x_1 + tx_2$$

$$y(t) = (1 - t)y_1 + ty_2$$

Para $t = 0{,}5$:

$$x = 0{,}5 \cdot 1{,}56 + 0{,}5 \cdot 3{,}75$$

$$x = 2{,}65$$

$$y = 0{,}5 \cdot 1{,}56 + 0{,}5 \cdot 0{,}58$$

$$y = 1{,}07$$

Ponto interpolado

$$I_1 = (2{,}65,\ 1{,}07)$$

Um segmento maior: P_2 até P_3

Pontos considerados:

$$P_2 = (3{,}75,\ 0{,}58)$$

$$P_3 = (5{,}99,\ 8{,}66)$$

A distância é aproximadamente:

$$d \approx 8{,}38$$

Logo:

$$n = \left\lceil \frac{8{,}38}{2} \right\rceil = 5$$

Portanto, o segmento será dividido em 5 partes.

Como teremos 5 segmentos, os pontos internos usam:

$$t = \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}$$

ou seja:

$$t = 0{,}2,\ 0{,}4,\ 0{,}6,\ 0{,}8$$

Interpretação
Cada valor de t gera um ponto intermediário entre P_2 e P_3.

Implementando a distância em Python

```python
import math

def distancia(p1, p2):
    x1, y1 = p1
    x2, y2 = p2

    dx = x2 - x1
    dy = y2 - y1

    return math.sqrt(dx**2 + dy
        **2)
```

Teste

```
p1 = (1.56, 1.56)
p2 = (3.75, 0.58)

print(distancia(p1, p2)
    )
```

Saída esperada

```
2.399
```

Implementando a interpolação linear

```
1  def interpolar(p1, p2, t):
2      x1, y1 = p1
3      x2, y2 = p2
4
5      x = (1 - t) * x1 + t * x2
6      y = (1 - t) * y1 + t * y2
7
8      return (x, y)
```

Exemplo

```
p1 = (1.56, 1.56)
p2 = (3.75, 0.58)

print(interpolar(p1, p2
    , 0.5))
```

Saída esperada

```
(2.655, 1.07)
```

Gerando pontos intermediários

```python
import math
def pontos_intermediarios(p1, p2, limite=2):
    d = distancia(p1, p2)
    n = math.ceil(d / limite)
    novos = []
    for k in range(1, n):
        t = k / n
        novos.append(interpolar(p1, p2, t))
    return novos
```

Ideia

Se $n = 1$, nenhum ponto novo é criado. Se $n = 5$, são criados 4 pontos internos.

Aplicando em todos os pontos

```
pontos = [
    (1.56, 1.56),
    (3.75, 0.58),
    (5.99, 8.66),
    (7.32, 6.01),
    (9.51, 7.08)
]
resultado = []
for i in range(len(pontos) - 1):
    p_atual = pontos[i]
    p_proximo = pontos[i + 1]
    resultado.append(p_atual)
    resultado.extend(
        pontos_intermediarios(p_atual, p_proximo)
    )
resultado.append(pontos[-1])
```

Separando coordenadas para o gráfico

```
x_original = [p[0] for p in pontos]
y_original = [p[1] for p in pontos]

x_interp = [p[0] for p in resultado]
y_interp = [p[1] for p in resultado]
```

Próximo passo
Usar essas listas para construir o gráfico antes e depois da interpolação.

Visualizando com Matplotlib

```
import matplotlib.pyplot as plt

plt.plot(x_original, y_original, 'o', label='originais')
plt.plot(x_interp, y_interp, '-o', label='interpolados')

plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```

Interpretação

Os novos pontos tornam a representação mais densa, mantendo o critério de distância máxima.

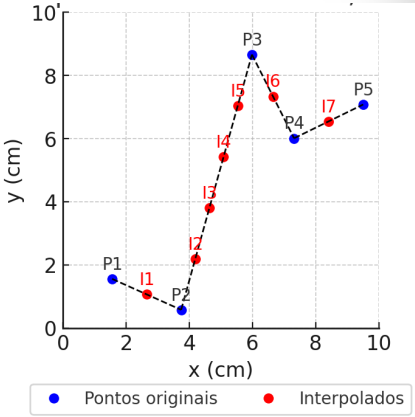
Resultado visual

Depois da interpolação, os segmentos longos foram subdivididos.

Resultado

A curva fica visualmente mais contínua, mas continua sendo construída por segmentos de reta.

A interpolação não cria a função verdadeira. Ela cria uma representação mais detalhada entre os pontos conhecidos.



Resumo do algoritmo

Para cada par de pontos consecutivos:

1. Calcular a distância euclidiana.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2. Calcular o número de segmentos:

$$n = \left\lceil \frac{d}{2} \right\rceil$$

3. Gerar os valores:

$$t = \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}$$

4. Aplicar:

$$x(t) = (1-t)x_i + tx_{i+1}$$

$$y(t) = (1-t)y_i + ty_{i+1}$$

Exercício em sala

Modifique o código para testar outros valores de distância máxima.

Tarefa
Execute o algoritmo usando:

$$limite = 1, \quad limite = 2, \quad limite = 3$$

Observe:

- quantos pontos novos são criados;
- como o gráfico muda;
- qual valor parece produzir uma representação mais adequada.

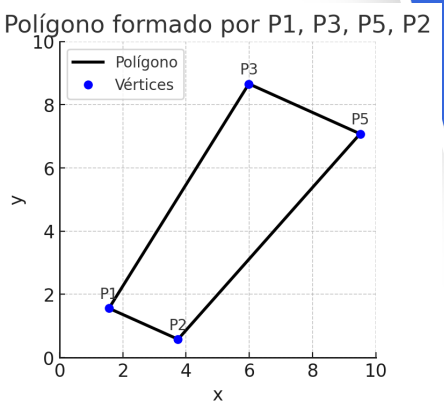
Gancho: interpolação em uma área

Até agora interpolamos ao longo de uma sequência de pontos.

Mas e se os pontos delimitarem uma região?

> **Pergunta**
> Seria possível criar pontos interpolados dentro de uma área?

Essa ideia aparece em mapas, superfícies, modelos ambientais e imagens.



Resumo da aula

Nesta prática, usamos interpolação linear para gerar novos pontos.

Ideias principais:

- pontos consecutivos formam segmentos;
- segmentos longos podem ser subdivididos;
- a distância euclidiana ajuda a decidir quantas divisões fazer;
- a interpolação linear gera os pontos intermediários;
- Python permite automatizar o processo;
- o gráfico ajuda a verificar o resultado.

Mensagem final
Interpolar também é transformar uma ideia geométrica em algoritmo.

Importante!

Este material é exclusivo de uso do autor. Proibido copiar ou replicar.

rogeriovargas@ufpr.br

rogerio.in

UFPR

Cálculo Numérico

Um guia prático com Python

Prof. Dr. Rogério Vargas[1]

[1]Centro de Estudos do Mar
Universidade Federal do Paraná

2026

rogerio.in

UFPR