Importando o NumPy

```
import numpy as np
```

O que faz: importa a biblioteca NumPy, que é essencial para lidar com matrizes e vetores de forma eficiente em Python.

Definindo a função principal

```
def gauss_seidel(A, b, x0=None, tol=1e-6, max_iter=100):
```

O que faz: define a função gauss_seidel que vai resolver o sistema Ax = b de forma iterativa.

- A: matriz dos coeficientes do sistema;
- b: vetor dos termos constantes;
- x0: chute inicial (opcional). Se não for passado, será usado o vetor zero;
- tol: tolerância do erro relativo (critério de parada);
- max_iter: número máximo de iterações.

Inicialização

```
    n = len(b)
    x = np.zeros(n) if x0 is None else x0.copy()
```

- n: número de variáveis (ou linhas de A e b).
- x: vetor solução aproximada. Começa como vetor nulo, a menos que o usuário forneça um chute inicial (x0).

Cabeçalho da saída

```
    print(f"{'Iteração':>9} | {'x':>20} | {'Erro relativo':>15}")
    print("-" * 55)
```

- Isso imprime um cabeçalho bonito para mostrar a evolução da solução em cada iteração.

Início do laço iterativo

```
for k in range(max_iter):
    x_old = x.copy()
```

- k: número da iteração atual.
- x_old: uma cópia da solução anterior, usada para calcular o erro relativo.

Atualização dos valores de x

```
for i in range(n):
    soma = sum(A[i][j] * x[j] for j in range(n) if j != i)
    x[i] = (b[i] - soma) / A[i][i]
```

Este é o núcleo do método Gauss-Seidel!

- Para cada variável x_i, calcula-se a soma dos outros termos da equação (com os x_j, j ≠ i).
- Depois, isola-se x_i na equação:

  $x_i^{(k+1)} = (1/A_{ii}) * (b_i - \Sigma_{j \neq i} A_{ij} x_j)$

- Usamos os valores atualizados de x imediatamente após serem calculados.

Cálculo do erro relativo

```
erro_rel = np.linalg.norm(x - x_old, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
```

- Calcula o erro relativo como:

  $||x^{(k)} - x^{(k-1)}||_{inf} / ||x^{(k)}||_{inf}$

- A norma infinito ($||\cdot||_{inf}$) pega o maior valor absoluto.

Impressão dos resultados parciais

```
        print(f"{k+1:9} | {x} | {erro_rel:15.2e}")
```

- Mostra a iteração atual, a estimativa de x, e o erro relativo em notação científica.

Verificação do critério de parada

```
        if erro_rel < tol:
            print("\nConvergência atingida.")
            return x
```

- Se o erro relativo ficar menor que a tolerância tol, o loop é encerrado e a solução é retornada.

Caso não convirja

```
    print("\nNúmero máximo de iterações atingido.")
    return x
```

- Se não atingir a convergência em max_iter iterações, ainda retorna o valor de x.

Exemplo de uso (fora da função)

```
A = np.array([[4, 1],
              [2, 3]], dtype=float)

b = np.array([7, 8], dtype=float)

x_aprox = gauss_seidel(A, b)
print("\nSolução aproximada:", x_aprox)
```

- Define o sistema linear:

4x1 + x2 = 7

2x1 + 3x2 = 8

- Executa o método e imprime o resultado.