Sistemas de Equações Algébricas Lineares

Disciplina: Métodos Numéricos em Termofluidos

Professor: Adriano Possebon Rosa

1 Motivação

Considere a equação do calor para o caso **unidimensional permanente** ($T = T(x)$),

$$\frac{d^2T}{dx^2} = 0 \ , \tag{1}$$

para $0 < x < 1$ e com condições de contorno dadas por

$$T(x = 0) = 0 \qquad \text{e} \qquad T(x = 1) = 1 \ . \tag{2}$$

Utilizando expansões em série de Taylor, podemos aproximar a segunda derivada de T com relação a x por

$$\frac{d^2T}{dx^2} \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} \ . \tag{3}$$

Essa aproximação contém um erro $O(\Delta x^2)$. Da substituição dessa aproximação de volta na equação governante (1), resulta

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} = 0 \ . \tag{4}$$

Essa é a EDF (Equação de Diferenças Finitas) desse problema. Podemos reescrevê-la como

$$T_{i+1} - 2T_i + T_{i-1} = 0 \ . \tag{5}$$

Note que agora **não** é possível encontrar diretamente o valor de cada T_i pois, ao tentar isolá-los em (5), temos

$$T_i = \frac{T_{i+1} + T_{i-1}}{2} \ , \tag{6}$$

o que não ajuda muito, pois não conhecemos os valores da temperatura nos pontos vizinhos de i. As temperaturas estão **acopladas**: para encontrar o valor de T_i em um ponto i, é necessário encontrar em todos os pontos.

Vamos analisar um caso particular em que o domínio é dividido em 5 partes iguais, ou seja, $N = 5$. A figura (1) mostra o nosso domínio discretizado. Pelas condições de contorno, os pontos dos extremos já estão definidos. Assim,

$$T_0 = 0 \qquad \text{e} \qquad T_5 = 1 \ . \tag{7}$$

Temos que descobrir, portanto, os valores de T_1, T_2, T_3 e T_4.

A equação geral, para qualquer ponto i, é a equação (5). Vamos abrir essa equação em cada ponto:

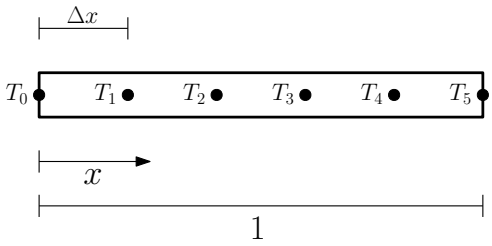

Figura 1: Barra discretizada com comprimento 1.

$$\begin{aligned} i = 1 \quad & T_2 - 2T_1 + T_0 = 0 \\ i = 2 \quad & T_3 - 2T_2 + T_1 = 0 \\ i = 3 \quad & T_4 - 2T_3 + T_2 = 0 \\ i = 4 \quad & T_5 - 2T_4 + T_3 = 0 \end{aligned} \tag{8}$$

Somando essas 4 equações com as condições de contorno, resulta no seguinte conjunto de equações lineares acopladas:

$$\begin{aligned} T_0 &= 0 \\ T_2 - 2T_1 + T_0 &= 0 \\ T_3 - 2T_2 + T_1 &= 0 \\ T_4 - 2T_3 + T_2 &= 0 \\ T_5 - 2T_4 + T_3 &= 0 \\ T_5 &= 1 \end{aligned} \tag{9}$$

Podemos deixar assim, ou podemos substituir os valores dos pontos dos cantos nas equações dos seus vizinhos. Isso é feito da seguinte maneira:

$$\begin{aligned} T_2 - 2T_1 &= 0 \\ T_3 - 2T_2 + T_1 &= 0 \\ T_4 - 2T_3 + T_2 &= 0 \\ -2T_4 + T_3 &= -1 \end{aligned} \tag{10}$$

Esse é o nosso **Sistema de Equações Algébricas Lineares**. Na forma matricial, ele pode ser reescrito como:

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \tag{11}$$

Temos 4 **equações e** 4 **incógnitas.** Encontrar a solução numérica aproximada da EDO original é equivalente a **encontrar a solução para esse sistema**.

Em muitos problemas numéricos (Volumes Finitos, Elementos Finitos, Elementos de Contorno), a etapa final da solução consiste na resolução de um sistema de equações lineares. **Por isso, o estudo de métodos para resolver esses sistemas é muito importante.** Nas próximas seções veremos alguns desses métodos. Antes, porém, serão apresentadas algumas definições sobre matrizes que nos serão úteis.

2 Definições

No caso geral, temos um **sistema com n equações** dado por

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n &= b_2 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots & \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n &= b_n \end{aligned} \tag{12}$$

Na representação indicial,

$$\sum_{j=1}^{n} A_{ij}x_j = b_i \qquad \text{para } i = 1, 2, \cdots, n \tag{13}$$

Na representação matricial,

$$\mathbf{Ax} = \mathbf{b}\ . \tag{14}$$

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \tag{15}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{16}$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{17}$$

A é a matriz dos coeficientes (*input*), **b** é a matriz dos termos independentes (*input*) e **x** é a matriz de incógnitas (*output*).
Na componente A_{ij} da matriz **A**, i representa a equação e j representa a variável, como mostra a figura (2).

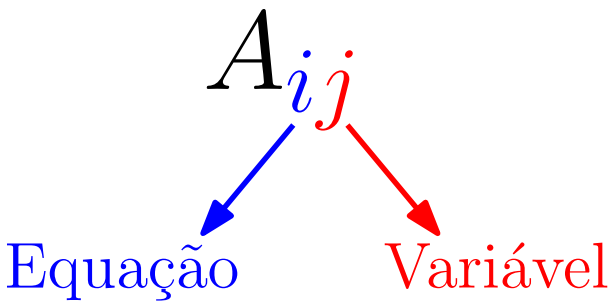

Figura 2: Componente da matriz **A**.

Exemplo: com $n = 2$ temos

$$\begin{cases} A_{11}x_1 + A_{12}x_2 = b_1 \\ A_{21}x_1 + A_{22}x_2 = b_2 \end{cases} \tag{18}$$

Nessa situação, temos 4 possibilidades (ver figura 3):

i) Solução Única
ii) Sem solução
iii) Número infinito de soluções
iv) Solução trivial ($\mathbf{x} = \mathbf{0}$) para um sistema homogêneo ($\mathbf{b} = \mathbf{0}$)

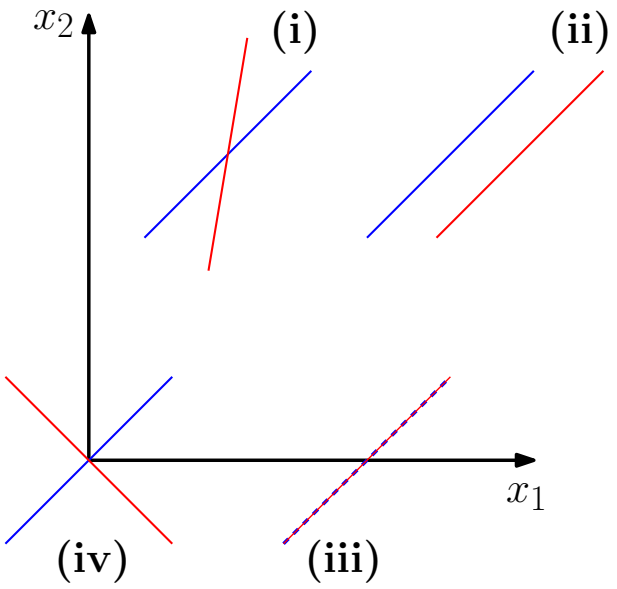

Figura 3: Possibilidades para o caso em que $n = 2$.

2.1 Produto de matrizes

Vamos definir o produto entre matrizes como

$$\mathbf{Ax} = \mathbf{b} \qquad b_i = \sum_{j=1}^{n} A_{ij} b_j \tag{19}$$

$$\mathbf{AB} = \mathbf{C} \qquad C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \tag{20}$$

2.2 Tipos Especiais de Matrizes

Matriz Quadrada: mesmo número de linhas e colunas.

Vetor coluna: n linhas e 1 coluna.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{21}$$

Vetor linha: 1 linha e n colunas.

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} \tag{22}$$

Matriz diagonal: $D_{ij} = 0$ se $i \neq j$.

$$\mathbf{D} = \begin{bmatrix} D_{11} & 0 & 0 & 0 \\ 0 & D_{22} & 0 & 0 \\ 0 & 0 & D_{33} & 0 \\ 0 & 0 & 0 & D_{44} \end{bmatrix} \tag{23}$$

Matriz identidade: $D_{ij} = 0$ se $i \neq j$ e $D_{ij} = 1$ se $i = j$ **ou** $D_{ij} = \delta_{ij}$.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

$$\mathbf{IA} = \mathbf{A} \tag{25}$$

$$\mathbf{Ix} = \mathbf{x} \tag{26}$$

Matriz Triangular Superior: $U_{ij} = 0$ se $i > j$.

$$\mathbf{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ 0 & U_{22} & U_{23} & U_{24} \\ 0 & 0 & U_{33} & U_{34} \\ 0 & 0 & 0 & U_{44} \end{bmatrix} \tag{27}$$

Matriz Tridiagonal:

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & 0 & 0 & 0 \\ T_{12} & T_{22} & T_{23} & 0 & 0 \\ 0 & T_{32} & T_{33} & T_{34} & 0 \\ 0 & 0 & T_{43} & T_{44} & T_{45} \\ 0 & 0 & 0 & T_{54} & T_{55} \end{bmatrix} \tag{28}$$

Matriz Transposta: $A_{ij}^T = A_{ji}$.

$$\mathbf{A}^T = \begin{bmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ A_{13} & A_{23} & A_{33} & A_{43} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{bmatrix} \tag{29}$$

Matriz Simétrica: $A_{ij} = A_{ji}$, **ou** $\mathbf{A}^T = \mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{12} & A_{22} & A_{23} & A_{24} \\ A_{13} & A_{23} & A_{33} & A_{34} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{bmatrix} \tag{30}$$

Matriz Esparsa: a maioria dos elementos é zero.

Matriz Diagonal Dominante: temos

$$|A_{ii}| \geq \sum_{j=1, j\neq i}^{n} |A_{ij}| \qquad \text{para } i = 1, 2, \cdots, n \tag{31}$$

O valor absoluto do elemento da diagonal principal é maior ou igual que a soma dos valores absolutos dos outros elementos da mesma linha, **com a desigualdade válida em pelo menos uma linha.**
Exemplo de matriz diagonal dominante:

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix} \tag{32}$$

Matriz Inversa: $\mathbf{A^{-1}}$. Propriedade:

$$\mathbf{A A^{-1}} = \mathbf{A^{-1} A} = \mathbf{I} \tag{33}$$

Para o sistema que estamos tentando encontrar a solução:

$$\mathbf{Ax} = \mathbf{b} \tag{34}$$

$$\mathbf{A^{-1}} (\mathbf{Ax}) = \mathbf{A^{-1} b} \tag{35}$$

$$\left(\mathbf{A^{-1} A}\right) \mathbf{x} = \mathbf{A^{-1} b} \tag{36}$$

$$\mathbf{Ix} = \mathbf{A^{-1} b} \tag{37}$$

$$\mathbf{x} = \mathbf{A^{-1} b} \tag{38}$$

2.3 Determinante

O **determinante** é um número associado a uma matriz quadrada e será representado por $\det(\mathbf{A})$ ou por $|\mathbf{A}|$.
Para $n = 2$:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{39}$$

$$\det(\mathbf{A}) = A_{11} A_{22} - A_{12} A_{21} \tag{40}$$

Para $n = 3$:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \tag{41}$$

$$\det(\mathbf{A}) = A_{11}A_{22}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} - A_{13}A_{22}A_{31} \quad (42)$$

Para $n \geq 4$: método dos cofatores.

Se $\det(\mathbf{A}) = 0$, a matriz $\mathbf{A}$ é singular (não admite inversa) e o **sistema linear correspondente não tem solução única** (pode ter infinitas soluções ou não ter solução).

3 Métodos de Solução de Sistemas Lineares

Temos dois tipos de métodos usados para resolver sistemas lineares: os **Métodos Diretos** e os **Métodos Iterativos**.

Métodos Diretos são aqueles que dão a **solução exata** (a menos do erro de máquina) após um certo número de passos. Exemplos de Métodos Diretos:

1. Eliminação Simples
2. Eliminação de Gauss
3. Decomposição LU
4. Eliminação de Gauss-Jordan
5. Algoritmo de Thomas

Métodos Iterativos, por sua vez, obtêm uma **solução aproximada** para o sistema de equações, a partir de uma resposta inicial (*chute*) e de um algoritmo de iteração. Alguns exemplos de Métodos Iterativos:

1. Gauss-Jacobi
2. Gauss-Seidel
3. SOR
4. Gradiente Conjugado
5. Multigrid

4 Método Direto: Eliminação de Gauss

Os métodos diretos consistem, basicamente, na aplicação de uma sequência de operações entre as equações para que se consiga isolar as incógnitas. O método mais conhecido é o de **eliminação de Gauss**. O objetivo é transformar o sistema original

$$\mathbf{Ax} = \mathbf{b} \quad (43)$$

em um sistema do tipo

$$\mathbf{Ux} = \mathbf{c}\ , \quad (44)$$

em que $\mathbf{U}$ é uma matriz triangular superior. Para isso, vamos considerar a matriz $\mathbf{A}^*$ do sistema,

$$\mathbf{A}^* = \begin{bmatrix} A_{11} & A_{12} & A_{13} & b_1 \\ A_{21} & A_{22} & A_{23} & b_2 \\ A_{31} & A_{32} & A_{33} & b_3 \end{bmatrix} \quad (45)$$

que envolve as matrizes $\mathbf{A}$ e $\mathbf{b}$. Podemos realizar as seguintes operações de linha (sem que o resultado seja alterado):

- qualquer linha pode ser multiplicada por uma constante (*scaling*);
- a ordem das linhas pode ser trocada (*pivoting*);
- qualquer linha pode ser trocada por uma combinação linear desta linha com qualquer outra linha (*elimination*).

Vamos ver como resolver um sistema usando o método de eliminação no exemplo a seguir.

Exemplo: considere o sistema

$$\begin{aligned} 4x_1 - 1x_2 - 1x_3 &= 1 \\ -1x_1 + 2x_2 - 1x_3 &= 1 \\ -1x_1 - 1x_2 + 6.5x_3 &= 1 \end{aligned} \tag{46}$$

Na forma matricial temos:

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 6.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{47}$$

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 6.5 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{48}$$

$$\mathbf{A}^* = \begin{bmatrix} 4 & -1 & -1 & 1 \\ -1 & 2 & -1 & 1 \\ -1 & -1 & 6.5 & 1 \end{bmatrix} \tag{49}$$

A primeira linha será chamada de L_1, a segunda linha de L_2 e a terceira de L_3. Vamos começar.

PASSO 1:

$$L_2 \leftarrow L_2 - (A_{21}/A_{11})L_1 \tag{50}$$

$$L_3 \leftarrow L_3 - (A_{31}/A_{11})L_1 \tag{51}$$

Resulta:

$$\mathbf{A}^* = \begin{bmatrix} 4 & -1 & -1 & 1 \\ 0 & 1.75 & -1.25 & 1.25 \\ 0 & -1.25 & 6.25 & 1.25 \end{bmatrix} \tag{52}$$

PASSO 2:

$$L_3 \leftarrow L_3 - (A_{32}/A_{22})L_2 \tag{53}$$

Resulta:

$$\mathbf{A}^* = \begin{bmatrix} 4 & -1 & -1 & 1 \\ 0 & 1.75 & -1.25 & 1.25 \\ 0 & 0 & 37.5/7 & 15/7 \end{bmatrix} \tag{54}$$

Com isso, nosso sistema passa a ser

$$\begin{bmatrix} 4 & -1 & -1 \\ 0 & 1.75 & -1.25 \\ 0 & 0 & 37.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.25 \\ 15 \end{bmatrix} \tag{55}$$

Transformamos a matriz **A** em uma matriz diagonal superior. O último passo consiste em resolver o problema começando pela última equação.

PASSO 4 (substituição para trás):

$$x_3 = \frac{b_3}{A_{33}} \tag{56}$$

$$x_2 = \frac{b_2 - A_{23}x_3}{A_{22}} \tag{57}$$

$$x_1 = \frac{b_1 - A_{12}x_2 - A_{13}x_3}{A_{11}} \tag{58}$$

Assim:

$$x_3 = 0.4 \tag{59}$$

$$x_2 = \frac{1.25 + 1.25 \times 0.4}{1.75} = 1 \tag{60}$$

$$x_1 = \frac{1 + 1 \times 1 + 1 \times 0.4}{4} = 0.6 \tag{61}$$

Solução final:

$$\mathbf{x} = \begin{bmatrix} 0.6 \\ 1 \\ 0.4 \end{bmatrix} \tag{62}$$

Chegamos na solução exata após um certo número de passos. Esse é um método direto típico.

Nosso foco aqui, porém, será nos **Métodos Iterativos**. Esses métodos são mais vantajosos para os nossos problemas, que envolvem matrizes esparsas. Temos pelo menos 3 vantagens:

- no método iterativo não é necessário montar a matriz **A** (esse é um problema sério em duas e três dimensões);
- com um *chute* inicial bom, o método iterativo converge com algumas poucas iterações;
- o método direto nos dá a solução exata, mas muitas vezes uma solução aproximada já é o suficiente, pois já existe um erro associado à própria aproximação das derivadas no problema.

Nas próximas seções veremos alguns métodos iterativos.

5 Métodos Iterativos

Métodos Iterativos resolvem

$$\mathbf{Ax} = \mathbf{b} \tag{63}$$

por meio de um algoritmo de repetição/iteração. Começamos com uma iteração inicial

$$\mathbf{x}^{(0)} \ . \tag{64}$$

O número entre parênteses indica a iteração. Essa solução, $\mathbf{x}^{(0)}$, juntamente com a equação original, é usada para calcular uma solução melhorada $\mathbf{x}^{(1)}$. Depois, $\mathbf{x}^{(1)}$ é usada para calcular $\mathbf{x}^{(2)}$ e assim sucessivamente. O processo é repetido até que a convergência seja alcançada, ou seja, até que não haja mais variação significativa entre a solução dada em uma iteração e a próxima.

5.1 Método de Jacobi

Considere o nosso sistema com n equações:

$$\sum_{j=1}^{n} A_{ij}x_j = b_i \qquad i = 1, 2, \cdots, n \ . \tag{65}$$

Separando o termo em que $i = j$ do somatório, podemos reescrever esse sistema como:

$$A_{ii}x_i + \sum_{j=1, i\neq j}^{n} A_{ij}x_j = b_i \ . \tag{66}$$

Isolando x_i, temos

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, i\neq j}^{n} A_{ij}x_j \right) \ . \tag{67}$$

Essa última equação sugere um procedimento iterativo para encontrar um valor de $\mathbf{x}$ melhorado, dado um valor inicial (bem parecido com o método do ponto fixo). Seguindo essa ideia:

$$x_i^{(1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, i\neq j}^{n} A_{ij}x_j^{(0)} \right) \ . \tag{68}$$

Depois, com os valores de $\mathbf{x}^{(1)}$, podemos encontrar os valores de $\mathbf{x}^{(2)}$:

$$x_i^{(2)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, i\neq j}^{n} A_{ij}x_j^{(1)} \right) \ . \tag{69}$$

De maneira geral, para conhecendo o valor de $\mathbf{x}^{(ITER)}$, podemos encontrar $\mathbf{x}^{(ITER+1)}$:

$$x_i^{(ITER+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, i\neq j}^{n} A_{ij}x_j^{(ITER)} \right) \ . \tag{70}$$

Esse é o Método de Jacobi. Vamos só trabalhar um pouco mais nessa última equação, para destacar a variação entre iterações consecutivas. Vamos adicionar e subtrair $x_i^{(ITER)}$ no lado direito da equação (70):

$$
\begin{aligned}
x_i^{(ITER+1)} &= x_i^{(ITER)} - x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - \sum_{j=1, i\neq j}^{n} A_{ij}x_j^{(ITER)}\right) \\
&= x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - A_{ii}x_i^{(ITER)} - \sum_{j=1, i\neq j}^{n} A_{ij}x_j^{(ITER)}\right) \\
&= x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n} A_{ij}x_j^{(ITER)}\right)
\end{aligned} \tag{71}
$$

Assim, o **Método de Jacobi** pode ser apresentado como:

$$
x_i^{(ITER+1)} = x_i^{(ITER)} + R_i^{(ITER)} \qquad i = 1, 2, \cdots, n \quad , \tag{72}
$$

em que

$$
R_i^{(ITER)} = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n} A_{ij}x_j^{(ITER)}\right) \quad . \tag{73}
$$

R_i é chamado de **resíduo**. As iterações convergem se a matriz **A** é diagonal dominante. Vamos ver alguns exemplos.

Exemplo 1. Vamos resolver o seguinte sistema usando o método de Jacobi:

$$
\begin{aligned}
4x_1 + 2x_2 + x_3 &= 7 \\
3x_2 + x_3 &= 4 \\
2x_1 + 2x_2 + 5x_3 &= 9
\end{aligned} \tag{74}
$$

Na forma matricial temos:

$$
\begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 1 \\ 2 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 9 \end{bmatrix} \tag{75}
$$

Começando com o chute inicial:

$$
\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{76}
$$

Para a primeira iteração:

$$
\begin{aligned}
R_1^{(0)} &= \frac{1}{A_{11}}\left(b_1 - A_{11}x_1^{(0)} - A_{12}x_2^{(0)} - A_{13}x_3^{(0)}\right) \\
&= \frac{1}{4}\left(7 - 4\times 0 - 2\times 0 - 1\times 0\right) = 1.75
\end{aligned} \tag{77}
$$

$$
x_1^{(1)} = x_1^{(0)} + R_1^{(0)} = 0 + 1.75 = 1.75 \tag{78}
$$

$$
\begin{aligned}
R_2^{(0)} &= \frac{1}{A_{22}}\left(b_2 - A_{21}x_1^{(0)} - A_{22}x_2^{(0)} - A_{23}x_3^{(0)}\right) \\
&= \frac{1}{3}\left(4 - 0\times 0 - 3\times 0 - 1\times 0\right) = 1.33
\end{aligned} \tag{79}
$$

$$x_2^{(1)} = x_2^{(0)} + R_2^{(0)} = 0 + 1.33 = 1.33 \tag{80}$$

$$\begin{aligned} R_3^{(0)} &= \frac{1}{A_{33}}\left(b_3 - A_{31}x_1^{(0)} - A_{32}x_2^{(0)} - A_{33}x_3^{(0)}\right) \\ &= \frac{1}{5}\left(9 - 2\times 0 - 2\times 0 - 5\times 0\right) = 1.80 \end{aligned} \tag{81}$$

$$x_3^{(1)} = x_3^{(0)} + R_3^{(0)} = 0 + 1.80 = 1.80 \tag{82}$$

Portanto, a primeira iteração nos dá

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1.75 \\ 1.33 \\ 1.80 \end{bmatrix} \tag{83}$$

Repetindo os mesmos passos para a segunda iteração, temos

$$\begin{aligned} R_1^{(1)} &= \frac{1}{A_{11}}\left(b_1 - A_{11}x_1^{(1)} - A_{12}x_2^{(1)} - A_{13}x_3^{(1)}\right) \\ &= \frac{1}{4}\left(7 - 4\times 1.75 - 2\times 1.33 - 1\times 1.80\right) = -1.12 \end{aligned} \tag{84}$$

$$x_1^{(2)} = x_1^{(1)} + R_1^{(1)} = 1.75 - 1.12 = 1.63 \tag{85}$$

$$\begin{aligned} R_2^{(1)} &= \frac{1}{A_{22}}\left(b_2 - A_{21}x_1^{(1)} - A_{22}x_2^{(1)} - A_{23}x_3^{(1)}\right) \\ &= \frac{1}{3}\left(4 - 0\times 1.75 - 3\times 1.33 - 1\times 1.80\right) = -0.60 \end{aligned} \tag{86}$$

$$x_2^{(2)} = x_2^{(1)} + R_2^{(1)} = 1.33 - 0.60 = 0.73 \tag{87}$$

$$\begin{aligned} R_3^{(1)} &= \frac{1}{A_{33}}\left(b_3 - A_{31}x_1^{(1)} - A_{32}x_2^{(1)} - A_{33}x_3^{(1)}\right) \\ &= \frac{1}{5}\left(9 - 2\times 1.75 - 2\times 1.33 - 5\times 1.80\right) = -1.23 \end{aligned} \tag{88}$$

$$x_3^{(2)} = x_3^{(1)} + R_3^{(1)} = 1.80 - 1.23 = 0.57 \tag{89}$$

Portanto, a segunda iteração nos dá

$$\mathbf{x}^{(2)} = \begin{bmatrix} 0.63 \\ 0.73 \\ 0.57 \end{bmatrix} \tag{90}$$

Repetindo esse processo iterativo, temos:

$$\mathbf{x}^{(3)} = \begin{bmatrix} 1.24 \\ 1.14 \\ 1.25 \end{bmatrix} \quad \mathbf{x}^{(4)} = \begin{bmatrix} 0.86 \\ 0.92 \\ 0.85 \end{bmatrix} \quad \mathbf{x}^{(5)} = \begin{bmatrix} 1.08 \\ 1.05 \\ 1.09 \end{bmatrix} \quad \mathbf{x}^{(6)} = \begin{bmatrix} 0.95 \\ 0.97 \\ 0.95 \end{bmatrix} \quad \cdots \tag{91}$$

$$\cdots \qquad \mathbf{x}^{(9)} = \begin{bmatrix} 1.01 \\ 1.01 \\ 1.01 \end{bmatrix} \qquad \cdots \tag{92}$$

E assim por diante. Note que as soluções obtidas pelo processo iterativo se aproximam cada vez mais da solução exata,

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{93}$$

mas nunca a alcançam. Outro ponto importante é que essa convergência parece ser muito **lenta**.

Exemplo 2. Vamos resolver agora o seguinte sistema:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 &= 7 \\ 2x_1 + 2x_2 + 5x_3 &= 9 \\ 3x_2 + x_3 &= 4 \end{aligned} \tag{94}$$

Na forma matricial temos:

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 2 & 5 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 9 \\ 4 \end{bmatrix} \tag{95}$$

Essas equações são **idênticas** às equações do exemplo anterior, mas estão em uma ordem diferente. Agora a matriz **A** do sistema não é mais **diagonalmente dominante**.
Vamos resolver, novamente, usando o método de Jacobi, começando com o chute inicial:

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{96}$$

Para a primeira iteração:

$$x_1^{(1)} = x_1^{(0)} + R_1^{(0)} \tag{97}$$

$$\begin{aligned} R_1^{(0)} &= \frac{1}{A_{11}} \left(b_1 - A_{11}x_1^{(0)} - A_{12}x_2^{(0)} - A_{13}x_3^{(0)} \right) \\ &= \frac{1}{4} \left(7 - 4 \times 0 - 2 \times 0 - 1 \times 0 \right) = 1.75 \end{aligned} \tag{98}$$

$$x_2^{(1)} = x_2^{(0)} + R_2^{(0)} \tag{99}$$

$$\begin{aligned} R_2^{(0)} &= \frac{1}{A_{22}} \left(b_2 - A_{21}x_1^{(0)} - A_{22}x_2^{(0)} - A_{23}x_3^{(0)} \right) \\ &= \frac{1}{2} \left(9 - 2 \times 0 - 2 \times 0 - 5 \times 0 \right) = 4.50 \end{aligned} \tag{100}$$

$$x_3^{(1)} = x_3^{(0)} + R_3^{(0)} \tag{101}$$

$$\begin{aligned} R_3^{(0)} &= \frac{1}{A_{33}} \left(b_3 - A_{31}x_1^{(0)} - A_{32}x_2^{(0)} - A_{33}x_3^{(0)} \right) \\ &= \frac{1}{1} \left(4 - 0 \times 0 - 3 \times 0 - 1 \times 0 \right) = 4.00 \end{aligned} \tag{102}$$

Portanto, a primeira iteração nos dá

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1.75 \\ 4.50 \\ 4.00 \end{bmatrix} \tag{103}$$

Repetindo esse processo iterativo, temos:

$$\mathbf{x}^{(2)} = \begin{bmatrix} -1.50 \\ -7.25 \\ -9.50 \end{bmatrix} \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 7.75 \\ 29.75 \\ 25.75 \end{bmatrix} \qquad \mathbf{x}^{(4)} = \begin{bmatrix} -19.5625 \\ -67.625 \\ -85.25 \end{bmatrix} \cdots \tag{104}$$

$$\cdots \qquad \mathbf{x}^{(10)} = \begin{bmatrix} -11533 \\ -39387 \\ -47866 \end{bmatrix} \qquad \cdots \tag{105}$$

E assim por diante. Note que as soluções obtidas **não convergem para a solução exata. Isso acontece porque a matriz dos coeficientes não é diagonal dominante.**

Conclusão: as equações nos dois exemplos são idênticas, mas a forma como são arranjadas é diferente, o que faz com que o método funcione em um caso e não funcione no outro.

5.2 Convergência e Critério de Parada

Ter uma **matriz de coeficientes diagonalmente dominante** é **condição suficiente** para garantir que os métodos de Jacobi, Gauss-Seidel e SOR (esses dois últimos serão apresentados daqui a pouco) convirjam para qualquer chute inicial. Se a matriz dos coeficientes não é diagonalmente dominante, deve ser usado um método direto para resolver o sistema.

O número de iterações para a convergência depende:

- do grau de dominância da diagonal principal;
- do método usado;
- da solução inicial (chute inicial);
- do critério de parada.

5.2.1 Critério de convergência ou parada

Como não sabemos a solução exata, vamos usar a variação entre a solução de uma iteração e a próxima como critério de parada. A **tolerância** ϵ é um valor definido por quem está resolvendo o problema. Temos dois tipos de critérios de parada: os absolutos e os relativos.

Critérios absolutos: você deve parar quando

$$\left|(R_i^{(ITER)})_{MAX}\right| \leq \epsilon \qquad \text{ou} \qquad \sum_{i=1}^{n} \left|R_i^{(ITER)}\right| \leq \epsilon \qquad \text{ou} \qquad \left[\sum_{i=1}^{n} \left(R_i^{(ITER)}\right)^2\right]^{1/2} \leq \epsilon \tag{106}$$

Critérios relativos: você deve parar quando

$$\left|\frac{(R_i^{(ITER)})_{MAX}}{x_i^{(ITER)}}\right| \leq \epsilon \qquad \text{ou} \qquad \sum_{i=1}^{n}\left|\frac{R_i^{(ITER)}}{x_i^{(ITER)}}\right| \leq \epsilon \qquad \text{ou} \qquad \left[\sum_{i=1}^{n}\left(\frac{R_i^{(ITER)}}{x_i^{(ITER)}}\right)^2\right]^{1/2} \leq \epsilon \tag{107}$$

Atenção: use apenas um dos critérios acima em seu código. O critério aparece como condição para sair do *loop* de iterações.

5.3 Método de Gauss-Seidel

O método de Gauss-Seidel é muito parecido com o método de Jacobi. A diferença é que na nova iteração já são utilizados os valores de x_i que já foram atualizados. Vamos ver os detalhes do desenvolvimento desse método.

Considere, novamente, o nosso sistema com n equações:

$$\sum_{j=1}^{n} A_{ij}x_j = b_i \qquad i = 1, 2, \cdots, n \quad . \tag{108}$$

Vamos dividir a somatória em 3 partes: para $i < j$, para $i = j$ e para $i > j$. Assim:

$$\sum_{j=1}^{n-1} A_{ij}x_j + A_{ii}x_i + \sum_{j=i+1}^{n} A_{ij}x_j = b_i \ . \tag{109}$$

Isolando x_i, temos

$$x_i = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n-1} A_{ij}x_j - \sum_{j=i+1}^{n} A_{ij}x_j\right) \ . \tag{110}$$

Essa última equação sugere um procedimento iterativo para encontrar um valor de $\mathbf{x}$ melhorado, dado um valor inicial. No entanto, note que os valores de x_i na primeira somatória do lado direito já foram calculados na nova iteração. Assim, vamos utilizar esses valores novos, em vez de utilizar os valores antigos. Seguindo essa ideia, o processo iterativo é dado por:

$$x_i^{(ITER+1)} = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n-1} A_{ij}x_j^{(ITER+1)} - \sum_{j=i+1}^{n} A_{ij}x_j^{(ITER)}\right) \ . \tag{111}$$

Esse é o Método de Gauss-Seidel. Vamos só trabalhar um pouco mais nessa última equação, para destacar a variação entre iterações consecutivas. Vamos adicionar e subtrair $x_i^{(ITER)}$ no lado direito da equação (111):

$$\begin{aligned} x_i^{(ITER+1)} &= x_i^{(ITER)} - x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n-1} A_{ij}x_j^{(ITER+1)} - \sum_{j=i+1}^{n} A_{ij}x_j^{(ITER)}\right) \\ &= x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n-1} A_{ij}x_j^{(ITER+1)} - A_{ii}x_i^{(ITER)} - \sum_{j=i+1}^{n} A_{ij}x_j^{(ITER)}\right) \\ &= x_i^{(ITER)} + \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{n-1} A_{ij}x_j^{(ITER+1)} - \sum_{j=i}^{n} A_{ij}x_j^{(ITER)}\right) \end{aligned} \tag{112}$$

Assim, o **Método de Gauss-Seidel** pode ser apresentado como:

$$x_i^{(ITER+1)} = x_i^{(ITER)} + R_i^{(ITER)} \qquad i = 1, 2, \cdots, n \tag{113}$$

em que

$$R_i^{(ITER)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{n-1} A_{ij} x_j^{(ITER+1)} - \sum_{j=i}^{n} A_{ij} x_j^{(ITER)} \right) \tag{114}$$

A diferença entre Jacobi e Gauss-Seidel está nessa primeira somatória do lado direito da equação (114). O método de Gauss-Seidel converge mais rapidamente (com um número menor de iterações) porque já utiliza na própria iteração os novos valores recentemente calculados.

No entanto, o método de Jacobi é paralelizável e o método de Gauss-Seidel não. Por quê?

Exemplo. Vamos resolver o seguinte sistema utilizando o método de Gauss-Seidel:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 &= 7 \\ 3x_2 + x_3 &= 4 \\ 2x_1 + 2x_2 + 5x_3 &= 9 \end{aligned} \tag{115}$$

Na forma matricial temos:

$$\begin{bmatrix} 4 & 2 & 1 \\ 0 & 3 & 1 \\ 2 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 9 \end{bmatrix} \tag{116}$$

Vamos resolver usando o método de Gauss-Seidel, começando com o chute inicial:

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{117}$$

Para a primeira iteração:

$$\begin{aligned} R_1^{(0)} &= \frac{1}{A_{11}} \left(b_1 - A_{11} x_1^{(0)} - A_{12} x_2^{(0)} - A_{13} x_3^{(0)} \right) \\ &= \frac{1}{4} \left(7 - 4 \times 0 - 2 \times 0 - 1 \times 0 \right) = 1.75 \end{aligned} \tag{118}$$

$$x_1^{(1)} = x_1^{(0)} + R_1^{(0)} = 0 + 1.75 = 1.75 \tag{119}$$

$$\begin{aligned} R_2^{(0)} &= \frac{1}{A_{22}} \left(b_2 - A_{21} x_1^{(1)} - A_{22} x_2^{(0)} - A_{23} x_3^{(0)} \right) \\ &= \frac{1}{3} \left(4 - 0 \times 1.75 - 3 \times 0 - 1 \times 0 \right) = 1.33 \end{aligned} \tag{120}$$

$$x_2^{(1)} = x_2^{(0)} + R_2^{(0)} = 0 + 1.33 = 1.33 \tag{121}$$

$$\begin{aligned} R_3^{(0)} &= \frac{1}{A_{33}} \left(b_3 - A_{31} x_1^{(1)} - A_{32} x_2^{(1)} - A_{33} x_3^{(0)} \right) \\ &= \frac{1}{5} \left(9 - 2 \times 1.75 - 2 \times 1.33 - 5 \times 0 \right) = 0.57 \end{aligned} \tag{122}$$

$$x_3^{(1)} = x_3^{(0)} + R_3^{(0)} = 0 + 0.57 = 0.57 \tag{123}$$

Portanto, a primeira iteração nos dá

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1.75 \\ 1.33 \\ 0.57 \end{bmatrix} \tag{124}$$

Repetindo os mesmo passos, para a segunda iteração temos

$$\begin{aligned} R_1^{(1)} &= \frac{1}{A_{11}}\left(b_1 - A_{11}x_1^{(1)} - A_{12}x_2^{(1)} - A_{13}x_3^{(1)}\right) \\ &= \frac{1}{4}\left(7 - 4 \times 1.75 - 2 \times 1.33 - 1 \times 0.57\right) = -0.81 \end{aligned} \tag{125}$$

$$x_1^{(2)} = x_1^{(1)} + R_1^{(1)} = 1.75 - 0.81 = 0.94 \tag{126}$$

$$\begin{aligned} R_2^{(1)} &= \frac{1}{A_{22}}\left(b_2 - A_{21}x_1^{(2)} - A_{22}x_2^{(1)} - A_{23}x_3^{(1)}\right) \\ &= \frac{1}{3}\left(4 - 0 \times 0.94 - 3 \times 1.33 - 1 \times 0.57\right) = -0.19 \end{aligned} \tag{127}$$

$$x_2^{(2)} = x_2^{(1)} + R_2^{(1)} = 1.33 - 0.19 = 1.14 \tag{128}$$

$$\begin{aligned} R_3^{(1)} &= \frac{1}{A_{33}}\left(b_3 - A_{31}x_1^{(1)} - A_{32}x_2^{(1)} - A_{33}x_3^{(0)}\right) \\ &= \frac{1}{5}\left(9 - 2 \times 0.94 - 2 \times 1.14 - 5 \times 0.57\right) = 0.40 \end{aligned} \tag{129}$$

$$x_3^{(2)} = x_3^{(1)} + R_3^{(1)} = 0.57 + 0.40 = 0.97 \tag{130}$$

Portanto, a segunda iteração nos dá

$$\mathbf{x}^{(2)} = \begin{bmatrix} 0.94 \\ 1.14 \\ 0.97 \end{bmatrix} \tag{131}$$

Repetindo esse processo iterativo, temos:

$$\mathbf{x}^{(3)} = \begin{bmatrix} 0.94 \\ 1.01 \\ 1.02 \end{bmatrix} \qquad \mathbf{x}^{(4)} = \begin{bmatrix} 0.99 \\ 0.99 \\ 1.01 \end{bmatrix} \qquad \cdots \tag{132}$$

E assim por diante. Note que as soluções obtidas pelo processo iterativo se aproximam cada vez mais da solução exata,

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{133}$$

mas nunca a alcançam (pra ver isso é necessário utilizar mais algarismos). Note também que agora o processo de **convergência** é **muito mais rápido** do que no caso com Jacobi. Aqui, com 4 iterações já temos um erro próximo a 1%. No método de Jacobi foram necessárias 9 iterações para alcançar precisão semelhante.

6 Método SOR

O método de Gauss-Seidel pode ser melhorado com a inclusão de um fator ω para multiplicar o resíduo em cada iteração. Esse novo método é conhecido como SOR (*Successive-over-relaxation*), e é dado por

$$x_i^{(ITER+1)} = x_i^{(ITER)} + \omega R_i^{(ITER)} \qquad i = 1, 2, \cdots, n \tag{134}$$

em que

$$R_i^{(ITER)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{n-1} A_{ij} x_j^{(ITER+1)} - \sum_{j=i}^{n} A_{ij} x_j^{(ITER)} \right) \tag{135}$$

ω é o fator de relaxação e geralmente

$$1 < \omega < 2 \; . \tag{136}$$

Para $\omega = 1$ temos o método de Gauss-Seidel e para $\omega = 2$ o método diverge. O valor ótimo de ω, ou seja, o valor para o qual o número de iterações é mínimo, deve ser determinado por tentativa e erro. Esse valor se aproxima de 2 para grandes sistemas.

7 Implementação

Em anexo é apresentada a implementação em Python dos dois exemplos aqui do texto. Nesses exemplos temos sistemas com 3 equações. **A generalização desses casos e a implementação da solução usando funções do Python vai ficar para vocês.**

Bibliografia

HOFFMAN, J. D. *Numerical Methods for Engineers and Scientists*, 1. ed. Nova Iorque: McGraw-Hill, 1992.

KIUSALAAS, J. *Numerical Methods in Engineering with Python 3*, 1. ed. Nova Iorque: Cambridge University Press, 2013.

Solução de Sistemas de Equações - Implementação

March 17, 2021

1 Solução de Sistemas de Equações Lineares

Vamos resolver aqui o sistema de equações lineares que foi apresentado no texto na forma de exemplo. O sistema é dado por:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 &= 7 \\ 3x_2 + x_3 &= 4 \\ 2x_1 + 2x_2 + 5x_3 &= 9 \end{aligned} \tag{1}$$

Vamos resolver primeiro com o **Método de Jacobi** e depois com o **Método de Gauss-Seidel**.

1.1 Método de Jacobi

O método é dado por

$$x_i^{(ITER+1)} = x_i^{(ITER)} + R_i^{(ITER)} \qquad i = 1, 2, \cdots, n \quad , \tag{2}$$

em que

$$R_i^{(ITER)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{n} A_{ij} x_j^{(ITER)} \right) \quad . \tag{3}$$

Implementação:

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Matriz A (dos coeficientes).
A = np.array([[4.0,2.0,1.0],[0.0,3.0,1.0],[2.0,2.0,5.0]])

# Matriz b (dos termos independentes).
b = np.array([7.0,4.0,9.0])

# Matriz x (das incógnitas).
```

```
x = np.zeros(3,float)

# Matriz x auxiliar para atualizar o valor.
x_new = np.copy(x)

# Resíduo.
R = np.zeros(3,float)

# Opção para que sejam printadas apenas duas
# casas decimais.
np.set_printoptions(precision=2)

# Loop nas iterações. Note que
# estamos fazendo um número
# pré-determinado de iterações.
# O melhor aqui é criar um loop
# while e sair da iteração utilizando
# um dos critérios de parada apresentados
# no texto. Crie você esse loop com o
# critério de parada.
for it in range(10):

    # Loop nas equações.
    for i in range(3):

        # Variável auxiliar para a somatória.
        soma = 0.0

        # Loop para calcular a somatória.
        for j in range(3):
            soma = soma + A[i,j]*x[j]

        # Resíduo
        R[i] = (1.0/A[i,i])*(b[i] - soma)

        # Calculando o novo valor de x
        # e armazenando na matriz auxiliar.
        x_new[i] = x[i] + R[i]

    # Atualizando o valor de x.
    x = np.copy(x_new)

    # Printando na tela os valores da
    # iteração, de R e de x em cada
    # iteração.
    print(f'Iteration {it+1}.')
    print(f'R = {R} ')
```

```
    print(f'x = {x} \n')
```

```
Iteration 1.
R = [1.75 1.33 1.8 ]
x = [1.75 1.33 1.8 ]

Iteration 2.
R = [-1.12 -0.6  -1.23]
x = [0.63 0.73 0.57]

Iteration 3.
R = [0.61 0.41 0.69]
x = [1.24 1.14 1.25]

Iteration 4.
R = [-0.38 -0.23 -0.41]
x = [0.86 0.92 0.85]

Iteration 5.
R = [0.22 0.14 0.24]
x = [1.08 1.05 1.09]

Iteration 6.
R = [-0.13 -0.08 -0.14]
x = [0.95 0.97 0.95]

Iteration 7.
R = [0.08 0.05 0.08]
x = [1.03 1.02 1.03]

Iteration 8.
R = [-0.04 -0.03 -0.05]
x = [0.98 0.99 0.98]

Iteration 9.
R = [0.03 0.02 0.03]
x = [1.01 1.01 1.01]

Iteration 10.
R = [-0.02 -0.01 -0.02]
x = [0.99 1.   0.99]
```

1.2 Método de Gauss-Seidel

O método é dado por:

$$x_i^{(ITER+1)} = x_i^{(ITER)} + R_i^{(ITER)} \qquad i = 1, 2, \cdots, n \tag{4}$$

em que

$$R_i^{(ITER)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{n-1} A_{ij} x_j^{(ITER+1)} - \sum_{j=i}^{n} A_{ij} x_j^{(ITER)} \right) \tag{5}$$

Implementação:

[3]:
```
A = np.array([[4.0,2.0,1.0],[0.0,3.0,1.0],[2.0,2.0,5.0]])

b = np.array([7.0,4.0,9.0])

x = np.zeros(3,float)

R = np.zeros(3,float)

for it in range(10):
    for i in range(3):
        soma = 0.0
        for j in range(3):
            soma = soma + A[i,j]*x[j]

        R[i] = (1.0/A[i,i])*(b[i] - soma)

        # Note aqui a diferença entre
        # Gauss-Seidel e SOR. Por que
        # é assim?
        x[i] = x[i] + R[i]

    print(f'Iteration {it+1}.')
    print(f'R = {R} ')
    print(f'x = {x} \n')
```

```
Iteration 1.
R = [1.75 1.33 0.57]
x = [1.75 1.33 0.57]

Iteration 2.
R = [-0.81 -0.19  0.4 ]
x = [0.94 1.14 0.97]

Iteration 3.
R = [-0.01 -0.13  0.06]
x = [0.94 1.01 1.02]
```

```
Iteration 4.
R = [ 0.05 -0.02 -0.01]
x = [0.99 0.99 1.01]

Iteration 5.
R = [ 0.01  0.   -0.01]
x = [1. 1. 1.]

Iteration 6.
R = [-0.  0. -0.]
x = [1. 1. 1.]

Iteration 7.
R = [-0.  0.  0.]
x = [1. 1. 1.]

Iteration 8.
R = [-1.90e-04 -9.90e-05  1.16e-04]
x = [1. 1. 1.]

Iteration 9.
R = [ 2.06e-05 -3.85e-05  7.15e-06]
x = [1. 1. 1.]

Iteration 10.
R = [ 1.75e-05 -2.38e-06 -6.03e-06]
x = [1. 1. 1.]
```