

Capítulo 2

Objetivos Específicos de Aprendizagem

Ao finalizar este Capítulo, você será capaz de:

- ▶ Executar a implementação computacional de métodos eliminativos para solver sistemas de equações lineares com poucos coeficientes nulos, embora estes tenham problemas de acúmulo de erros de arredondamento, especialmente os sistemas mal condicionados; e
- ▶ Executar a implementação computacional de métodos iterativos clássicos para solver sistemas de equações lineares com muitos coeficientes nulos.

2 Solução Computacional de Sistemas de Equações Lineares



Solver –

Utilizamos a palavra "solver" em vez de "resolver" com o intuito de lembrá-lo que devemos escolher o método mais adequado para obter a solução de determinado problema matemático, assegurar a sua qualidade e não apenas reproduzir resultados.



As obras “Análise Numérica”, de Burden e Faires, e “Numerical Mathematics and Computing”, de Cheney e Kincaid, são excelentes referências para estudo dirigido.

Vamos estudar neste Capítulo a implementação computacional de métodos básicos para **solver** sistemas de equações lineares densos ou esparsos (com muitos coeficientes nulos), de acordo com métodos **clássicos disponíveis na literatura**, objetivando resolver os sistemas que surgem nos modelos matemáticos abordados nos cursos de graduação.

Um sistema de n equações lineares a n incógnitas é toda expressão do tipo:

$$A * X = B \rightarrow \begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases} \rightarrow \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (1)$$

em que A é a matriz de coeficientes, X é o vetor de incógnitas e B é o vetor de termos independentes. Dessas três notações, a primeira apenas facilita o uso algébrico do sistema linear, e a terceira possibilita a sua implementação computacional.

A seguir, um exemplo de sistema de equações lineares:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \end{cases} \quad (2)$$

$$\begin{bmatrix} 0.448 & 0.832 & 0.193 \\ 0.421 & 0.784 & -0.207 \\ -0.319x_1 & 0.884 & 0.279 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (3)$$

Podemos também armazenar as n equações de um sistema de equações lineares através de uma estrutura de dados matricial única. Para tal representação matricial do sistema (2), vamos adotar a estrutura em forma de matriz expandida, que consiste no armazenamento da matriz de coeficientes A concatenada com a coluna de termos independentes B , de forma que cada linha dessa matriz expandida armazene uma equação, conforme segue:

$$[A \ B] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & : & a_{1n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & : & a_{2n+1} \\ : & : & \cdots & : & : \\ a_{n1} & a_{n2} & \cdots & a_{nn} & : & a_{nn+1} \end{bmatrix}$$

Note que, na matriz expandida, cada $a_{i,n+1} = b_i$, $i = 1, 2, \dots, n$.

Nos algoritmos, normalmente a matriz expandida $[A \ B]$ será denotada e armazenada como única matriz A .

A solução de um sistema de equações lineares $A * X = B$ é toda matriz coluna S que o satisfaça, isto é, tal que $A * S = B$. Quanto à sua solução S , um sistema de equações lineares pode ser possível e determinado, quando S for único; possível e indeterminado, quando existirem infinitas soluções S ; e impossível, quando não existir uma solução S .

Desconsiderando o **método direto de Cramer** devido à sua ineficiência, os métodos de solução de sistemas de equações lineares podem ser agrupados em três famílias:

- a) Métodos Eliminativos:** nesta metodologia, transformamos a matriz A do sistema $A * X = B$ em uma matriz mais simples, porém preservando a solução S do sistema original. Isso é sempre possível quando fazemos uso das operações elemen-

O método de Cramer é um teorema da álgebra linear que dá a solução de um sistema de equações lineares em termos de determinantes. Para saber mais, consulte: <https://pt.wikipedia.org/wiki/Regra_de_Cramer>. Acesso em: 26 set. 2016.



tares sobre linhas da matriz representativa do sistema. Tais operações são a troca de linhas, a adição de linhas, bem como a multiplicação de uma linha por uma constante não nula. Da álgebra linear sabemos que tais operações não alteram a solução S do sistema, uma vez que uma operação elementar sobre linha equivale a uma operação sobre a respectiva equação.

- b) Métodos Iterativos:** nesta metodologia, para solver $A * X = B$, inicialmente atribuímos um candidato $X_{(0)}$ para a solução S e posteriormente geramos uma sequência recursiva de vetores coluna X_k , $k = 1, 2, \dots$, cujo limite, se existir, será a solução S procurada.
- c) Métodos de Otimização:** nesta metodologia, partindo do sistema $A * X = B$, geramos uma função matricial $F(X) = X^T * A * X - 2 * B^T * X$ (em que T indica transposta) e tentamos obter o seu mínimo funcional, que é sempre a matriz $A^{-1} * B$, justamente a solução S do sistema linear.

Devido à complexidade algébrica dos Métodos de Otimização, que está fora do escopo deste livro, não apresentaremos nenhum método dessa família.

2.1 Solução Eliminativa de $A * X = B$

A solução eliminativa de $A * X = B$ é normalmente utilizada em sistemas lineares com matrizes **densas** (com poucos coeficientes nulos) e de **ordem n de porte médio** (valor relativo ao processador utilizado).

A seguir, vamos implementar alguns algoritmos de métodos eliminativos mais comuns, avaliar os **erros** que afetam as soluções fornecidas, bem como a sua **complexidade** computacional.

2.1.1 Método de Eliminação de Gauss

O método de **eliminação de Gauss** consiste em aplicar um conjunto de operações elementares em um sistema linear com o objetivo de torná-lo um sistema de resolução mais simples. Usando esse conjunto de operações, podemos alterar sua matriz expandida, de modo que a matriz inicial seja transformada em uma **matriz triangular**, superior ou inferior, visando simplificar o sistema de equações para que a sua solução possa ser obtida diretamente por **substituições retroativas** ou **sucessivas**, respectivamente.

Esse processo de eliminação, também denominado **escalonamento**, é obtido através da aplicação sucessiva de operações elementares sobre linhas na matriz expandida, buscando anular elementos não nulos para torná-la uma matriz triangular. Podemos associá-lo a um processo de pivotamento parcial ou total que promove a troca seletiva de linhas (ou colunas) visando tomar pivôs (elementos das diagonais principais) com maior módulo possível, para evitar a presença de pivôs nulos e minimizar a acumulação de erros de arredondamento.

Exemplo 2.1: resolva o sistema de equações lineares, a seguir, pelo método de eliminação de Gauss (sem trocas de linhas e/ou colunas) adotando operações aritméticas com apenas 4 dígitos significativos totais e arredondamento ponderado, para exemplificar o efeito do acúmulo de arredondamentos.

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \end{cases}$$

Solução:

Na forma matricial, temos:

$$\begin{bmatrix} 0.448 & 0.832 & 0.193 \\ 0.421 & 0.784 & -0.207 \\ -0.319 & 0.884 & 0.279 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

E, na forma de matriz expandida, temos:

$$A = \begin{bmatrix} 0.448 & 0.832 & 0.193 & : & 1 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ -0.319 & 0.884 & 0.279 & : & 0 \end{bmatrix}$$

Processo de Triangularização

O processo de triangularização deve seguir uma sequência otimizada de passos k (que definem um **algoritmo**). Nesse caso, cada passo k corresponderá à utilização da linha $i = k$ para eliminar os coeficientes da coluna $j = k$ usando operações elementares sobre linhas.

Os índices comumente utilizados para acessar um elemento de uma matriz a_{ij} são: i , para acessar a linha (1^{o} índice); e j , para acessar a coluna (2^{o} índice), enquanto o índice k vai definir os passos do algoritmo. A i -ésima linha é denotada por L_i , e a j -ésima coluna é denotada por C_j .

Vamos acompanhar os passos de resolução do **Exemplo 2.1** para depois estabelecê-los na forma de algoritmo computacional.

Primeiro passo: definido pelo índice $k=1$, usamos a primeira linha $i=1$ (em negrito) para eliminar a primeira coluna $j=1$ das linhas abaixo da linha $i=1$. Poderíamos estabelecer outra ordem para as operações elementares sobre linhas, mas vamos adotar um algoritmo genérico que independa dos valores dos coeficientes e use o menor número de operações aritméticas possível. Então, nesse primeiro passo, substituímos as linhas $i=2$ e $i=3$ pelo resultado da subtração delas próprias com a linha $i=1$ multiplicada por um fator adequado, de tal modo que as suas colunas $j=1$ sejam zeradas, logo:

$$\left[\begin{array}{ccc|c} \mathbf{0.448} & \mathbf{0.832} & \mathbf{0.193} & : 1 \\ 0.421 & 0.784 & -0.207 & : 0 \\ -0.319 & 0.884 & 0.279 & : 0 \end{array} \right] \begin{aligned} L_2 &\leftarrow L_2 - (0.421 / \mathbf{0.448})L_1 \Rightarrow L_2 \leftarrow L_2 - 0.9397L_1 \\ L_3 &\leftarrow L_3 - (-0.319 / \mathbf{0.448})L_1 \Rightarrow L_3 \leftarrow L_3 + 0.7121L_1 \end{aligned}$$

Observações:

- a) Na linha L_3 , por exemplo,

$$L_3 \leftarrow L_3 - (-0.319 / 0.448)L_1 \Rightarrow L_3 \leftarrow L_3 + 0.7121 * L_1,$$

no elemento $i=3$ e $j=1$, teremos a seguinte operação de eliminação: $a_{31} = a_{31} - (-0.319 / 0.448)a_{11}$

$a_{31} = -0.319 - (-0.319 / 0.448)0.448 = 0$ (na ausência de arredondamentos).

Então, podemos definir o fator multiplicativo sempre como a razão entre o valor do coeficiente que deve ser zerado e o valor do coeficiente pivô (diagonal principal) da linha correspondente ao passo, $i=k$. Esse mesmo fator multiplicativo deve ser aplicado na operação elementar sobre toda a linha $i=3$, em todas as suas colunas j .

- b) As linhas 2 e 3 sofrem alterações através da adição dos termos $-0.9397 * L_1$ e $0.7121 * L_1$, ou seja, se esses termos adicionais tiverem arredondamentos, serão levados para as linhas 2 e 3 resultantes.
- c) Como esses mesmos fatores multiplicativos devem ser aplicados para todas as colunas j das linhas $i=2$ e $i=3$, é importante que sejam determinados uma única vez, armazenados e usados em todas as colunas de cada linha i .
- d) Esses fatores multiplicativos, aplicados previamente na linha $i=1$, -0.9397 e 0.7121 , sofreram arredondamentos no quarto dígito decimal (quarto dígito significativo), logo levarão esse erro no quarto dígito decimal para as novas linhas 2 e 3 geradas a seguir.

Aplicando essas operações elementares sobre a linhas $i=2$ e $i=3$, zeraremos a coluna $j=1$, conforme segue:

$$\left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : 1 \\ 0 & 0.002170 & -0.3884 & : -0.9397 \\ 0 & 1.476 & 0.4164 & : 0.7121 \end{array} \right]$$

Note que as linhas 2 e 3 foram alteradas pelos arredondamentos gerados, mas, na coluna $j=1$, os resultados previstos são exatos e nulos, logo não é necessário efetuar essas operações aritméticas, basta atribuir esses resultados nulos no algoritmo.

Segundo passo: definido pelo índice $k=2$, usamos a segunda linha $i=2$ (em negrito) para eliminar a segunda coluna $j=2$:

$$\left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : & 1 \\ \mathbf{0} & \mathbf{0.002170} & -0.3884 & : & -0.9397 \\ 0 & 1.476 & 0.4604 & : & 0.9397 \end{array} \right] \begin{matrix} L_3 \leftarrow L_3 - (1.476 / \mathbf{0.002170})L_2 \\ L_3 \leftarrow L_3 - 680.2L_2 \end{matrix}$$

Observe que o fator multiplicativo na linha 2 é muito maior do que a unidade, 680.2, então sofreu arredondamentos no primeiro dígito decimal (quarto dígito significativo), logo levará esse erro no primeiro dígito decimal para a nova linha 3, gerada a seguir:

$$\left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : & 1 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \\ 0 & 0 & 264.6 & : & -639.9 \end{array} \right]$$

Observe também que a linha 3 sofreu eliminações duas vezes, nos passos $k=1$ e $k=2$, e isso caracteriza um processo cumulativo de operações aritméticas com erros de arredondamento. Nessa fase, o sistema simplificado já está na forma de matriz triangular superior, que deve ser equivalente ao sistema original apresentado nesse exemplo, pois as operações elementares sobre linhas aplicadas não podem alterar as equações.

Então, a solução do sistema original:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \end{cases}$$

deveria ser a mesma do sistema de equações simplificado a seguir:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0x_1 + 0.002170x_2 - 0.3884x_3 = -0.9397 \\ 0x_1 + 0x_2 + 264.6x_3 = 639.9 \end{cases}$$

Processo de Retrosubstituição Sucessiva

Realizamos o processo de retrosubstituição sucessiva a partir da última equação, determinando a última incógnita e substituindo-a na equação anterior, e assim sucessivamente até obtermos todas as incógnitas.

Primeiro passo: note que o valor de x_3 pode ser diretamente obtido a partir da equação 3, uma vez que essa equação independe de x_1 e x_2 :

$$0x_1 + 0x_2 + 264.6x_3 = 639.9 \quad x_3 = 639.9 / 264.6 \quad x_3 = 2.418$$

Segundo passo: posteriormente, podemos obter os valores de x_2 e x_1 das equações 2 e 1, respectivamente:

$$x_2 = (-0.9397 - (-0.3884x_3)) / 0.002170 \quad x_2 = -0.2529 \text{ (com } x_3 = 2.418\text{)}$$

$$x_1 = (1 - 0.832x_2 - 0.193x_3) / 0.448 \quad x_1 = 1.660 \text{ (com } x_3 = 2.418 \text{ e } x_2 = -0.2529\text{)}$$

Portanto, a solução obtida para o sistema do **Exemplo 2.1**, com precisão de apenas 4 dígitos significativos, é:

$$S = \{ 1.660, -0.2529, 2.418 \}$$

Lembre-se que essa solução S foi obtida de um sistema simplificado, cujos coeficientes já sofreram acúmulo de arredondamentos, e não do sistema original.

Se os resíduos ($R = |A \cdot S - B|$) de cada uma das equações do sistema linear proposto fossem obtidos para aferição da solução, deveríamos ter valores nulos, mas normalmente obtemos valores residuais, não nulos, decorrentes dos arredondamentos acumulados. Por exemplo, se substituirmos a solução S em cada equação original do **Exemplo 2.1**, teremos os seguintes resíduos:

$$\begin{aligned} R_1 &= |0.448x_1 + 0.832x_2 + 0.193x_3 - 1| = 0.0000588 \cong 0.00006 \\ R_2 &= |0.421x_1 + 0.784x_2 - 0.207x_3 - 0| = 0.0000604 \cong 0.00006 \\ R_3 &= |-0.319x_1 + 0.884x_2 + 0.279x_3 - 0| = 0.07848 \cong 0.08 \end{aligned}$$

O resíduo da equação 3 ficou alto, equivalente ao primeiro dígito decimal (0.08 \cong 0.1). Seria esperado um resíduo no quarto dígito decimal, pois temos uma precisão de 4 dígitos significativos. Esse erro decorre principalmente dos arredondamentos dos fatores multiplicativos de cada linha. No próximo exemplo, vamos aplicar uma forma de reduzir esse resíduo.

Para comparação, também podemos calcular o erro de arredondamento estimado sobre cada x_i da solução usando uma estimativa do valor exato da solução S_{exato} . Nesse método, temos **somente erros de arredondamento** associados à solução, então o seu valor exato estimado pode ser obtido usando variáveis com mais precisão, com mais de 4 dígitos significativos, para ter menos arredondamentos. Na solução, a seguir, usamos o mesmo algoritmo aplicado no **Exemplo 2.1**, mas operamos com as variáveis *double* de 16 dígitos significativos, gerando os seguintes resultados:

$$S_{\text{exato}} = \{1.561414494051542, -0.199881764024819, 2.418590333334499\}$$

Os resíduos dessa solução exata estimada, obtida com precisão *double*, são:

$$\begin{aligned} R_{\text{exato}} &= \{0000000000e+00, 1.11022302462e-16, 5.68434188608e-14\} \\ &\quad (\text{em notação científica}) \end{aligned}$$

E os erros estimados, $\text{Erro} = |S - S_{\text{exato}}|$, são:

$$\text{Erro } x_1 = |1.660 - 1.561414494051542| = 0.098585505948458 \approx 0.1$$

$$\text{Erro } x_2 = |-0.2529 - (-0.199881764024819)| = -0.053018235975181 \approx 0.05$$

$$\text{Erro } x_3 = |2.418 - 2.418590333334499| = 0.000590333334499 \approx 0.0006$$

Perceba que os erros de arredondamento acumulados no processo de eliminação foram propagados também para o primeiro dígito decimal, nos valores de x_1 e x_2 , e, para o quarto dígito, no valor do x_3 . Observe que os arredondamentos afetam toda a solução, pois em um sistema de equações não temos como dizer que um valor de x é mais exato do que outro. Assim, a solução está afetada no primeiro dígito.

Podemos resumir o algoritmo de **eliminação gaussiana** no seguinte formulário:

Triangularização:

$$k=1, \dots, n-1 \quad (\text{define o passo } k)$$

$$i=k+1, \dots, n \quad (\text{define linhas } i \text{ a serem zeradas})$$

$$j=k+1, \dots, n+1 \quad (\text{define colunas } j \text{ operadas em cada linha } i)$$

$$a_{ij} = a_{ij} - \left(\frac{a_{ik}}{a_{kk}} \right) a_{kj} \quad (\text{corresponde a } L_i \leftarrow L_i - \left(\frac{a_{ik}}{a_{kk}} \right) L_k)$$

Retrosubstituição:

$$i=n$$

$$x_i = a_{i,n+1} / a_{ii} \quad (\text{corresponde à resolução da última equação})$$

$$i=n-1, \dots, 1$$

$$x_i = \left(a_{i,n+1} - \sum_{j=i+1}^n a_{ij} * x_j \right) / a_{ii} \quad (\text{corresponde à resolução da } i\text{-ésima equação})$$

No próximo exemplo, vamos implementar o processo de **pivotação parcial** visando evitar linhas com possíveis zeros na diagonal principal de cada linha $i=k$ e ainda minimizar erros de arredondamento acumulados.

2.1.1.1 Pivotação Parcial

A **pivotação parcial** consiste em:

- a) no início de cada passo k das eliminações, escolher para a_{kk} o elemento de maior módulo entre os coeficientes a_{ik} , $i=k, k+1, \dots, n$; e
- b) trocar as linhas k e i , se necessário.

Exemplo 2.2: resolva o sistema de equações lineares, a seguir, pelo método de eliminação de Gauss com **pivotamento parcial** utilizando operações aritméticas com 4 dígitos significativos e arredondamento ponderado. Esse sistema é o mesmo do **Exemplo 2.1**, apenas com as equações em outra ordem.

$$\begin{cases} -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \end{cases}$$

Solução:

Na forma de matriz expandida, temos:

$$A = \begin{bmatrix} -0.319 & 0.884 & 0.279 & : & 0 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ 0.448 & 0.832 & 0.193 & : & 1 \end{bmatrix}$$

Processo de Triangularização

Antes de promover o processo de triangularização, devemos escolher e reposicionar a melhor linha $i=k$ possível.

Primeiro passo: definido pelo índice $k=1$, a **pivotação parcial** deve redefinir a melhor linha $i=1$, correspondente ao primeiro pivô ($k=1$). Para tal:

- a) Localizamos o maior módulo da coluna, $j=1$ nesse caso, entre os coeficientes destacados em **negrito**:

$$j = 1$$

$$i = 3 \begin{bmatrix} \mathbf{-0.319} & 0.884 & 0.279 & 0 \\ \mathbf{0.421} & 0.784 & -0.207 & 0 \\ \mathbf{(0.448)} & 0.832 & 0.193 & 1 \end{bmatrix} \quad (\text{maior módulo está na linha } i=3)$$

- b) Trocamos linhas, entre a linha $i=3$, com o melhor coeficiente, e a linha $i=1$:

$$\begin{bmatrix} -0.319 & 0.884 & 0.279 & 0 \\ 0.421 & 0.784 & -0.207 & 0 \\ 0.448 & 0.832 & 0.193 & 1 \end{bmatrix} \begin{array}{l} L_1 \leftarrow L_3 \\ L_3 \leftarrow L_1 \end{array} \quad (\text{troca da linha } L_1 \text{ com } L_3 \text{ e vice-versa})$$

- c) Geramos a matriz pivotada:

$$\begin{bmatrix} 0.448 & 0.832 & 0.193 & : & 1 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ -0.319 & 0.884 & 0.279 & : & 0 \end{bmatrix}$$

Observe que, na pivotação parcial, apenas trocamos a ordem das equações mudando-as de linhas.

- d) Depois da pivotação, procedemos à **eliminação gaussiana** correspondente ao primeiro passo ($k=1$):

$$\begin{bmatrix} 0.448 & 0.832 & 0.193 & : & 1 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ -0.319 & 0.884 & 0.279 & : & 0 \end{bmatrix} \begin{array}{l} L_2 \leftarrow L_2 - (0.421 / 0.448)L_1 \Rightarrow L_2 \leftarrow L_2 - 0.9397L_1 \\ L_3 \leftarrow L_3 - (-0.319 / 0.448)L_1 \Rightarrow L_3 \leftarrow L_3 + 0.7121L_1 \end{array}$$

- e) Geramos nova matriz escalonada:

$$\begin{bmatrix} 0.448 & 0.832 & 0.193 & : & 1 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \\ 0 & 1.476 & 0.4164 & : & 0.7121 \end{bmatrix}$$

Segundo passo: definido pelo índice $k=2$, a **pivotação parcial** deve redefinir a melhor linha $i=2$, correspondente ao segundo pivô ($k=2$). Para tal:

- a) Realizamos a busca parcial do maior módulo da coluna $j=2$ (busca a partir da segunda linha, em **negrito**, pois a primeira linha já foi usada para anular a primeira coluna no passo anterior e não pode ser usada novamente para anular a segunda coluna, uma vez que alteraria a coluna $j=1$ já zerada):

$$j = 2$$

$$i = 3 \begin{bmatrix} 0.448 & 0.832 & 0.193 & : & 1 \\ 0 & \mathbf{0.002170} & -0.3884 & : & -0.9397 \\ 0 & \mathbf{(1.476)} & 0.4164 & : & 0.7121 \end{bmatrix} \quad (\text{maior módulo em } i=3)$$

b) Trocamos linhas, entre a linha $i=3$ e a linha $i=2$:

$$\left[\begin{array}{ccc|cc} (0.448) & 0.832 & 0.193 & : & 1 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \\ 0 & (1.476) & 0.4164 & : & 0.7121 \end{array} \right] \begin{matrix} L_2 \leftarrow L_3 \\ L_3 \leftarrow L_2 \end{matrix}$$

c) Geramos a matriz escalonada:

$$\left[\begin{array}{ccc|cc} (0.448) & 0.832 & 0.193 & : & 1 \\ 0 & (1.476) & 0.4164 & : & 0.7121 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \end{array} \right]$$

Observe que, no **Exemplo 2.1**, o pivô antigo, $a_{22} = 0.002170$, era um valor pequeno e gerava um fator multiplicativo elevado, 680.2, que carregava um erro de arredondamento no primeiro dígito fracionário (quarto significativo).

d) Procedemos a triangularização, correspondente ao segundo passo ($k=2$):

$$\left[\begin{array}{ccc|cc} (0.448) & 0.832 & 0.1930 & : & 1 \\ 0 & (1.476) & 0.4164 & : & 0.7121 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \end{array} \right] \begin{matrix} L_3 \leftarrow L_3 - (0.002170 / 1.476)L_2 \\ L_3 - 0.001470L_2 \end{matrix}$$

Agora o novo pivô, $a_{22} = 1.476$, é um valor maior e gera um fator multiplicativo menor, 0.001470, que, nesse caso, sofreu arredondamentos no sexto dígito fracionário (quarto significativo), logo levará apenas esses erros para a nova linha 3 gerada, conforme segue:

$$\left[\begin{array}{ccc|cc} (0.448) & 0.832 & 0.1930 & : & 1 \\ 0 & (1.476) & 0.4164 & : & 0.7121 \\ 0 & 0 & -0.3893 & : & -0.9412 \end{array} \right]$$

Agora a nova linha $i=3$ ficou com valores próximos aos anteriores à eliminação deste passo, ou seja, a linha $i=3$ foi pouco alterada.

Processo de Retrosubstituições Sucessivas

O sistema de equações simplificado do **Exemplo 2.2**, gerado a partir do processo de eliminação gaussiana, é:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0x_1 + 1.476x_2 + 0.4164x_3 = 0.7121 \\ 0x_1 + 0x_2 - 0.3893x_3 = -0.9412 \end{cases}$$

Note que podemos obter diretamente o valor de x_3 a partir da equação 3, e posteriormente os de x_2 e x_1 a partir dos valores obtidos anteriormente.

$$\begin{array}{ll} x_3 = -0.9412 / (-0.3893) & x_3 = 2.418 \\ x_2 = (0.7121 - 0.4164 x_3) / 1.476 & x_2 = -0.1997 \\ x_1 = (1 - 0.832 x_2 - 0.193 x_3) / 0.448 & x_1 = 1.561 \end{array}$$

Portanto, a solução do sistema dado no **Exemplo 2.2** é:

$$S = \{ 1.561, -0.1997, 2.418 \}$$

Os resíduos ($R = |A * S - B|$) dessa solução são:

$$\begin{array}{ll} R_1 = |-0.319x_1 + 0.884x_2 + 0.279x_3 - 0| = 0.0001282 \approx 0.0001 \\ R_2 = |0.421x_1 + 0.784x_2 - 0.207x_3 - 0| = 0.0000902 \approx 0.0001 \\ R_3 = |0.448x_1 + 0.832x_2 + 0.193x_3 - 1| = 0.0001484 \approx 0.0001 \end{array}$$

Pelos resíduos informados, observamos que a solução obtida no **Exemplo 2.2** está mais próxima da exata do que a obtida no **Exemplo 2.1**, pois os resíduos estão no quarto dígito decimal, enquanto no **Exemplo 2.1** os resíduos chegavam ao primeiro dígito.

Nesse caso, também podemos calcular o erro de arredondamento comparando a solução obtida com precisão de 4 dígitos a uma solução mais precisa obtida com o mesmo algoritmo usando precisão *double*:

$$S_{\text{exato}} = \{ 1.561414494051471, -0.199881764024781, 2.418590333334500 \}$$

Observe que os erros estimados, $\text{Erro} = |S - S_{\text{exato}}|$, são:

$$\text{Erro } x_1 = |1.561 - 1.561414494051471| = 0.000414494051471 \approx 0.0004$$

$$\text{Erro } x_2 = |-0.1997 - (-0.199881764024781)| = 0.000181764024781 \approx 0.0002$$

$$\text{Erro } x_3 = |2.418 - 2.418590333334500| = 0.000590333334500 \approx 0.0006$$

Agora, perceba que os erros estimados de S também foram menores no **Exemplo 2.2** com pivotação parcial, ou seja, a solução S com pivotação parcial acumulou menos erros de arredondamento.

Com o processo de pivotamento parcial:

- a)eliminamos possíveis pivôs nulos, caso haja possibilidades de troca de linhas; e
- b)conseguimos uma redução nos efeitos cumulativos de erros de arredondamento (diminuição da perda de significação), pois os novos pivôs são os maiores possíveis de cada coluna, gerando menores fatores multiplicativos, cujos arredondamentos ocorrem em dígitos menos significativos.

Na próxima seção, vamos implementar o processo de **pivotação total** visando obter soluções computacionalmente mais estáveis em relação às perturbações introduzidas por arredondamentos.

2.1.1.2 Pivotação Total

Alternativamente, podemos implementar o método de eliminação de Gauss usando a **pivotação total**, que é computacionalmente um pouco mais eficiente, induzindo normalmente a um menor erro de arredondamento acumulado, de forma a obter soluções mais estáveis em relação às perturbações introduzidas por arredondamentos.

No **pivotamento total, ou completo**, procuramos o elemento de maior módulo entre todos os elementos disponíveis na matriz de coeficientes promovendo trocas de linhas e colunas. Para avaliar as consequências dessas trocas de linhas e colunas, devemos interpretar os elementos da matriz expandida como termos das equações do sistema, assim:

- a) a troca de **linhas** significa apenas trocar a ordem na apresentação das **equações**; e
- b) a troca de **colunas** significa trocar a ordem de apresentação das **incógnitas** do sistema.

Exemplo 2.3: resolva o sistema de equações lineares, a seguir, usando **pivotação total**, operações aritméticas com 4 dígitos significativos e arredondamento ponderado.

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \end{cases}$$

Solução:

Na forma de matriz expandida, temos:

$$A = \left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : & 1 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ -0.319 & 0.884 & 0.279 & : & 0 \end{array} \right]$$

Primeiro passo: na pivotação total, correspondente ao primeiro pivô ($k=1$):

- a) Buscamos o maior módulo entre todos os elementos da matriz de coeficientes:

$$j = 2$$

$$i = 3 \left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : & 1 \\ 0.421 & 0.784 & -0.207 & : & 0 \\ \mathbf{-0.319} & \mathbf{0.884} & 0.279 & : & 0 \end{array} \right]$$

$$x_1 \quad x_2 \quad x_3$$

(o coeficiente de maior módulo está na coluna $j=2$ e na linha $i=3$, em negrito).

Observe que foi acoplada uma linha adicional, embaixo da matriz de coeficientes, para o armazenamento da ordem de apresentação das incógnitas envolvidas. Então, inicialmente temos a ordem natural das incógnitas x_1 , x_2 e x_3 , na primeira, segunda e terceira colunas.

b) Efetuamos a troca de linhas e colunas:

$$\left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : 1 \\ 0.421 & 0.784 & -0.207 & : 0 \\ -0.319 & \mathbf{0.884} & 0.279 & : 0 \end{array} \right] \begin{array}{l} L_1 \leftarrow L_3 \\ L_3 \leftarrow L_1 \end{array}$$

$$\begin{array}{ccc} x_1 & x_2 & x_3 \\ C_1 & C_2 & \\ \uparrow & \uparrow & \\ C_2 & C_1 & \end{array}$$

Trocamos a linha L_1 com L_3 e vice-versa, e a coluna C_1 com C_2 e vice-versa.

c) Geramos a matriz pivotada:

$$\left[\begin{array}{ccc|c} \mathbf{0.884} & -0.319 & 0.279 & : 0 \\ 0.784 & 0.421 & -0.207 & : 0 \\ 0.832 & 0.448 & 0.193 & : 1 \end{array} \right]$$

$$\begin{array}{ccc} x_2 & x_1 & x_3 \end{array}$$

Note que, no processo de pivotamento total, a ordem de apresentação das incógnitas x_i envolvidas foi alterada para x_2 , x_1 e x_3 .

Segundo passo: no processo de triangularização, correspondente ao primeiro passo ($k=1$), temos:

$$\left[\begin{array}{ccc|c} \mathbf{0.884} & -0.319 & 0.279 & : 0 \\ 0.784 & 0.421 & -0.207 & : 0 \\ 0.832 & 0.448 & 0.193 & : 1 \end{array} \right] \begin{array}{l} L_2 \leftarrow L_2 - (0.784 / 0.884) L_1 \Rightarrow L_2 \leftarrow L_2 - 0.8869 L_1 \\ L_3 \leftarrow L_3 - (0.832 / 0.884) L_1 \Rightarrow L_3 \leftarrow L_3 - 0.9412 L_1 \end{array}$$

$$\begin{array}{ccc} x_2 & x_1 & x_3 \end{array}$$

$$\left[\begin{array}{ccc|c} 0.884 & -0.319 & 0.279 & : 0 \\ 0 & 0.7039 & -0.4544 & : 0 \\ 0 & 0.7482 & -0.06959 & : 1 \end{array} \right]$$

$$\begin{array}{ccc} x_2 & x_1 & x_3 \end{array}$$

Terceiro passo: na pivotação total, correspondente ao segundo pivô ($k=2$):

- a) Buscamos o maior módulo entre os elementos da matriz, a partir da segunda linha e segunda coluna:

$$j = 2$$

$$i = 3 \begin{bmatrix} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & 0.7039 & -0.4544 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \end{bmatrix}$$

$$x_2 \quad x_1 \quad x_3$$

(elemento de maior módulo localizado na coluna $j=2$ e linha $i=3$).

- b) Efetuamos somente troca de linhas, uma vez que, na coluna, a matriz já está naturalmente pivotada:

$$\begin{bmatrix} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & 0.7039 & -0.4544 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \end{bmatrix} \begin{array}{l} L_2 \leftarrow L_3 \\ L_3 \leftarrow L_2 \end{array}$$

$$x_2 \quad x_1 \quad x_3$$

- c) Geramos a matriz pivotada:

$$\begin{bmatrix} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \\ 0 & 0.7039 & -0.4544 & : & 0 \end{bmatrix}$$

$$x_2 \quad x_1 \quad x_3$$

Quarto passo: no processo de triangularização, correspondente ao segundo passo ($k=2$), temos:

$$\begin{bmatrix} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \\ 0 & 0.7039 & -0.4544 & : & 0 \end{bmatrix} \begin{array}{l} L_3 \leftarrow L_3 - (0.7039 / 0.7482)L_2 \\ L_3 \leftarrow L_3 - 0.9408L_2 \end{array}$$

$$x_2 \quad x_1 \quad x_3 \quad L_3 \leftarrow L_3 - 0.9408L_2$$

$$\begin{bmatrix} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \\ 0 & 0 & -0.3889 & : & -0.9408 \end{bmatrix}$$

$$x_2 \quad x_1 \quad x_3$$

Quinto passo: no processo de retrosubstituição, temos:

$$\left[\begin{array}{ccc|cc} 0.884 & -0.319 & 0.279 & : & 0 \\ 0 & \mathbf{0.7482} & -0.06959 & : & 1 \\ 0 & 0 & -0.3889 & : & -0.9408 \end{array} \right] \\ x_2 \quad x_1 \quad x_3$$

Retornando à forma de equações (note a nova ordem das incógnitas):

$$\begin{cases} 0.884x_2 - 0.319x_1 + 0.279x_3 = 0 \\ 0x_2 + 0.7482x_1 - 0.06959x_3 = 1 \\ 0x_2 + 0x_1 - 0.3889x_3 = -0.9408 \end{cases}$$

$$\begin{aligned} x_3 &= -0.9408 / (-0.3889) & x_3 &= 2.419 \\ x_1 &= (1 + 0.06959 x_3) / 0.7482 & x_1 &= 1.561 \\ x_2 &= (0 + 0.319 x_1 - 0.279 x_3) / 0.884 & x_2 &= -0.2002 \end{aligned}$$

Neste ponto, precisamos reordenar a solução sabendo que a ordem das incógnitas obtidas na retrosubstituição é $\{x_2, x_1, x_3\}$, enquanto a solução ordenada é $S = \{x_1, x_2, x_3\} = \{1.561, -0.2002, 2.419\}$.

Os resíduos ($R = |A^*S - B|$) dessa solução são:

$$\begin{aligned} R_1 &= |0.448x_1 + 0.832x_2 + 0.193x_3 - 1| = 0.0005 \\ R_2 &= |0.421x_1 + 0.784x_2 - 0.207x_3 - 0| = 0.0003 \\ R_3 &= |-0.319x_1 + 0.884x_2 + 0.279x_3 - 0| = 0.0003 \end{aligned}$$

E os erros estimados, $\text{Erro} = |S - S_{\text{exato}}|$, obtidos foram:

$$\begin{aligned} \text{Erro } x_1 &= |1.561 - 1.561414494051471| = 0.000414494051471 \approx 0.0004 \\ \text{Erro } x_2 &= |-0.2002 - (-0.199881764024781)| = -0.000318235975219 \approx 0.0003 \\ \text{Erro } x_3 &= |2.419 - 2.418590333334500| = 0.000590333334500 \approx 0.0006 \end{aligned}$$

Os resultados obtidos com pivotação total no **Exemplo 2.3** são equivalentes aos obtidos com pivotação parcial no **Exemplo 2.2**, considerando os resíduos e os erros estimados. Estatisticamente, temos poucos tipos de sistemas que têm solução melhor com pivotação total em relação à sua solução com pivotação parcial.

2.1.1.3 Quantitativo de Soluções de $A * X = B$

As soluções dos sistemas de equações lineares podem ser classificadas segundo três possibilidades distintas:

- a) **Sistema Determinado:** quando o sistema de equações tem solução única, a matriz de coeficientes é não singular, isto é, o seu determinante é não nulo. Ocorre quando todas as equações do sistemas são linearmente independentes, ou seja, nenhuma equação é combinação linear de outras.
- b) **Sistema Indeterminado:** quando o sistema de equações tem infinitas soluções, a matriz de coeficientes é singular, ou seja, o seu determinante é nulo. Ocorre quando temos um sistema de equações lineares cujos coeficientes possuem alguma relação de dependência, por exemplo, uma equação é gerada a partir da combinação linear de outra(s), ou já temos menos equações do que incógnitas. Podemos concluir que a(s) equação(ões) gerada(s) a partir da combinação linear de outras existentes não acrescenta(m) informação(ões) nova(s) ao conjunto de equações do sistema. Dessa forma, o sistema se comporta como se tivesse menos equações do que incógnitas, deixando alguma(s) incógnita(s) livre(s) de equações próprias que restrinjam o seu(s) valor(es). No método de eliminação de Gauss, podemos constatar esse fato observando que, no final do processo de eliminação, uma ou mais linhas inteiras da matriz expandida se anulam, inclusive seu termo independente se anula.
- c) **Sistema Impossível:** quando o sistema de equações não tem solução alguma, a matriz de coeficientes também é singular. Ocorre quando alguma equação do sistema é impossível de ser satisfeita. Por exemplo, se alguma equação do sistema é gerada a partir da combinação linear “parcial”, de apenas um dos lados de duas outras equações, então geramos uma equação inconsistente. No método de Gauss, podemos constatar esse fato observando que, no final do processo de eliminação, uma ou mais linhas da matriz de coeficientes se anulam, mas o termo independente não se anula. Na prática, significa ter uma equação com todos os coeficientes

de x nulos e com um termo independente não nulo, o que é impossível de ser satisfeito.

Nas possibilidades (b) e (c), os elementos nulos podem não ser necessariamente iguais a zero, mas podem assumir valores residuais.

O algoritmo de retrosubstituições adotado neste livro tratará dessas possibilidades.

Um **sistema possível** de resolver, mas **indeterminado**, se comporta conforme o exemplo a seguir:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \quad (\text{com } L_3 = L_2) \end{cases}$$

Se aplicarmos a eliminação gaussiana, chegaremos à seguinte matriz triangular superior na forma expandida:

$$\left[\begin{array}{ccc|cc} 0.448 & 0.832 & 0.193 & : & 1 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \\ 0 & 0 & 0 & : & 0 \end{array} \right]$$

(usando aritmética exata em i=3)

Observe que a terceira linha, ou equação, foi completamente eliminada no segundo passo, $k=2$, não contribuindo mais com a solução do sistema. Logo, temos um sistema efetivo de $n=2$ equações, mas com três incógnitas:

$$\begin{aligned} & \begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0x_1 + 0.002170x_2 - 0.3884x_3 = -0.9397 \\ 0x_1 + 0x_2 + 0x_3 = 0 \end{cases} \\ \Rightarrow & \begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0x_1 + 0.002170x_2 - 0.3884x_3 = -0.9397 \end{cases} \end{aligned}$$

Assim, o nosso algoritmo da função retrosubstituição deve tratar sistemas com uma linha a menos do que o número de incógnitas, ou seja, com uma equação sendo combinação linear de outra(s),

cujo coeficiente da diagonal principal da última linha e o seu termo independente sejam numericamente nulos. Desse modo, se esses coeficientes tiverem valores residuais, devido aos arredondamentos, então atribuiremos um valor escolhido para o x_n , pois o algoritmo de retrosubstituição inicia com o cálculo do x_n . Além disso, a linha que se torna nula será a última, sempre que a pivotação for aplicada; e, se o método não tiver pivotação associada, algum pivô poderá se tornar nulo antes, ainda dentro do algoritmo de eliminação.

Já um **sistema impossível** de resolver se comporta conforme o exemplo a seguir:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 0 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 50 \end{cases}$$

($L3$ difere de $L2$ somente pelo termo independente)

Se aplicarmos a eliminação gaussiana, chegaremos à seguinte matriz expandida:

$$\left[\begin{array}{ccc|c} 0.448 & 0.832 & 0.193 & : & 1 \\ 0 & 0.002170 & -0.3884 & : & -0.9397 \\ 0 & 0 & 0 & : & 49.0603 \end{array} \right]$$

(usando aritmética exata em $i=3$)

Observe que a terceira linha, ou equação, foi parcialmente eliminada no passo $k=2$, mas ficou com um valor não nulo no termo independente:

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0x_1 + 0.002170x_2 - 0.3884x_3 = -0.9397 \\ 0x_1 + 0x_2 + 0x_3 = 49.0603 \end{cases}$$

Lembre-se que o algoritmo da função retrosubstituição também deve levar em conta essa possibilidade de ter o coeficiente da diagonal principal da última linha numericamente nulo, enquanto o seu termo independente é não nulo. Portanto, a última equação desse sistema é **impossível** de ser satisfeita, pois não possui solução, ou seja, uma



O símbolo IEEE *NaN* é um resultado para operações indefinidas como $(\text{Inf} - \text{Inf})$, $(0/0)$ e para quaisquer operações envolvendo *NaN*.



Nos algoritmos exemplos deste livro, usamos principalmente o software livre GNU Octave, compatível com MathLab, disponível em: <<https://www.gnu.org/software/octave/>>. Dessa forma, com facilidade, podemos copiar os algoritmos e testar exemplos e exercícios.

das equações do sistema é “inconsistente”. Nesse caso, o algoritmo não tem como tratar um sistema inconsistente. Como alternativa, podemos apenas interromper o algoritmo, ou atribuir um valor ***NaN*** (*Not a Number*) para o vetor solução, indicando solução como conjunto vazio.

Na sequência, vamos apresentar o algoritmo de eliminação gaussiana com pivotação parcial ou total aplicado ao **Exemplo 2.3** e observações importantes sobre o método de eliminação.

Neste Capítulo e ao longo desta obra, vamos apresentar todos os **algoritmos** (compactados) no **Caderno de Algoritmos**, que disponibilizamos no *link* <<http://sergiopeters.prof.ufsc.br/algoritmos-livro/>>, indicando o título do algoritmo correspondente ao método ou exemplo apresentado. Aproveite este momento para fazer o download deste Caderno e conferir o primeiro algoritmo, disponibilizado no arquivo **Cap2ElimGauss.m**.

No algoritmo de eliminação de Gauss, o processo de pivotação, total ou parcial, está inserido antes da etapa da eliminação.

No processo de escalonamento, as operações nas colunas envolvem algebricamente todos os j , $\text{for } j=1:n+1$, pois são operações sobre as linhas i inteiras; mas a operação na coluna $j=k$ tem resultado conhecido, uma vez que sabemos que esse resultado exato é sempre nulo. Assim, essa operação aritmética conhecida foi suprimida no algoritmo, $\text{for } j=k+1:n+1$, e atribuímos o resultado nulo conhecido para a coluna $j=k$ depois do encerramento do *for j*. Além disso, as operações nas colunas j à esquerda de k ($j=1:k$) envolveriam somente operações com coeficientes nulos, que também são desnecessárias e por isso também são suprimidas.

Sobre os **Exemplos 2.1, 2.2 e 2.3** que desenvolvemos, algumas considerações são importantes e merecem destaque:

- Os resíduos encontrados servem de parâmetros para validar uma solução, ou seja, servem para verificar o número

de dígitos exatos da solução, mas não para sistemas mal condicionados, nos quais não basta calcular os resíduos, como veremos mais adiante.

- b) Os erros calculados nos exemplos capturam somente os erros de arredondamento da solução, pois não temos erros de truncamento envolvidos no método de eliminação gaussiana.
- c) As soluções com menores erros de arredondamento acumulados foram obtidas com pivotação parcial.
- d) Para minimizar o efeito cumulativo dos erros de arredondamento, podemos tentar modificar as operações elementares do processo de escalonamento da seguinte forma:
 - i) Substituímos a linha a ser eliminada pelo produto entre a própria linha e o elemento pivô subtraído do produto entre a linha do pivô e o elemento a ser eliminado. Por exemplo, se a linha L_3 sofre a seguinte operação de eliminação do elemento a_{31} com o pivô da primeira linha a_{11} no passo $k=1$, temos:

$$L_3 \leftarrow L_3 - \frac{a_{31}}{a_{11}} L_1$$

Sabendo que o objetivo dessa operação é anular o elemento a_{31} , podemos modificá-la desde que mantenhamos o resultado nulo na primeira coluna. Assim, se

$$L_3 - \frac{a_{31}}{a_{11}} L_1 = 0, \text{ para } j = 1, \quad a_{31} - \frac{a_{31}}{a_{11}} a_{11} = 0$$

Então, podemos multiplicar essa equação pelo elemento pivô a_{11} , resultando em uma forma alternativa equivalente que mantém o resultado nulo para primeira coluna da matriz. Mas esse processo sem divisões exige uma multiplicação a mais em cada linha, aumentando o número total de operações aritméticas:

$$a_{11} * L_3 - a_{31} * L_1 = 0 \text{ para } j=1, a_{11} * a_{31} - a_{31} * a_{11} = 0$$

Com essa forma alternativa de aplicar as operações de eliminação, temos um menor acúmulo de erros de arredondamento, pois não haverá divisões ao longo do processo de eliminação, sendo necessária apenas uma divisão no final do processo de retrosubstituição, no momento de determinar as incógnitas x_i . Mesmo assim, esse procedimento alternativo pode gerar erros por perda de significação, especialmente quando temos pivôs de grande magnitude ou quando o número de equações é elevado. Nesses casos, ocorre um acúmulo de operações de multiplicação sobre as linhas do sistema, amplificando seus valores e podendo atingir a região de *overflow*, sobretudo nas últimas linhas da matriz escalonada.

2.1.2 Método de Gauss-Jordan

O método de Gauss-Jordan consiste em transformar a matriz dos coeficientes, que compõe a matriz expandida, em identidade. Então, depois de transformada, a última coluna dessa matriz expandida será a solução de $A * X = B$. Essa transformação é obtida através da aplicação sucessiva de operações elementares sobre linhas buscando a eliminação de todos os coeficientes da respectiva coluna, exceto o da diagonal principal. Também podemos associar esse método a um processo de pivotamento parcial ou total.

Não detalharemos o algoritmo desse método aqui porque ele necessita do dobro de operações artiméticas em relação ao método de eliminação gaussiana e, por conseguinte, acumula mais arredondamentos.

2.1.3 Método da Inversão de Matrizes

No método de inversão de matrizes, multiplicamos o sistema $A * X = B$ pela matriz inversa A^{-1} de A :

$$A^{-1}(A * X) = A^{-1}(B)$$

Utilizando a associatividade do produto matricial, o sistema resulta em:

$$\begin{aligned} (A^{-1} * A)X &= A^{-1} * B \\ (I)X &= A^{-1} * B \\ X &= A^{-1} * B \end{aligned}$$

Portanto, podemos obter o vetor de incógnitas X apenas multiplicando a matriz inversa A^{-1} pelo vetor de termos independentes B . Trata-se de um método eficiente quando dispomos da inversa da matriz A ; caso contrário, teremos o custo adicional da determinação da inversa.

Podemos obter a matriz inversa de A de várias maneiras. Aqui, vamos usar o próprio método de escalonamento de Gauss-Jordan aplicado à matriz aumentada $[A : I]$, conforme o **Exemplo 2.4**.

Exemplo 2.4: resolva o sistema de equações lineares, a seguir, usando o método da inversão de matrizes. Utilize o processo de pivotamento parcial para evitar pivôs nulos e diminuir o acúmulo de arredondamentos.

$$\begin{cases} 3x_1 + 1.5x_2 + 4.75x_3 = 8 \\ 4x_1 + 2x_2 + 3x_3 = 7 \\ 2x_1 + 5x_2 + 3x_3 = -12 \end{cases}$$

Solução:

Agora, adotamos operações aritméticas com 4 dígitos significativos e arredondamento ponderado. Assim, na forma matricial, temos:

$$\begin{bmatrix} 3 & 1.5 & 4.75 \\ 4 & 2 & 3 \\ 2 & 5 & 3 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 7 \\ -12 \end{bmatrix}$$

Para a geração da matriz inversa de A , adotamos um método prático clássico: geramos a matriz aumentada $[A : I]$ composta da matriz A concatenada com a matriz identidade I da mesma ordem de A :

$$\begin{bmatrix} 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ 4 & 2 & 3 & : & 0 & 1 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{bmatrix}$$

Através de operações elementares sobre linhas, transformamos a matriz A na matriz identidade I , e consequentemente a matriz identidade inicial transforma-se na inversa A^{-1} . Trata-se de um procedimento análogo ao que utilizamos no método de Gauss-Jordan, mas, nesse caso, estendido para $2n$ colunas.

A seguir, vamos apresentar os passos para a obtenção de A^{-1} .

Primeiro passo: utilizando o método de pivotação parcial, correspondente ao primeiro pivô ($k=1$):

a) Buscamos o maior módulo da coluna $j=1$:

$$j=1$$

$$i=2 \begin{bmatrix} 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ (4) & 2 & 3 & : & 0 & 1 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{bmatrix}$$

(elemento de maior módulo da coluna $j=1$ está na segunda linha $i=2$)

b) Trocamos linhas:

$$\begin{bmatrix} 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ (4) & 2 & 3 & : & 0 & 1 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{bmatrix} L_1 \leftarrow L_2$$

$$\begin{bmatrix} (4) & 2 & 3 & : & 0 & 1 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \\ 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \end{bmatrix} L_2 \leftarrow L_1$$

(troca da linha L_1 com L_2 e vice-versa)

c) Geramos a matriz pivotada:

$$\left[\begin{array}{ccc|ccc} 4 & 2 & 3 & : & 0 & 1 & 0 \\ 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{array} \right]$$

Segundo passo: no processo de normalização do primeiro pivô ($k=1$), temos:

$$\left[\begin{array}{ccc|ccc} 4 & 2 & 3 & : & 0 & 1 & 0 \\ 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{array} \right] L_1 \leftarrow L_1 / 4$$

$$\left[\begin{array}{ccc|ccc} 1 & 0.5 & 0.75 & : & 0 & 0.25 & 0 \\ 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{array} \right]$$

Terceiro passo: no processo de diagonalização, correspondente ao primeiro passo ($k=1$), temos:

$$\left[\begin{array}{ccc|ccc} 1 & 0.5 & 0.75 & : & 0 & 0.25 & 0 \\ 3 & 1.5 & 4.75 & : & 1 & 0 & 0 \\ 2 & 5 & 3 & : & 0 & 0 & 1 \end{array} \right] L_2 \leftarrow L_2 - 3 L_1$$

$$\left[\begin{array}{ccc|ccc} 1 & 0.5 & 0.75 & : & 0 & 0.25 & 0 \\ 0 & 0 & 2.5 & : & 1 & -0.75 & 0 \\ 2 & 5 & 3 & : & 0 & -0.5 & 1 \end{array} \right]$$

Quarto passo: na pivotação parcial, correspondente ao segundo pivô ($k=2$):

a) Buscamos o maior módulo da coluna $k=2$ (a partir da segunda linha, pois a primeira linha já foi utilizada no processo de eliminação):

$$j = 2$$

$$i = 3 \quad \left[\begin{array}{ccc|ccc} 1 & 0.5 & 0.75 & : & 0 & 0.25 & 0 \\ 0 & 0 & 2.5 & : & 1 & -0.75 & 0 \\ 0 & (4) & 1.5 & : & 0 & -0.5 & 1 \end{array} \right]$$

(o elemento de maior módulo da coluna $j=2$ está na linha $i=3$).

b) Trocamos linhas:

$$\left[\begin{array}{cccc|cc} 1 & 0.5 & 0.75 & 0 & 0.25 & 0 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \\ 0 & (4) & 1.5 & 0 & -0.5 & 1 \end{array} \right] \begin{array}{l} L_2 \leftarrow L_3 \\ L_3 \leftarrow L_2 \end{array} \quad (\text{troca da linha } L_2 \text{ com } L_3 \text{ e vice-versa})$$

c) Geramos a matriz pivotada:

$$\left[\begin{array}{cccc|cc} 1 & 0.5 & 0.75 & 0 & 0.25 & 0 \\ 0 & 4 & 1.5 & 0 & -0.5 & 1 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \end{array} \right]$$

Note que a pivotação parcial eliminou um pivô nulo.

Quinto passo: no processo de normalização do segundo pivô ($k=2$), temos:

$$\left[\begin{array}{cccc|cc} 1 & 0.5 & 0.75 & 0 & 0.25 & 0 \\ 0 & 4 & 1.5 & 0 & -0.5 & 1 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \end{array} \right] L_2 \leftarrow L_2 / 4$$

$$\left[\begin{array}{cccc|cc} 1 & 0.5 & 0.75 & 0 & 0.25 & 0 \\ 0 & 1 & 0.375 & 0 & -0.125 & 0.25 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \end{array} \right]$$

Sexto passo: no processo de diagonalização, correspondente ao segundo passo ($k=2$), temos:

$$\left[\begin{array}{cccc|cc} 1 & 0.5 & 0.75 & 0 & 0.25 & 0 \\ 0 & 1 & 0.375 & 0 & -0.125 & 0.25 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \end{array} \right] \begin{array}{l} L_1 \leftarrow L_1 - 0.5L_2 \\ L_3 \leftarrow L_3 - 0L_2 \end{array}$$

$$\left[\begin{array}{cccc|cc} 1 & 0 & 0.5625 & 0 & 0.3125 & -0.125 \\ 0 & 1 & 0.375 & 0 & -0.125 & 0.25 \\ 0 & 0 & 2.5 & 1 & -0.75 & 0 \end{array} \right]$$

Sétimo passo: no processo de normalização do terceiro pivô ($k=3$), temos:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0.5625 & : & 0 & 0.3125 & -0.125 \\ 0 & 1 & 0.375 & : & 0 & -0.125 & 0.25 \\ 0 & 0 & 2.5 & : & 1 & -0.75 & 0 \end{array} \right] L_3 \leftarrow L_3 / 2.5$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0.5625 & : & 0 & 0.3125 & -0.125 \\ 0 & 1 & 0.375 & : & 0 & -0.125 & 0.25 \\ 0 & 0 & 1 & : & 0.4 & -0.3 & 0 \end{array} \right]$$

Oitavo passo: no processo de diagonalização, correspondente ao terceiro passo ($k=3$), temos:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0.5625 & : & 0 & 0.3125 & -0.125 \\ 0 & 1 & 0.375 & : & 0 & -0.125 & 0.25 \\ 0 & 0 & 1 & : & 0.4 & -0.3 & 0 \end{array} \right] L_1 \leftarrow L_1 - 0.5625L_3, \quad L_2 \leftarrow L_2 - 0.375L_3$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & : & -0.225 & 0.4812 & -0.125 \\ 0 & 1 & 0 & : & -0.15 & -0.0125 & 0.25 \\ 0 & 0 & 1 & : & 0.4 & -0.3 & 0 \end{array} \right]$$

Então,

$$A^{-1} = \begin{bmatrix} -0.225 & 0.4812 & -0.125 \\ -0.15 & -0.0125 & 0.25 \\ 0.4 & -0.3 & 0 \end{bmatrix}$$

Para obter o vetor solução x , efetuamos o produto entre A^{-1} e B , logo:

$$X = A^{-1} * B = \begin{bmatrix} -0.225 & 0.4812 & -0.125 \\ -0.15 & -0.0125 & 0.25 \\ 0.4 & -0.3 & 0 \end{bmatrix} * \begin{bmatrix} 8 \\ 7 \\ -12 \end{bmatrix} = \begin{bmatrix} 3.069 \\ -4.288 \\ 1.100 \end{bmatrix}$$

Observe que a troca de linhas efetuadas na pivotação parcial não afeta a matriz inversa e não deve alterar a ordem dos elementos do termo independente. Então, a solução do sistema é a seguinte:

$$S = \{ 3.069, -4.288, 1.100 \}$$

A **vantagem** desse método de inversão de matriz sobre a eliminação gaussiana é a possibilidade de **reuso da matriz inversa** A^{-1} para outros sistemas com a mesma matriz A de coeficientes e com diferentes vetores B de termos independentes.

O algoritmo do método de inversão de matrizes necessita de quase o triplo de operações aritméticas que o método de eliminação gaussiana, por conseguinte acumula mais arredondamentos. Por isso, só vale o esforço computacional se a matriz inversa resultante puder ser reutilizada muitas vezes.

A seguir, vamos apresentar métodos que fazem uso de decomposição de matrizes em vez de eliminação.

2.1.4 Método de Decomposição LU (de Crout)

Matrizes quadradas não singulares podem ser decompostas no produto de duas matrizes triangulares L e U , em que L é uma matriz triangular inferior e U é uma matriz triangular superior, de modo que $A = L * U$.

Além disso, se atribuirmos valores fixos aos elementos da diagonal principal, seja na matriz L ($l_{ii} = 1$, método de Doolittle) ou na U ($u_{ii} = 1$, método de Crout), essa decomposição será única.

Para a solução de $A * X = B$, podemos decompor A , segundo o método de Crout, em:

$$L = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ \vdots & \vdots & \dots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \text{ e } U = \begin{bmatrix} 1 & u_{12} & u_{1n} \\ 0 & 1 & u_{2n} \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad (4)$$

tal que $L * U = A$. Então, o sistema $A * X = B$ torna-se $(L * U) * X = B$.

Também podemos associar o produto das matrizes L e U , $(L * U) * X = B$, como $L * (U * X) = B$.

Considerando $U * X = C$ (vetor auxiliar C desconhecido), resolvemos primeiro $L * C = B$ determinando C e, depois, $U * X = C$ determinando X .

A vantagem do método de decomposição LU sobre o método de eliminação gaussiana é a possibilidade de reuso das matrizes L e U decompostas para outros sistemas que usem a mesma matriz A de coeficientes e com diferentes vetores B de termos independentes.

Vamos exemplificar a determinação dos elementos de L e U para uma matriz 3 por 3 na qual a multiplicação das matrizes $L * U$ pode ser usada para definir os valores de l_{ij} e u_{ij} em função de a_{ij} , pois o resultado do produto $L * U$ é a matriz A original, logo:

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} * \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Então, nesse exemplo, o produto $L * U$ gera 9 equações lineares com 9 incógnitas, uma para cada elemento da matriz:

$$\begin{bmatrix} l_{11} * 1 + 0 * 0 + 0 * 0 & l_{11} * u_{12} + 0 * 1 + 0 * 0 & l_{11} * u_{13} + 0 * u_{23} + 0 * 1 \\ l_{21} * 1 + l_{22} * 0 + 0 * 0 & l_{21} * u_{12} + l_{22} * 1 + 0 * 0 & l_{21} * u_{13} + l_{22} * u_{23} + 0 * 1 \\ l_{31} * 1 + l_{32} * 0 + l_{33} * 0 & l_{31} * u_{12} + l_{32} * 1 + l_{33} * 0 & l_{31} * u_{13} + l_{32} * u_{23} + l_{33} * 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Podemos obter de forma direta a solução desse sistema de equações usando os seguintes passos:

Passo $k=1$: resolvemos as equações correspondentes à primeira coluna $j=1$ da matriz $L * U = A$:

$$\begin{aligned} l_{11} * 1 + 0 * 0 + 0 * 0 &= a_{11} & l_{11} &= a_{11} \\ l_{21} * 1 + l_{22} * 0 + 0 * 0 &= a_{21} \Rightarrow l_{21} &= a_{21} \\ l_{31} * 1 + l_{32} * 0 + l_{33} * 0 &= a_{31} & l_{31} &= a_{31} \end{aligned}$$

Para a primeira linha $i=1$ (à direita da diagonal principal), temos:

$$\begin{aligned} l_{11} * u_{12} + 0 * 1 + 0 * 0 &= a_{12} \Rightarrow u_{12} = a_{12} / l_{11} \\ l_{11} * u_{13} + 0 * u_{23} + 0 * 1 &= a_{13} \Rightarrow u_{13} = a_{13} / l_{11} \end{aligned}$$

Passo k=2: resolvemos as equações correspondentes à segunda coluna $j=2$ (a partir da diagonal principal):

$$\begin{aligned} l_{21} * u_{12} + l_{22} * 1 + 0 * 0 &= a_{22} \Rightarrow l_{22} = a_{22} - l_{21} * u_{12} \\ l_{31} * u_{12} + l_{32} * 1 + l_{33} * 0 &= a_{32} \Rightarrow l_{32} = a_{32} - l_{31} * u_{12} \end{aligned}$$

Para a segunda linha $i=2$ (à direita da diagonal principal), temos:

$$l_{21} * u_{13} + l_{22} * u_{23} + 0 * 1 = a_{23} \Rightarrow u_{23} = (a_{23} - l_{21} * u_{13}) / l_{22}$$

Passo k=3: resolvemos a equações correspondentes à terceira coluna $j=3$ (a partir da diagonal principal):

$$l_{31} * u_{13} + l_{32} * u_{23} + l_{33} * 1 = a_{33} \Rightarrow l_{33} = (a_{33} - (l_{31} * u_{13} + l_{32} * u_{23}))$$

E não temos linhas à direita da diagonal principal da última linha.

Resumindo:

$k=1$

$$\begin{aligned} l_{11} &= a_{11} \\ l_{21} &= a_{21} & u_{12} &= a_{12} / l_{11} \\ l_{31} &= a_{31} & u_{13} &= a_{13} / l_{11} \end{aligned}$$

$k=2$

$$\begin{aligned} l_{22} &= a_{22} - l_{21} * u_{12} \\ l_{32} &= a_{32} - l_{31} * u_{13} & u_{23} &= (a_{23} - l_{21} * u_{13}) / l_{22} \end{aligned}$$

$k=3$

$$l_{33} = a_{33} - l_{31} * u_{13} - l_{32} * u_{23}$$

Então, o algoritmo de decomposição segue esta sequência de cálculos:

- determinamos os elementos da primeira coluna de L e da primeira linha de U ; e
- determinamos os elementos da segunda coluna de L e da segunda linha de U e assim sucessivamente.

Obtidos L e U , devemos resolver $L^* C = B$ determinando C por substituições sucessiva à frente:

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} * \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{cases} l_{11} * c_1 = b_1 \\ l_{21} * c_1 + l_{22} * c_2 = b_2 \\ l_{31} * c_1 + l_{32} * c_2 + l_{33} * c_3 = b_3 \end{cases} \Rightarrow \begin{cases} c_1 = (b_1) / l_{11} \\ c_2 = (b_2 - (l_{21} * c_1)) / l_{22} \\ c_3 = (b_3 - (l_{31} * c_1 + l_{32} * c_2)) / l_{33} \end{cases}$$

Note que o cálculo de C , do sistema $L^* C = B$, segue a mesmo formato do cálculo dos elementos de U . Uma vez obtidos L , U e C , devemos resolver $U^* X = C$ determinando X por retrosubstituições sucessivas:

$$\begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\begin{cases} 1 * x_1 + u_{12} * x_2 + u_{13} * x_3 = c_1 \\ 0 * x_1 + 1 * x_2 + u_{23} * x_3 = c_2 \\ 0 * x_1 + 0 * x_2 + 1 * x_3 = c_3 \end{cases} \Rightarrow \begin{cases} x_1 = c_1 - (u_{12} * x_2 + u_{13} * x_3) \\ x_2 = c_2 - (u_{23} * x_3) \\ x_3 = c_3 \end{cases}$$

Sugerimos usar o armazenamento das colunas L e U sobrepostas na mesma área de memória de A , o que otimiza o armazenamento e

as operações de pivotação, uma vez que a diagonal de U é unitária e não precisa ser armazenada:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} l_{11} & u_{12} & u_{13} \\ l_{21} & l_{22} & u_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

De uma forma geral, para sistemas de ordem n , teremos o seguinte formulário:

Primeiro passo: $k=1$

$$\begin{aligned} l_{i1} &= a_{i1} & \forall i = 1, 2, 3, \dots, n \\ u_{1j} &= a_{1j} / l_{11} & \forall j = 2, 3, \dots, n \end{aligned}$$

Segundo passo: $k=2,3,\dots,n-1$

$$\begin{aligned} l_{ij} &= a_{ij} - \sum_{r=1}^{j-1} l_{ir} * u_{rj} & \forall i = k, k+1, \dots, n \text{ e } j = k \\ u_{ij} &= \frac{1}{l_{ii}} \left(a_{ij} - \sum_{r=1}^{i-1} l_{ir} * u_{rj} \right) & \forall j = k+1, \dots, n \text{ e } i = k \end{aligned}$$

Terceiro e último passo: $k=n$

$$l_{ij} = a_{ij} - \sum_{r=1}^{j-1} l_{ir} * u_{rj} \quad i=j=n$$

O método de Crout com pivotação gera erros menores do que sem pivotação. O efeito da pivotação sobre o método de Crout é análogo ao seu efeito sobre o método de Gauss apresentado na seção anterior, ou seja, reduz o acúmulo de erros de arredondamento, conforme o **Exemplo 2.5**.

Exemplo 2.5: resolva o sistema, a seguir, pelo método de Crout **com pivotamento parcial** (é imprescindível incluir o(s) vetor(es) B do sistema na troca de linhas da pivotação para não alterar

as equações), usando 4 dígitos significativos e arredondamento ponderado.

$$\begin{cases} 0.448x_1 + 0.832x_2 + 0.193x_3 = 1 \\ 0.421x_1 + 0.784x_2 - 0.207x_3 = 2 \\ -0.319x_1 + 0.884x_2 + 0.279x_3 = 0 \end{cases}$$

Observações:

- a) o(s) vetor(es) de termos independentes B será(ão) colocado(s) à direita da matriz para que seja(m) pivotado(s) juntamente com as linhas de A ; e
- b) no algoritmo, os valores da matriz decomposta L e U serão armazenados na própria área de memória de A .

Na sequência do **Exemplo 2.5**, os valores da matriz decomposta L e U também são escritos na mesma matriz A , sendo diferenciados da matriz A original da seguinte forma:

- a) representação em negrito: valores novos já decompostos em L e U ; e
- b) representação normal: valores originais da matriz A .

$$\left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array} \right] \text{ depois da decomposição } LU \rightarrow \left[\begin{array}{ccc} \mathbf{l}_{11} & \mathbf{u}_{12} & \mathbf{u}_{13} \\ \mathbf{l}_{21} & \mathbf{l}_{22} & \mathbf{u}_{23} \\ \mathbf{l}_{31} & \mathbf{l}_{32} & \mathbf{l}_{33} \end{array} \right]$$

Solução:

Primeiro passo: na decomposição LU (método de Crout), correspondente ao primeiro passo ($k=1$):

- a) Definimos a primeira coluna $j=1$ da matriz L , l_{i1} para $i=1,2,3$:
 $\rightarrow l_{i1} = a_{i1} \quad i=1, 2, 3, \dots, n$ (L equivale à coluna original de A)

$$\left[\begin{array}{ccc} 0.448 & 0.832 & 0.193 \\ 0.421 & 0.784 & -0.207 \\ -0.319 & 0.884 & 0.279 \end{array} \right] \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$$

C_1



C_1

resultando em

$$\begin{bmatrix} \mathbf{(0.448)} & 0.832 & 0.193 \\ \mathbf{0.421} & 0.784 & -0.207 \\ \mathbf{-0.319} & 0.884 & 0.279 \end{bmatrix} \begin{array}{l} 1 \\ 2 \\ 3 \end{array}$$

b) Observamos que a pivotação parcial deve ocorrer depois do cálculo dos valores de l_{ij} na coluna $j=k$ e antes do cálculo dos valores u_{ij} da linha $i=k$, para que cada elemento l_{kk} , que acaba de ser calculado na diagonal principal, seja não nulo e também tenha o maior módulo possível. Para o primeiro passo ($k=1$), a pivotação poderia ser feita ainda antes do cálculo de l_{11} , pois $l_{11}=a_{11}$. A matriz l_{ij} , parcialmente calculada (em negrito), já se encontra pivotada neste exemplo:

$$\begin{bmatrix} \mathbf{(0.448)} & 0.832 & 0.193 \\ \mathbf{0.421} & 0.784 & -0.207 \\ \mathbf{-0.319} & 0.884 & 0.279 \end{bmatrix} \begin{array}{l} 1 \\ 2 \\ 3 \end{array}$$

c) Definimos a primeira linha $i=1$ da matriz U , u_{1j} para $j=2,3$:

$$\rightarrow u_{1j} = a_{1j} / l_{11} \quad j=2,3,\dots,n$$

$$\begin{bmatrix} \mathbf{(0.448)} & 0.832 & 0.193 \\ \mathbf{0.421} & 0.784 & -0.207 \\ \mathbf{-0.319} & 0.884 & 0.279 \end{bmatrix} \begin{array}{l} 1 \\ 2 \\ 3 \end{array} \quad L_1 \leftarrow L_1 / l_{11}$$

resultando em

$$\begin{bmatrix} \mathbf{(0.448)} & \mathbf{1.857} & \mathbf{0.4308} \\ \mathbf{0.421} & 0.784 & -0.207 \\ \mathbf{-0.319} & 0.884 & 0.279 \end{bmatrix} \begin{array}{l} 1 \\ 2 \\ 3 \end{array}$$

Segundo passo: na decomposição LU , correspondente ao segundo passo ($k=2$):

a) Definimos a coluna $j=2$ da matriz L , l_{i2} para $i=2, 3$:

$$\rightarrow l_{ij} = a_{ij} - \sum_{r=1}^{j-1} l_{ir} * u_{rj} \quad (i = k, k+1, \dots, n \text{ e } j = k)$$

$$\left[\begin{array}{ccc|c} 0.448 & 1.857 & 0.4308 & 1 \\ 0.421 & 0.784 & -0.207 & 2 \\ -0.319 & 0.884 & 0.279 & 3 \end{array} \right]$$

$$\begin{matrix} C_2 \\ \uparrow \end{matrix}$$

$$C_2 - l_{i1} * u_{12}$$

resultando em

$$\left[\begin{array}{ccc|c} 0.448 & 1.857 & 0.4308 & 1 \\ 0.421 & 0.002203 & -0.207 & 2 \\ -0.319 & 1.476 & 0.279 & 3 \end{array} \right]$$

Observe que a pivotação parcial, correspondente ao segundo pivô ($k=2$), deve ocorrer depois do cálculo dos valores de l_{ij} na coluna $j=k$ e antes do cálculo dos valores u_{ij} da linha $i=k$, conforme mencionado no item (b) do primeiro passo.

- b) Procedemos com a busca parcial do maior módulo da coluna $j=2$ (busca a partir da segunda linha):

$$\left[\begin{array}{ccc|c} 0.448 & 1.857 & 0.4308 & 1 \\ 0.421 & 0.002203 & -0.207 & 2 \\ -0.319 & (1.476) & 0.279 & 3 \end{array} \right]$$

Maior coeficiente em módulo encontrado na linha $i=3$.

- c) Trocamos linhas:

$$\left[\begin{array}{ccc|c} 0.448 & 1.857 & 0.4308 & 1 \\ 0.421 & 0.002203 & -0.207 & 2 \\ -0.319 & (1.476) & 0.279 & 3 \end{array} \right] \quad \begin{matrix} & & & 1 \\ & & L_2 \leftarrow L_3 & 2 \\ & & L_3 \leftarrow L_2 & 3 \end{matrix}$$

resultando em

$$\left[\begin{array}{ccc|c} 0.448 & 1.857 & 0.4308 & 1 \\ -0.319 & (1.476) & 0.279 & 3 \\ 0.421 & 0.002203 & -0.207 & 2 \end{array} \right]$$

d) Definimos a linha $i=2$ da matriz U , u_{2j} para $j=3$:

$$\rightarrow u_{ij} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{r=1}^{i-1} l_{ir} * u_{rj} \right) \quad (j = k+1, \dots, n \text{ e } i = k)$$

$$\begin{bmatrix} \mathbf{0.448} & \mathbf{1.857} & \mathbf{0.4308} \\ \mathbf{-0.319} & \mathbf{(1.476)} & 0.279 \\ \mathbf{0.421} & \mathbf{0.002203} & -0.207 \end{bmatrix} \begin{matrix} 1 \\ 3 \\ 2 \end{matrix} \quad L_2 \leftarrow (L_2 - l_{21} * u_{1j}) / l_{22}$$

resultando em

$$\begin{bmatrix} \mathbf{0.448} & \mathbf{1.857} & \mathbf{0.4308} \\ \mathbf{-0.319} & \mathbf{(1.476)} & \mathbf{0.2821} \\ \mathbf{0.421} & \mathbf{0.002203} & -0.207 \end{bmatrix} \begin{matrix} 1 \\ 3 \\ 2 \end{matrix}$$

Terceiro passo: na decomposição LU correspondente ao terceiro passo ($k=3$):

a) Definimos a coluna $j=3$ da matriz L , l_{i3} para $i=3$:

$$\rightarrow l_{ij} = a_{ij} - \sum_{r=1}^{j-1} l_{ir} * u_{rj} \quad (i = j=n)$$

$$\begin{bmatrix} \mathbf{0.448} & \mathbf{1.857} & \mathbf{0.4308} \\ \mathbf{-0.319} & \mathbf{(1.476)} & \mathbf{0.2821} \\ \mathbf{0.421} & \mathbf{0.002203} & -0.207 \end{bmatrix} \begin{matrix} 1 \\ 3 \\ 2 \end{matrix}$$

C_3
↑

$$C_3 - l_{31} * u_{13} - l_{32} * u_{23}$$

resultando em

$$\begin{bmatrix} \mathbf{0.448} & \mathbf{1.857} & \mathbf{0.4308} \\ \mathbf{-0.319} & \mathbf{(1.476)} & \mathbf{0.2821} \\ \mathbf{0.421} & \mathbf{0.002203} & \mathbf{-0.3890} \end{bmatrix} \begin{matrix} 1 \\ 3 \\ 2 \end{matrix}$$

Logo,

$$L = \begin{bmatrix} 0.448 & 0 & 0 \\ -0.319 & 1.476 & 0 \\ 0.421 & 0.002203 & -0.3890 \end{bmatrix} \text{ e } \begin{bmatrix} 1 & 1.857 & 0.4308 \\ 0 & 1 & 0.2821 \\ 0 & 0 & 1 \end{bmatrix}$$

Decomposta a matriz A , vamos agora obter a solução do sistema através da solução de dois sistemas triangulares:

$$L * C = B \Rightarrow \begin{cases} 0.448c_1 + 0c_2 + 0c_3 = 1 \\ -0.319c_1 + 1.476c_2 + 0c_3 = 0 \\ 0.421c_1 + 0.002203c_2 - 0.3890c_3 = 2 \end{cases}$$

Observe que o vetor B do sistema também trocou de linhas em razão da pivotação:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \quad L_2 \leftarrow L_3 \Rightarrow \begin{bmatrix} b_1 \\ b_3 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

Aplicando substituições sucessivas à frente, a partir de c_1 , temos:

$$\begin{aligned} c_1 &= (1)/0.448 & c_1 &= 2.232 \\ c_2 &= (0 - (-0.319 c_1))/1.476 & \Rightarrow & c_2 = 0.4824 \\ c_3 &= (2 - (0.421 c_1 + 0.002203 c_2))/(-0.3890) & & c_3 = -2.723 \end{aligned}$$

Resolvendo

$$U * X = C \Rightarrow \begin{cases} 1x_1 + 1.857x_2 + 0.4308x_3 = 2.232 \\ 0x_1 + 1x_2 + 0.2821x_3 = 0.4824 \\ 0x_1 + 0x_2 + 1x_3 = -2.723 \end{cases}$$

E aplicando o processo de retrosubstituições sucessivas a partir de x_3 :

$$\begin{aligned} x_3 &= -2.723 & x_3 &= -2.723 \\ x_2 &= (0.4824 - 0.2821 x_3) & \Rightarrow & x_2 = 1.251 \\ x_1 &= (2.232 - (1.857 x_2 + 0.4308 x_3)) & & x_1 = 1.082 \end{aligned}$$

Então, a solução obtida é:

$$S = \{ 1.082, 1.251, -2.723 \}$$

Para determinar o erro da solução anterior, obtemos a solução exata calculada com 16 dígitos significativos e pivotação parcial, resultando em:

$$S_{\text{exato}} = \{ 1.08321566094182, 1.25034738133745, -2.72315874287407 \}$$

Daí o erro de arredondamento da solução S será:

$$\text{Erro} = |S - S_{\text{exato}}|:$$

$$\text{Erro } x_1 = |1.082 - 1.08321566094182| \cong 1.22e-03$$

$$\text{Erro } x_2 = |1.251 - 1.25034738133745| \cong 6.53e-04$$

$$\text{Erro } x_3 = |-2.723 - (-2.72315874287407)| \cong 1.59e-04$$

Podemos ver que os **erros** da solução S obtida com **pivotação parcial** são compatíveis com a precisão de 4 dígitos significativos adotados, embora na primeira equação ainda ocorra uma propagação de erros para o terceiro dígito fracionário.

Exemplo 2.6: resolva os três sistemas, a seguir, pelo método de Crout com **pivotamento parcial** (é imprescindível incluir o(s) vetor(es) B do sistema na troca de linhas da pivotação) e com 4 dígitos significativos:

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 1 \\ x_1 + 2x_2 - 3x_3 - 4x_4 = 0 \\ -x_1 + 0x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - 4x_2 - x_3 + x_4 = 2 \end{cases} \quad \begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 1 \\ x_1 + 2x_2 - 3x_3 - 4x_4 = 0 \\ -x_1 + 0x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - 4x_2 - x_3 + x_4 = -2 \end{cases} \quad \begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = -1 \\ x_1 + 2x_2 - 3x_3 - 4x_4 = 0 \\ -x_1 + 0x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - 4x_2 - x_3 + x_4 = 2 \end{cases}$$

Observe que esse exemplo demonstra a real aplicação do método de decomposição LU , que ocorre quando vários sistemas têm a mesma matriz de coeficientes A , mas com diferentes vetores de termos independentes B . Portanto, com as mesmas matrizes L e U , decompostas de A , podemos resolver os três sistemas diferentes.

Agora, vamos gerar uma matriz com os coeficientes das incógnitas A concatenada com os três vetores B^1, B^2, B^3 de termos

independentes, para que, na pivotação parcial, a troca de linhas corresponda a trocas de equações:

$$\begin{array}{c} A : B^1, B^2, B^3 \\ \left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 2 & -3 & -4 \\ -1 & 0 & 2 & 3 \\ 1 & -4 & -1 & 1 \end{array} \right] : \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 2 \end{array} \right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ -2 \end{array} \right] \left[\begin{array}{c} -1 \\ 0 \\ 1 \\ 2 \end{array} \right] \end{array} \text{ ou } \begin{array}{c} A : B \\ \left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 2 & -3 & -4 \\ -1 & 0 & 2 & 3 \\ 1 & -4 & -1 & 1 \end{array} \right] : \left[\begin{array}{ccc} 1 & 1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & -2 & 2 \end{array} \right] \end{array}$$

Solução:

Primeiro passo: $k=1$

- a) Definimos a primeira coluna $j=1$ da matriz L , l_{i1} ($i=1,2,3,4$), que é igual à matriz A original:

$$\begin{array}{c} \left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 2 & -3 & -4 \\ -1 & 0 & 2 & 3 \\ 1 & -4 & -1 & 1 \end{array} \right] : \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 2 \end{array} \right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ -2 \end{array} \right] \left[\begin{array}{c} -1 \\ 0 \\ 1 \\ 2 \end{array} \right] \end{array}$$

- b) Lembraemos que a pivotação parcial, correspondente ao primeiro pivô ($k=1$), deve ocorrer depois do cálculo dos valores de l_{ik} . Nesse caso, a matriz já se encontra pivotada:

$$\begin{array}{c} \left[\begin{array}{cccc} (1) & 2 & 3 & 4 \\ 1 & 2 & -3 & -4 \\ -1 & 0 & 2 & 3 \\ 1 & -4 & -1 & 1 \end{array} \right] : \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 2 \end{array} \right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ -2 \end{array} \right] \left[\begin{array}{c} -1 \\ 0 \\ 1 \\ 2 \end{array} \right] \end{array}$$

- c) Definimos a primeira linha $i=1$ da matriz U , u_{1j} ($j=2,3,4$), que é igual à matriz A dividida pelo pivô $l_{11} = 1$:

$$\begin{array}{c} \left[\begin{array}{cccc} (1) & 2 & 3 & 4 \\ 1 & 2 & -3 & -4 \\ -1 & 0 & 2 & 3 \\ 1 & -4 & -1 & 1 \end{array} \right] : \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 2 \end{array} \right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ -2 \end{array} \right] \left[\begin{array}{c} -1 \\ 0 \\ 1 \\ 2 \end{array} \right] \end{array}$$

Segundo passo: $k=2$

- a) Definimos a segunda coluna $j=2$ da matriz L , l_{ij} ($i=2, 3, 4$), que é igual à matriz A menos um somatório de produtos cruzados da sua linha e coluna, conforme demonstramos ao lado das matrizes:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & -3 & -4 \\ -1 & 2 & 2 & 3 \\ 1 & (-6) & -1 & 1 \end{bmatrix} : \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ -2 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 0 \\ 2 \end{bmatrix}, \text{ em que } \begin{aligned} 0 &= 2 - (1 * 2) \\ 2 &= 0 - (-1 * 2) \\ -6 &= -4 - (1 * 2) \end{aligned}$$

- b) Observamos que a pivotação parcial, correspondente ao segundo pivô ($k=2$), deve ocorrer depois do cálculo dos valores de l_{ik} . Nesse caso, a linha 4 é trocada com a linha 2, em A e B :

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & (-6) & -1 & 1 \\ -1 & 2 & 2 & 3 \\ 1 & 0 & -3 & -4 \end{bmatrix} : \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

- c) Definimos a segunda linha $i=2$ da matriz U , u_{2j} ($j=3, 4$), que é igual à matriz A menos um somatório de produtos cruzados da sua linha e coluna, respectivamente, e dividido pelo pivô $l_{22} = -6$, conforme demonstramos ao lado das matrizes:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & (-6) & 0.6667 & 0.5 \\ -1 & 2 & 2 & 3 \\ 1 & 0 & -3 & -4 \end{bmatrix} : \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

em que

$$0.6667 = [-1 - (1 * 3)] / (-6)$$

$$0.5 = [1 - (1 * 4)] / (-6)$$

Terceiro passo: $k=3$

- a) Definimos a terceira coluna $j=3$ da matriz L , l_{ij} ($i=3, 4$):

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & -6 & 0.6667 & 0.5 \\ -1 & 2 & 3.6667 & 3 \\ 1 & 0 & (-6) & -4 \end{bmatrix} : \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

em que

$$\begin{aligned} 3.667 &= 2 - (-1 * 3 + 2 * 0.6667) \\ -6 &= -3 - (1 * 3 + 0 * 0.6667) \end{aligned}$$

b) Observamos que a pivotação parcial, correspondente ao terceiro pivô ($k=3$), ocorre depois do cálculo dos valores de l_{ik} . Nesse caso, a linha 4 é trocada com a linha 3, em A e B :

$$\left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & -6 & 0.6667 & 0.5 \\ 1 & 0 & (-6) & -4 \\ -1 & 2 & 3.667 & 3 \end{array} \right] : \left[\begin{array}{ccc|c} 1 & 1 & -1 \\ 2 & -2 & 2 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array} \right]$$

c) Definimos a terceira linha $i=3$ da matriz U , u_{3j} ($j=4$):

$$\left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & -6 & 0.6667 & 0.5 \\ 1 & 0 & (-6) & 1.333 \\ -1 & 2 & 3.667 & 3 \end{array} \right] : \left[\begin{array}{ccc|c} 1 & 1 & -1 \\ 2 & -2 & 2 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array} \right]$$

em que

$$1.333 = [-4 - (1 * 4 + 0 * 0.5)] / (-6)$$

Quarto passo: $k=4$

a) Definimos a quarta coluna $j=4$ da matriz L , l_{i4} ($i=4$):

$$\left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & (-6) & 0.6667 & 0.5 \\ 1 & 0 & (-6) & 1.333 \\ -1 & 2 & 3.667 & 1.112 \end{array} \right] : \left[\begin{array}{ccc|c} 1 & 1 & -1 \\ 2 & -2 & 2 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array} \right]$$

em que

$$1.112 = 3 - (-1 * 4 + 2 * 0.5 + 3.667 * 1.333)$$

Observe que a decomposição LU é feita uma única vez e agora podemos aplicá-la aos três sistemas, fazendo as substituições para cada B :

$$A * X = B^1$$

$$A * X = B^2$$

$$A * X = B^3$$

Primeiro sistema – $A * X = B^1$:

Resolvendo $L * C = B^1$ (observe que B^1 foi pivotado), temos

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -6 & 0 & 0 \\ 1 & 0 & -6 & 0 \\ -1 & 2 & 3.667 & 1.112 \end{bmatrix} * \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.1667 \\ 0.1667 \\ 1.549 \end{bmatrix}$$

Resolvendo $U * X = C^1$, temos

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0.6667 & 0.5 \\ 0 & 0 & 1 & 1.333 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.1667 \\ 0.1667 \\ 1.549 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -0.1516 \\ 0.3248 \\ -1.898 \\ 1.549 \end{bmatrix}$$

Segundo sistema – $A * X = B^2$:

Resolvendo $L * C = B^2$, temos

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -6 & 0 & 0 \\ 1 & 0 & -6 & 0 \\ -1 & 2 & 3.667 & 1.112 \end{bmatrix} * \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ 0.1667 \\ 0.3495 \end{bmatrix}$$

Resolvendo $U * X = C^2$, temos

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0.6667 & 0.5 \\ 0 & 0 & 1 & 1.333 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ 0.1667 \\ 0.3495 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -0.5498 \\ 0.5247 \\ -0.2992 \\ 0.3495 \end{bmatrix}$$

Terceiro sistema – $A * X = B^3$:

Resolvendo $L * C = B^3$, temos

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -6 & 0 & 0 \\ 1 & 0 & -6 & 0 \\ -1 & 2 & 3.667 & 1.112 \end{bmatrix} * \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} -1 \\ -0.5 \\ -0.1667 \\ 1.449 \end{bmatrix}$$

Resolvendo $U * X = C^3$, temos

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0.6667 & 0.5 \\ 0 & 0 & 1 & 1.333 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1 \\ -0.5 \\ -0.1667 \\ 1.449 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -0.8504 \\ 0.1742 \\ -2.098 \\ 1.449 \end{bmatrix}$$

Confira o algoritmo de decomposição LU , de Crout, aplicado ao

Exemplo 2.6 de três sistemas lineares com opção de pivotação parcial, no link <http://sergiopeters.prof.ufsc.br/algoritmoslivro/> sob o título **Cap2LUCrout.m**.

No algoritmo de decomposição LU , de Crout, o processo de pivotação parcial está inserido depois do cálculo dos valores de l_{ik} e imediatamente antes do cálculo dos valores de u_{kj} .

Na Tabela 2.1, retomamos o número de operações aritméticas envolvidas em cada um dos métodos numéricos diretos apresentados até este momento.

Tabela 2.1 – Comparativo entre as operações aritméticas realizadas com métodos diretos

MÉTODOS	ORDEM DO NÚMERO DE OPERAÇÕES
Gauss	$O(2n^3 / 3)$
Gauss-Jordan	$O(4n^3 / 3)$
Inversão	$O(8n^3 / 3)$
Crout	$O(2n^3 / 3)$

Fonte: Elaboração própria

Essa ordem do número de operações aritméticas define a complexidade do algoritmo de um método numérico. Os métodos de inversão de matrizes e Gauss-Jordan são computacionalmente os menos eficientes, pois envolvem o maior número de operações aritméticas, enquanto os métodos de Gauss e Crout são os que envolvem o menor esforço computacional, porém todos são de ordem $O(n^3)$ operações aritméticas.

Tabela 2.2 – Comparativo entre cada tipo de operação artimética realizada pelos dois métodos mais eficientes

OPERAÇÕES	MÉTODO DE CROUT	MÉTODO DE GAUSS
Adição e Subtração	$(2n^3 + 9n^2 - 5n - 6) / 6$	$(2n^3 + 3n^2 + n - 6) / 6$
Multiplicação	$(2n^3 + 3n^2 - 5n) / 6$	$(2n^3 + 3n^2 - 5n) / 6$
Divisão	$(n^2 + n) / 2$	$(n^2 + n) / 2$
Total	$(4n^3 + 15n^2 - 7n - 6) / 6$	$(4n^3 + 9n^2 - n - 6) / 6$

Fonte: Elaboração própria

No método de Crout, temos um pouco mais de adições e subtrações do que no método de Gauss, mas é uma diferença de ordem inferior, $6n^2$, e as outras operações são de mesmo número.

Assim, quando temos que resolver **vários sistemas** com a mesma matriz de coeficientes A e com m **diferentes vetores** de termos independentes do tipo $A \cdot X^m = B^m$, podemos:

- a) Utilizar o método de Crout para obter as matrizes decompostas L e U uma única vez e efetuar as m substituições sucessivas $L \cdot C^m = B^m$ e $U \cdot X^m = C^m$ tantas vezes quantas forem necessárias para obter as respectivas soluções C^m e X^m , correspondentes a cada B^m . Nesses casos, temos apenas um custo de $(2 * n^2 - n)$ operações, referente à substituição dupla para obter cada solução X^m , e um custo de armazenamento de duas matrizes, L e U que podem ser sobrepostas em uma única matriz, conforme feito no algoritmo de decomposição LU , de Crout, apresentado.
- b) Obter a matriz inversa A^{-1} uma única vez e aplicar apenas o produto final $X^m = A^{-1} \cdot B^m$ para cada sistema m . O custo dessa etapa de operações do produto da inversa é igual ao das duas substituições sucessivas com as matrizes L e U , e custo de armazenamento de uma única matriz.
- c) Aplicar o método de eliminação de Gauss a uma matriz estendida com todas as colunas de B^m , incluindo-as conforme esta matriz com dados do **Exemplo 2.6**:

$$\begin{aligned}
 & \left[\begin{array}{cccc|ccc} 1 & 2 & 3 & 4 & : & 1 & 1 & -1 \\ 1 & 2 & -3 & -4 & : & 0 & 0 & 0 \\ -1 & 0 & 2 & 3 & : & 1 & 1 & 1 \\ 1 & -4 & -1 & 1 & : & 2 & -2 & 2 \end{array} \right] \\
 \Rightarrow & \left[\begin{array}{cccc|ccc} 1 & 2 & 3 & 4 & : & 1 & 1 & -1 \\ 0 & -6 & -4 & -3 & : & 1 & -3 & 3 \\ 0 & 0 & -4 & -8 & : & -1 & -1 & 1 \\ 0 & 0 & 0 & 1.1111... & : & 1.7222... & 0.3888... & 1.6111... \end{array} \right]
 \end{aligned}$$

Depois de obter a matriz triangularizada dos coeficientes, que também é uma matriz U triangular superior, efetuamos as retrosubstituições sucessivas $U^*X^m = B^m$ para obter a solução X^m correspondente a cada B^m escalonado, com custo de n^2 operações, como nas seguintes soluções, aqui obtidas em precisão double:

$$X = \begin{bmatrix} -0.150000 & -0.500000 & -0.850000 \\ 0.325000 & 0.525000 & 0.175000 \\ -1.900000 & -0.300000 & -2.100000 \\ 1.550000 & 0.350000 & 1.450000 \end{bmatrix}$$

Nesse caso, não há possibilidade de reuso da matriz triangularizada, como podemos fazer com a matriz decomposta ou com a matriz inversa. Essa metodologia pode ser indicada para resolver sistemas de equações lineares que estejam dentro de um processo iterativo que atualiza os coeficientes das equações do sistema a cada iteração, não havendo necessidade de armazenar a matriz transformada para reuso posterior.

2.1.5 Método de Cholesky

No caso particular de sistemas com matrizes **simétricas e positivas definidas**, podemos obter uma simplificação no método de decomposição para obter as matrizes L e U , chamado de método de Cholesky ($O(n^3 / 6)$ operações). Com esse método, podemos obter a matriz U diretamente pela transposta de L , ou seja, $U = L^T$, ($A = L * L^T$), segundo as seguintes equações:

a) operações no primeiro passo: $k = 1$

$$l_{11} = \sqrt{a_{11}}$$

$$l_{i1} = a_{i1} / l_{11} \quad \forall i = 2, 3, \dots, n$$

b) operações nos passos: $k = 2, 3, \dots, n - 1$

$$l_{kk} = \left[a_{kk} - \sum_{r=1}^{k-1} l_{kr}^2 \right]^{1/2}$$

$$l_{ik} = \frac{1}{l_{kk}} \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} * l_{kr} \right) \quad \forall i = k+1, \dots, n$$

c) operações no último passo: $k = n$

$$l_{nn} = \left[a_{nn} - \sum_{r=1}^{n-1} l_{nr}^2 \right]^{1/2}$$

O método de Cholesky também é utilizado para verificar eficientemente se uma dada matriz simétrica A é positiva definida ou não; pois, na decomposição anterior, se não aparecerem radicandos negativos, então a matriz simétrica é positiva definida.

No **algoritmo de Cholesky**, o processo de pivotação não pode ser aplicado, para não alterar a simetria da matriz, conforme podemos conferir no arquivo **Cap2LUCholesky.m** do **Caderno de Algoritmos**.

Na sequência, vamos apresentar metodologias para solver sistemas representados em matrizes do tipo banda.

2.1.6 Solução de Sistemas com a Matriz de Coeficientes do Tipo Banda

Alguns modelos matemáticos envolvem sistemas de equações lineares especiais deste tipo:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \cdots & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \cdots & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & \cdots & a_{(n-1,n-2)} & a_{(n-1,n-1)} & a_{(n-1,n)} \\ 0 & 0 & \cdots & \cdots & 0 & a_{(n,n-1)} & a_{(n,n)} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \quad (4a)$$

Nesse sistema, a matriz dos coeficientes A é constituída de apenas três faixas de elementos não nulos: a faixa da diagonal principal e as faixas supra e infradiagonal. Esse tipo de sistema é denominado de **tridiagonal** e, para solvê-lo, é natural considerarmos que:

- a) Não é necessário armazenar toda a matriz A , mas apenas as suas três faixas; bem como é desnecessário o uso da bidimensionalidade para armazenar o sistema, mas apenas quatro vetores unidimensionais, obtendo-se considerável economia de memória no armazenamento quando a ordem n for elevada. Por exemplo, se $n = 10^6$, na representação matricial teríamos que reservar $n^2 = 10^{12}$ posições de memória para a matriz A ; já na notação vetorial seriam apenas $3n = (3 * 10^6)$, obtendo-se uma economia de 99.9997% na demanda de memória. Assim, o sistema dado pela eq. (4a) será representado conforme a equação a seguir:

$$\begin{bmatrix} r_1 & d_1 & & & & & \\ t_2 & r_2 & d_2 & & & & \\ t_3 & r_3 & d_3 & & & & \\ \ddots & \ddots & \ddots & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & t_{n-1} & r_{n-1} & d_{n-1} & & \\ & & & t_n & r_n & & \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \quad (4b)$$

Observe que no sistema (4b) não existem os elementos t_1 e d_n . Na implementação computacional desse tipo de sistema, podemos atribuir quaisquer valores para t_1 e d_n , uma vez que não serão utilizados.

Para solver o sistema dado pela eq. (4b), a eliminação gaussiana é uma metodologia eficiente, uma vez que anula apenas os elementos da faixa infradiagonal. Assim, temos os seguintes passos:

Primeiro passo: $k = 1$, eliminação do t_2 :

$$\left[\begin{array}{ccc|c} r_1 & d_1 & & b_1 \\ t_2 & r_2 & d_2 & b_2 \\ t_3 & r_3 & d_3 & \vdots \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \\ t_{n-1} & r_{n-1} & d_{n-1} & b_{n-1} \\ t_n & r_n & & b_n \end{array} \right] L_2 \leftarrow L_2 - \frac{t_2}{r_1} L_1$$

$$\left[\begin{array}{ccc|c} r_1 & d_1 & & b_1 \\ 0 & r_2 & d_2 & b_2 \\ t_3 & r_3 & d_3 & \vdots \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \\ t_{n-1} & r_{n-1} & d_{n-1} & b_{n-1} \\ t_n & r_n & & b_n \end{array} \right]$$

Nas quais devem ser operadas apenas as colunas de r_2 e b_2 :

$$r_2 = r_2 - \frac{t_2}{r_1} d_1 \text{ e } b_2 = b_2 - \frac{t_2}{r_1} b_1, \text{ pois } d_2 \text{ não é alterado, } d_2 = d_2 - \frac{t_2}{r_1} 0, \text{ e}$$

nenhuma outra coluna da linha $i = 2$ é alterada, pois $0 = 0 - \frac{t_2}{r_1} 0$.

Segundo passo: generalizando para um passo k qualquer que altera apenas a linha $i = k + 1$, temos:

$$\left[\begin{array}{ccc|c} r_1 & d_1 & & b_1 \\ 0 & r_2 & d_2 & b_2 \\ \ddots & \ddots & \ddots & \vdots \\ 0 & r_{i-1} & d_{i-1} & b_{i-1} \\ t_i & r_i & d_i & b_i \\ \ddots & \ddots & \ddots & \vdots \\ t_n & r_n & & b_n \end{array} \right] L_i \leftarrow L_i - \frac{t_i}{r_{i-1}} L_{i-1}$$

Então, geramos a matriz triangularizada a seguir:

$$\left[\begin{array}{ccc|c} r_1 & d_1 & & b_1 \\ 0 & r_2 & d_2 & b_2 \\ \ddots & \ddots & \ddots & \vdots \\ 0 & r_{i-1} & d_{i-1} & b_{i-1} \\ 0 & r_i & d_i & b_i \\ \ddots & \ddots & \ddots & \vdots \\ 0 & r_n & & b_n \end{array} \right] \quad (5)$$

Em que os novos coeficientes são $r_i = r_i - \frac{t_i}{r_{i-1}} d_{i-1}$ e $b_i = b_i - \frac{t_i}{r_{i-1}} b_{i-1}$, para $i = 2, 3, \dots, n$.

Assim, podemos gerar um algoritmo simples para efetuar a triangularização da matriz de coeficientes. Genericamente, para $i = 2, 3, \dots, n$ (equivalente a $k = 1$ até $n-1$), temos:

$$\begin{aligned} aux &= \frac{t_i}{r_{i-1}} \\ r_i &= r_i - aux * d_{i-1} \\ b_i &= b_i - aux * b_{i-1} \end{aligned}$$

Com r_1 , d_1 e b_1 inalterados na primeira linha.

Então, por retrosubstituições sucessivas aplicadas sobre as equações simplificadas, dadas pela eq. (5), obtemos:

$$x_n = b_n / r_n$$

Para $i = n-1, n-2, \dots, 2, 1$, temos:

$$x_i = (b_i - d_i * x_{i+1}) / r_i$$

Dessa forma, obtemos a solução do sistema tridiagonal com o mínimo de operações possíveis manipulando apenas os elementos não nulos.

Nesses casos, o pivotamento de linhas ou colunas não deve ser aplicado, pois isso alteraria a estrutura em forma de banda da matriz.

Esses sistemas nunca têm diagonal principal nula, pois são sistemas resultantes de modelos matemáticos de discretização de domínios nos quais são relacionados um ponto central i , com seus vizinhos anterior $i - 1$ e posterior $i + 1$, como no método das diferenças finitas (BURDEN; FAIRES, 2011), que não faz parte do escopo deste livro.

Esse algoritmo requer apenas $8n - 7$ operações, sendo $5(n - 1)$ operações para a eliminação e $3(n - 1) + 1$ operações para o procedimento de retrosubstituições.

Exemplo 2.7: resolva o sistema de equações, a seguir, de forma otimizada:

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 1 & 1 & -1 & 0 & 0 & : & 1 \\ 0 & 1 & -1 & 1 & 0 & : & 2 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right]$$

Solução:

a) Triangularização baseada no pivô $k = 1$ opera a linha $i = 2$:

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 1 & 1 & -1 & 0 & 0 & : & 1 \\ 0 & 1 & -1 & 1 & 0 & : & 2 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right] L_2 \leftarrow L_2 - \frac{1}{1} L_1$$

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 1 & -1 & 1 & 0 & : & 2 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right]$$

b) Triangularização baseada no pivô $k = 2$ opera a linha $i = 3$:

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 1 & -1 & 1 & 0 & : & 2 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right] L_3 \leftarrow L_3 - \frac{1}{2}L_2$$

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 0 & -0.5 & 1 & 0 & : & 1.5 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right]$$

c) Triangularização baseada no pivô $k = 3$ opera a linha $i = 4$:

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 0 & -0.5 & 1 & 0 & : & 1.5 \\ 0 & 0 & -1 & 1 & 1 & : & -1 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right] L_4 \leftarrow L_4 - \frac{(-1)}{(-0.5)}L_3$$

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 0 & -0.5 & 1 & 0 & : & 1.5 \\ 0 & 0 & 0 & -1 & 1 & : & -4 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right]$$

d) Triangularização baseada no pivô $k=4$ opera a linha $i=5$:

$$\left[\begin{array}{cccccc} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 0 & -0.5 & 1 & 0 & : & 1.5 \\ 0 & 0 & 0 & -1 & 1 & : & -4 \\ 0 & 0 & 0 & -1 & 2 & : & -2 \end{array} \right] L_5 \leftarrow L_5 - \frac{(-1)}{(-1)}L_4$$

$$\left[\begin{array}{cccccc|c} 1 & -1 & 0 & 0 & 0 & : & 0 \\ 0 & 2 & -1 & 0 & 0 & : & 1 \\ 0 & 0 & -0.5 & 1 & 0 & : & 1.5 \\ 0 & 0 & 0 & -1 & 1 & : & -4 \\ 0 & 0 & 0 & 0 & 1 & : & 2 \end{array} \right]$$

e) Retrosubstituições:

$$\begin{cases} 1x_5 = 2 \\ -1x_4 + 1x_5 = -4 \\ -0.5x_3 + 1x_4 = 1.5 \\ 2x_2 - 1x_3 = 1 \\ 1x_1 - 1x_2 = 0 \end{cases}$$

Então, a solução obtida é:

$$S = \{5, 5, 9, 6, 2\} \text{ (temos uma solução exata, pois não houve arredondamentos).}$$

Não devemos utilizar a pivotação no algoritmo de Gauss otimizado para matriz tridiagonal, para não alterar a sua tridiagonalidade, conforme apresentamos no algoritmo sob título **Cap2tridiagonal.m** do seu **Caderno de Algoritmos**.

No Algoritmo de Gauss para Matriz Tridiagonal, os valores de t_i e d_n foram preenchidos pelo símbolo *NaN* (*Not a Number*), para completar as n posições dos vetores t_i e d_i , e também para chamar atenção que esses valores não existem no sistema de equações e não podem ser utilizados, mas poderia ser qualquer outro valor numérico, como o zero.

Podemos estender esses métodos específicos para matrizes bandas com mais faixas, como a **matriz pentadiagonal**. Também podemos generalizar a otimização de métodos eliminativos para



Temos um exemplo de sistema com matriz pentadiagonal no exercício 2.19.

qualquer **matriz esparsa não estruturada em faixas** propondo um algoritmo que armazene as alterações necessárias em um sistema ao longo de cada passo do método eliminativo, na forma de um mapeamento de índices de cada um desses passos, e, depois, aplicando o método de eliminação com esse mapeamento predefinido para operar apenas os coeficientes não nulos necessários por quantas vezes forem necessárias.

A seguir, vamos apresentar uma característica inerente a alguns sistemas lineares que, combinada com erros de arredondamento nos coeficientes do sistema, pode afetar a confiabilidade da solução.

Confira outros exemplos de matrizes esparsas não estru-



turadas em faixas e um algoritmo de escalonamento otimizado genérico para suas soluções na seção Complementando... ao final deste Capítulo.

2.1.7 Sistemas Lineares Mal Condicionados

Consideremos o seguinte sistema linear:

$$\begin{cases} x_1 + 3x_2 = 4 \\ x_1 + 3.00001x_2 = 4.00001 \end{cases}$$

cuja solução exata é $S = \{ 1, 1 \}$.

Tomemos agora um sistema de equações, derivado do sistema dado, que sofreu uma pequena perturbação em dois de seus coeficientes, impondo uma variação ao coeficiente a_{22} de 3.00001 para 2.99999 e ao b_2 de 4.00001 para 4.00002, conforme segue:

$$\begin{cases} x_1 + 3x_2 = 4 \\ x_1 + 2.99999x_2 = 4.00002 \end{cases}$$

cuja solução exata é $\bar{S} = \{ 10, -2 \}$.

Note que essa pequena variação em dois coeficientes, da ordem de 0.025%, acarreta uma variação enorme na solução do sistema da ordem de 900%.

Classificamos esses sistemas altamente sensíveis a variações nos seus coeficientes como sistemas **mal condicionados**.

Imagine agora que essas pequenas variações são devidas a erros de arredondamento, que certamente estão presentes em todos os métodos diretos, no armazenamento e manipulação dos coeficientes das equações. Então, nesses casos, podemos obter soluções irreais dos sistemas originais, uma vez que as soluções são obtidas em sistemas simplificados, por operações sucessivas de eliminação ou decomposição, que inevitavelmente acumulam pequenas alterações nos coeficientes das matrizes simplificadas.

Por isso, é necessário tomar cuidados especiais na resolução desses sistemas, devido à grande sensibilidade da solução em relação a variações nos seus coeficientes. Então, devemos primeiramente identificar o sistema de equações mal condicionado, de preferência antes de resolver o sistema. Para tal, podemos recorrer a alguns critérios, como:

- a) Se o determinante da matriz A for considerado pequeno; ou se o determinante normalizado de A , denotado por $\|\det(A)\|$, for pequeno:

$$\|\det(A)\| = \frac{|\det(A)|}{\prod_{i=1}^n \alpha_i} \ll 1 \quad (6a)$$



O símbolo \ll
representa
muito menor.

Em que $\alpha_i = \sqrt{\sum_{i=1}^n a_{ij}^2} \quad \forall i \in \{1, 2, \dots, n\}$, e α_i mede a magnitude de cada linha i e deve ser calculado sobre a **matriz original**.

Então, se $\|\det(A)\| \ll 1$, o sistema é considerado **mal condicionado**. Na prática, adotamos um limite conservativo, por exemplo, se $\|\det(A)\| < 0.01$, consideramos conservativamente que o sistema $A \cdot X = B$ é mal condicionado e podemos adotar as devidas precauções.

- b) Se o número de condicionamento $Cond(A)$ for considerado grande, ou seja:

$$Cond(A) = \|A\|_\infty * \|A^{-1}\|_\infty \gg 1 \quad (6b)$$



O símbolo \gg
representa
muito maior.

Então, se $Cond(A) \gg 1$, o sistema é considerado mal condicionado.

A norma infinita de A , denotada por $\|A\|_\infty$, é definida da seguinte forma:

$$\|A\|_\infty = \underset{1 \leq i \leq n}{\operatorname{Max}} \left\{ \sum_{j=1}^n |a_{ij}| \right\}$$

e corresponde ao maior valor entre as somas dos módulos dos elementos de cada linha da matriz A . Também, para efeitos práticos, podemos estabelecer que, se $\operatorname{Cond}(A) > 100$, então o sistema $A * X = B$ é conservativamente mal condicionado.

A dificuldade de aplicação desse último critério está na obtenção do valor da matriz inversa A^{-1} , cujo custo é aproximadamente o triplo do custo do método de eliminação gaussiana, enquanto o primeiro critério precisa apenas do determinante de A , que podemos obter através do produto da diagonal principal da matriz triangularizada (no método de Gauss, $\det(A) = \prod_{i=1}^n a_{ii}$, em que os a_{ii} são elementos da diagonal da matriz A triangularizada) ou através do produto da diagonal principal da matriz L decomposta de A (no método LU , de Crout, $\det(A) = \prod_{i=1}^n l_{ii} * 1$, em que os l_{ii} são elementos da diagonal de L e a diagonal de U é composta de valores unitários).

Uma desvantagem desse cálculo do determinante sobre a matriz simplificada pelas operações elementares de eliminação, ou decomposição, é que essa forma final simplificada já sofreu o acúmulo de erros de arredondamento sucessivos, não é exata e pode gerar alguma inconsistência em sistemas muito mal condicionados ou singulares. Por exemplo, em matrizes singulares, podemos avaliar um determinante residual, diferente de zero, gerando um determinante não nulo inconsistente para a matriz original singular. Logo, precisamos considerar determinantes residuais como nulos, como fizemos no algoritmo de retrosubstituições anterior, que leva em conta a possibilidade de uma das linhas da matriz ser **dependente** das demais.

Exemplo 2.8: avalie o condicionamento do sistema

$$\begin{cases} x_1 + 3x_2 = 4 \\ x_1 + 3.00001x_2 = 4.00001 \end{cases}$$

Solução:

Para tal, temos de:

a) Verificar se $\|\det(A)\| \ll 1$:

$$|\det(A)| = 0.00001$$

$$\alpha_1 = \sqrt{1^2 + 3^2} = \sqrt{10}$$

$$\alpha_2 = \sqrt{1^2 + (3.00001)^2} \approx \sqrt{10}$$

$$\|\det(A)\| = \frac{0.00001}{\sqrt{10}\sqrt{10}} = 10^{-6}$$

Se $\|\det(A)\| = 10^{-6} \ll 1 \rightarrow$ mal condicionado

(conservativamente, consideraremos $\ll 1$ como < 0.01)

b) Ou verificar se $\text{Cond}(A) >> 1$:

$$A = \begin{bmatrix} 1 & 3 \\ 1 & 3.00001 \end{bmatrix}$$

$$\|A\|_{\infty} = |1| + |3.00001| = 4.0001 \approx 4$$

A norma infinita $\|A\|_{\infty}$ é o valor máximo entre as somas dos módulos dos coeficientes de cada linha da matriz.

$$A^{-1} = \begin{bmatrix} 300000.9999980346 & -299999.9999980346 \\ -99999.9999993449 & 99999.9999993449 \end{bmatrix}$$

(obtida em computador com 16 dígitos significativos)

$$\|A^{-1}\|_{\infty} \approx |300001| + |300000| \approx 600001$$

$$\text{Cond}(A) = 4.00001 * 600001 \approx 2.4 * 10^6$$

Se $\text{Cond}(A) \approx 2.4 * 10^6 >> 1 \Rightarrow$ confirma o mal condicionamento

(conservativamente, consideraremos $>> 1$ como > 100)

Depois de identificar um sistema como mal condicionado, é muito importante **minimizar os efeitos de alterações nos seus coeficientes**, então recomendamos:

- a) Usar um método de solução que não altere a forma original das equações, como é o caso dos **métodos iterativos**, que veremos mais adiante; porém, os métodos iterativos nem sempre convergem para a solução.
- b) Adotar algoritmos com o **menor número de operações aritméticas possível**, consequentemente menor acúmulo de arredondamentos. Sugerimos a aplicação de métodos diretos como a eliminação gaussiana ou a decomposição LU , uma vez que têm o menor número total de operações entre os métodos diretos apresentados, da ordem de $(O(n^3 / 3))$. Naturalmente, recomendamos que cada algoritmo seja internamente otimizado, e use o mínimo de operações aritméticas, como os que apresentamos neste livro.
- c) Usar algum processo de **pivotamento**, indispensável sempre que surgem zeros nas posições dos pivôs durante a aplicação das operações sucessivas de eliminação ou decomposição. No entanto, nos casos específicos de sistemas mal condicionados, o processo de pivotação também é indispensável para minimizar a propagação dos efeitos dos arredondamentos.
- d) Utilizar a **precisão máxima** disponível na implementação dos cálculos envolvidos na resolução do sistema. É interessante testar o sistema de equações calculando resultados com precisões diferentes, como simples e dupla (por exemplo em C: *float* e *double*), para fazer uma comparação entre os resultados e verificar se os erros decorrentes de arredondamentos sucessivos são significativos ou não.

Em sistemas de equações mal condicionados, temos uma característica peculiar na avaliação dos resíduos das equações, ou seja, nem sempre a solução mais próxima da exata gera os menores resíduos.

Por exemplo, suponha duas soluções aproximadas diferentes, S_1 e S_2 , para o sistema apresentado no **Exemplo 2.8**,

$$\begin{cases} x_1 + 3x_2 = 4 \\ x_1 + 3.00001x_2 = 4.00001 \end{cases}$$

Calculando os resíduos $\{R_1, R_2\}$ dessas duas equações, temos

$$\begin{cases} R_1 = |(x_1 + 3x_2) - 4| \\ R_2 = |(x_1 + 3.00001x_2) - 4.00001| \end{cases}$$

Para $S_1 = \{4, 0\} \Rightarrow$ Resíduos de $S_1 = \{0, 0.00001\}$ e

Para $S_2 = \{1.5, 1\} \Rightarrow$ Resíduos de $S_2 = \{0.5, 0.5\}$

Aparentemente, pela avaliação dos resíduos das equações, a solução mais próxima da exata é S_1 , pois gera menores resíduos; porém, sabendo que a solução exata é $S = \{1, 1\}$, percebemos que a solução S_2 está mais próxima da exata do que S_1 .

Concluímos que a avaliação dos resíduos das equações de um sistema não pode ser generalizada como critério para análise da proximidade de uma solução em relação ao seu valor exato.

Ainda sobre os métodos diretos, observamos que:

- a) Em sistemas de equações lineares que relacionam incógnitas com **grandezas de magnitudes diferentes**, como metros e milímetros, podem existir coeficientes também com magnitudes bastante diferentes, por isso também recomendamos o uso de processos de pivotação.
- b) Todos os **métodos diretos**, para resolução de sistemas de equações lineares, não apresentam **erros de truncamento**, pois são métodos teoricamente exatos. Esses têm apenas erros de arredondamento acumulados, que podemos avaliar comparando a solução aproximada obtida com outra

solução com mais dígitos exatos (por exemplo, com o dobro de dígitos significativos, se disponível).

2.2 Solução de Sistemas Lineares por Métodos Iterativos

Quando o sistema de equações lineares $A * X = B$ tiver algumas características especiais, como: a ordem n elevada (n é o número de equações); a matriz dos coeficientes A possuir grande quantidade de elementos nulos (matriz esparsa); e os coeficientes puderem ser gerados através de alguma lei de formação; em geral, será mais eficiente resolvê-lo através de um método iterativo (repetitivo), **sem operar com os coeficientes nulos**, desde que a convergência seja possível.

Se um algoritmo de um método direto, como o do método de Gauss, for aplicado a um sistema com matriz de coeficientes esparsa, muitas operações aritméticas desnecessárias serão empregadas, por exemplo, na manipulação de zeros com outros zeros, a menos que sejam evitadas pelo próprio algoritmo, como em 2.1.6.

Uma solução iterativa de $A * X = B$ consiste em tomar uma solução inicial $X^{(0)}$ (estimada de alguma forma) para a solução $S = \{x_1, x_2, x_3, \dots, x_n\}$ procurada e construir uma sequência $X^{(k)} = \{x_i^{(k)}\}_{k=0}^{\infty}$ de soluções aproximadas tal que, no limite:

$$\lim_{k \rightarrow \infty} X^{(k)} = S, \text{ se essa sequência for convergente.}$$

Observe que as soluções de sistemas de equações lineares obtidas por **métodos iterativos** são soluções aproximadas a cada iteração k (contador de repetições), portanto têm **erros de truncamento**, além de erros de arredondamento; mas, nesse caso, os arredondamentos não são acumulados, uma vez que os coeficientes das equações não são alterados ao longo do processo.

Então, a questão fundamental é como gerar essa sequência convergente.

Em $A * X = B$, tomando a matriz A e particionando-a na forma $A = C + E$, com C não singular, temos:

$$(C + E) * X = B \Rightarrow C * X = B - E * X \Rightarrow X = C^{-1} (B - E * X) \quad (7)$$

Então, aplicando uma solução inicial estimada $X^{(0)} = \{x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \dots, x_n^{(0)}\}$ no lado direito da eq. (7), temos:

$$X^{(1)} = C^{-1} (B - E * X^{(0)})$$

fazendo $X^{(0)} = X^{(1)}$ e reaplicando-o no lado direito da eq. (7), obtemos um $x^{(2)}$ e assim sucessivamente, resultando em uma sequência $x^{(k+1)}$ de soluções S aproximadas:

$$\begin{aligned} X^{(2)} &= C^{-1} * (B - E * X^{(1)}) \\ X^{(3)} &= C^{-1} * (B - E * X^{(2)}) \\ &\vdots \\ X^{(k+1)} &= C^{-1} * (B - E * X^{(k)}) \end{aligned} \quad (8)$$

em que

$$S = \left\{ X^{(k)} \right\}_{k=0}^{\infty}$$

Se a sequência dada na eq. (8) for convergente, então existe:

$$\lim_{k \rightarrow \infty} X^{(k+1)} = \lim_{k \rightarrow \infty} X^{(k)} \quad (9)$$

Aplicando o limite a ambos os lados da eq. (8), temos:

$$\lim_{k \rightarrow \infty} X^{(k+1)} = C^{-1} \left(B - E * \lim_{k \rightarrow \infty} X^{(k)} \right) \quad (10)$$

Aplicando a eq. (9) na eq. (10), temos:

$$\lim_{k \rightarrow \infty} X^{(k+1)} = C^{-1} * B - C^{-1} * E * \lim_{k \rightarrow \infty} X^{(k+1)} \Rightarrow (1 + C^{-1} * E) * \lim_{k \rightarrow \infty} X^{(k+1)} = C^{-1} * B \quad (11)$$

Multiplicando a eq. (11) pela matriz C , temos:

$$(C + E) * \lim_{k \rightarrow \infty} X^{(k+1)} = B \quad (12)$$

Mas $(C + E) = A$, então:

$$A * \lim_{k \rightarrow \infty} X^{(k+1)} = B$$

Logo,

$$\lim_{k \rightarrow \infty} X^{(k+1)} = S \text{ é a solução de } A * X = B$$

O problema agora é que, para gerar a sequência (8), necessitamos da matriz inversa C^{-1} . Portanto, essa inversa, por questão de eficiência, tem que ser obtida facilmente.

A seguir, vamos abordar duas maneiras, não exclusivas, de partitionar a matriz A de modo que a matriz C seja trivialmente invertida.

2.2.1 Método Iterativo de Jacobi

No sistema $A * X = B$, tomamos a matriz A e a partitionamos na forma $A = L + D + U$, em que:

- a) $D =$ diagonal principal de A ;
- b) $L =$ matriz triangular inferior estrita, obtida apenas dos elementos infradiagonais de A ; e
- c) $U =$ matriz triangular superior estrita, obtida apenas dos elementos supradiagonais de A .

Então, considerando que $C = D \Rightarrow E = L + U$ e aplicando-as na eq. (8), temos:

$$X^{(k+1)} = D^{-1} * (B - (L + U) * X^{(k)}) \quad (13)$$

Obtemos trivialmente os elementos de D^{-1} tomando os recíprocos dos coeficientes da diagonal principal D de A . Assim, expressando a eq. (13) na forma desenvolvida para cada equação i , resulta que:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j < i}}^{j=i} a_{ij} x_j^{(k)} - \sum_{\substack{j=i+1 \\ j > i}}^n a_{ij} x_j^{(k)} \right) \quad (14)$$

Então, para resolver $A^*X = B$ pelo método Jacobi, tomamos uma solução inicial $X^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\}$, isolamos diretamente a i -ésima incógnita x_i na i -ésima equação i , e aplicamos a eq. (14) somente com os coeficientes a_{ij} **não nulos** necessários (os coeficientes nulos não influenciam no resultado e não devem ser considerados nos cálculos para fins de otimização). Esses algoritmos são específicos para cada sistema, pois usam apenas os coeficientes não nulos, que devem ser identificados por algum mapeamento prévio.



Para esse sistema de pequeno porte, com coeficientes não nulos,

seria recomendado usar um método direto, mas aplicaremos um método iterativo para permitir a reprodução manual do processo envolvido, com objetivos didáticos.

Exemplo 2.9: resolva o **sistema**, a seguir, pelo método de Jacobi:

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 \\ x_1 + 3x_2 + x_3 = 5 \\ x_1 - x_2 + 2x_3 = 2 \end{cases}$$

Solução:

Iniciamos montando as equações evolutivas para cada incógnita do sistema e isolando x_i da sua respectiva equação i . Na forma matricial, é equivalente a isolar os valores de x_i de cada diagonal principal:

$$\begin{cases} x_1^{(k+1)} = (1 + x_2^{(k)} + x_3^{(k)}) / 3 \\ x_2^{(k+1)} = (5 - x_1^{(k)} - x_3^{(k)}) / 3 \\ x_3^{(k+1)} = (2 - x_1^{(k)} + x_2^{(k)}) / 2 \end{cases}$$

Valor inicial: $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0, 0, 0)$

Consideremos k como passo iterativo, no qual uma nova solução $x_i^{(k)}$ é calculada. Assim, usando a solução inicial em $k = 0$, podemos calcular a solução em $k + 1$ a seguir:

$$\begin{cases} x_1^{(1)} = (1+0+0)/3 = 0.333 \\ x_2^{(1)} = (5-0-0)/3 = 1.667 \\ x_3^{(1)} = (2-0+0)/2 = 1.000 \end{cases}$$

A partir dessa primeira solução aproximada, atualizamos os valores iniciais $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$ (0.333, 1.667, 1.000) e calculamos uma segunda solução aproximada e assim sucessivamente, conforme a Tabela 2.3:

Tabela 2.3 – Valores da solução aproximada para cada iteração pelo método de Jacobi

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0	0	0
1	0.333	1.667	1
2	1.222	1.222	1.667
3	1.296	0.704	1
4	0.901	0.901	0.704
5	0.868	1.132	1
6	1.044	1.044	1.132

Fonte: Elaboração própria

Note que a aproximação $(x_1^{(6)}, x_2^{(6)}, x_3^{(6)}) = (1.044, 1.044, 1.132)$, obtida em 6 iterações, está convergindo para uma solução S (nesse caso, $S_{exato} = \{1, 1, 1\}$).

Neste ponto, precisamos estabelecer um critério de parada que determine quão próxima da solução exata está a sequência convergente $x_i^{(k)}$, sem conhecermos a solução exata. Entre os critérios mais comuns, estão:

a) $\text{Max} \left\{ |x_i^{(k+1)} - x_i^{(k)}| \right\} \leq \varepsilon \Rightarrow i = 1, 2, 3, \dots, n$

Corresponde à máxima diferença absoluta entre os valores da iteração nova e da iteração anterior, entre todas as incógnitas i .

$$\text{b)} \ Max \left\{ \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| \right\} \leq \varepsilon \Rightarrow i = 1, 2, 3, \dots, n$$

Corresponde à máxima diferença relativa entre os valores da iteração nova e da iteração anterior, entre todas as incógnitas i .

$$\text{c)} \ Max \left\{ |r_i^{(k+1)}| \right\} \leq \varepsilon \Rightarrow i = 1, 2, 3, \dots, n$$

Corresponde ao maior resíduo entre todas as equações

$i, r_i^{(k+1)} = b_i - \sum_{j=1}^n a_{ij} x_j^{(k+1)}$, que já foi aplicado para aferição de soluções de métodos diretos, mas pode gerar valores inconsistentes para sistemas mal condicionados.

No caso do **Exemplo 2.9**, quando aplicamos o critério $\text{Max} \left\{ |x_i^{(k+1)} - x_i^{(k)}| \right\} \leq \varepsilon$, das diferenças de valores sucessivos, temos:

Tabela 2.4 – Valores da solução aproximada pelo método de Jacobi com critério de parada

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$ x_1^{(k+1)} - x_1^{(k)} $	$ x_2^{(k+1)} - x_2^{(k)} $	$ x_3^{(k+1)} - x_3^{(k)} $
0	0	0	0	-	-	-
1	0.333	1.667	1	0.333	1.667	1
2	1.222	1.222	1.667	0.889	0.555	0.667
3	1.296	0.704	1	0.074	0.518	0.667
4	0.901	0.901	0.704	0.395	0.197	0.296
5	0.868	1.132	1	0.033	0.197	0.296
6	1.044	1.044	1.132	0.176	0.088	0.132

Fonte: Elaboração própria

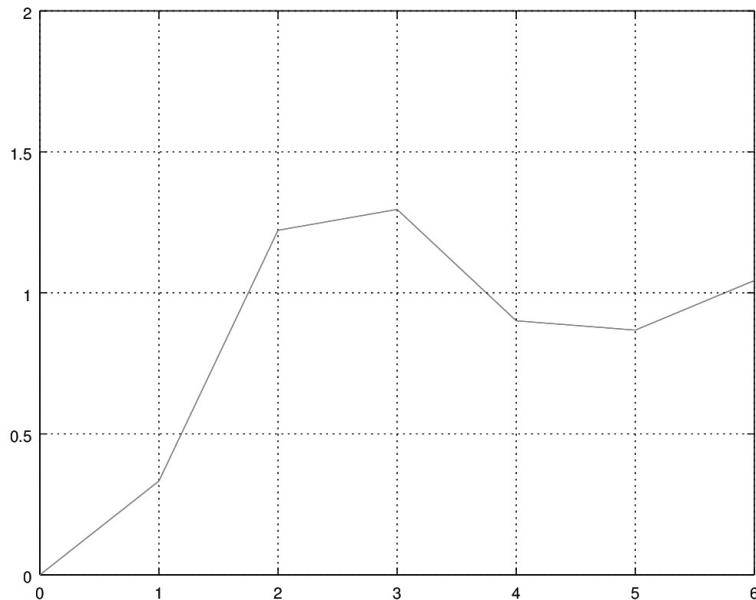
Portanto, no **Exemplo 2.9**, o critério alcançado depois de 6 iterações é:

$$\text{Max} |x_i^{(k+1)} - x_i^{(k)}| = 0.176$$

Note que, nesse exemplo, o processo iterativo é do tipo oscilatório, isto é, as incógnitas aumentam e diminuem ao longo das iterações, o que prejudica a convergência, tornando o processo mais lento.

Veja, no Gráfico 2.1, a evolução de $x_1^{(k)}$ (no eixo vertical) com o nível iterativo k (no eixo horizontal) usando o método de Jacobi.

Gráfico 2.1 – Evolução de $x_1^{(k)}$ (no eixo vertical) com o nível iterativo k (no eixo horizontal) usando o método de Jacobi



Fonte: Elaboração própria

A seguir, vamos apresentar um método iterativo normalmente mais eficiente do que o método de Jacobi.

2.2.2 Método Iterativo de Gauss-Seidel

No método iterativo de Gauss-Seidel, temos $A \cdot X = B$ com a partição de A na forma $A = L + D + U$, como no método de Jacobi, porém tomando $C = D + L$, $E = U$ e aplicando na eq. (8):

$$X^{(k+1)} = (D + L)^{-1} (B - U \cdot X^{(k)}) \quad (15)$$

Multiplicando a eq. (15) pela matriz $(D+L)$, temos:

$$\begin{aligned} (D+L)X^{(k+1)} &= B - U * X^{(k)} \\ D * X^{(k+1)} &= B - L * X^{(k+1)} - U * X^{(k)} \\ X^{(k+1)} &= D^{-1}(B - L * X^{(k+1)} - U * X^{(k)}) \end{aligned} \quad (16)$$

que, na forma desenvolvida, torna-se:

$$x_i^{(k+1)} = (b_i - \sum_{\substack{j=1 \\ j < i}}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{\substack{j=i+1 \\ j > i}}^n a_{ij} x_j^{(k)}) / a_{ii} \Rightarrow i = 1, 2, \dots, n \quad (17)$$

Essa operacionalização é semelhante à do método de Jacobi, exceto que, na eq. (16), utilizamos sempre os valores de x_j **mais atualizados** possíveis no lado direito da equação iterativa, ou seja, já são usados os valores anteriormente calculados de $x_j^{(k+1)}$ das equações com $j < i$ dentro da própria iteração em andamento e, em (14), utilizamos apenas os valores de $x_j^{(k)}$ da iteração anterior, $\forall j$.

Exemplo 2.10: resolva o sistema, a seguir, pelo método de Gauss-Seidel:

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 \\ x_1 + 3x_2 + x_3 = 5 \\ x_1 - x_2 + 2x_3 = 2 \end{cases}$$

Solução:

Iniciamos montando as equações evolutivas para cada incógnita do sistema e isolando x_i da respectiva equação i , mas usando as incógnitas mais atualizadas disponíveis no lado direito das equações, que correspondem às aquelas multiplicadas pelos coeficientes de L da eq. (16):

$$\begin{cases} x_1^{(k+1)} = (1 + x_2^{(k)} + x_3^{(k)}) / 3 \\ x_2^{(k+1)} = (5 - x_1^{(k+1)} - x_3^{(k)}) / 3 \\ x_3^{(k+1)} = (2 - x_1^{(k+1)} + x_2^{(k+1)}) / 2 \end{cases}$$

Note que as equações evolutivas utilizam os valores disponíveis mais atualizados possíveis.

Valor inicial: $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0, 0, 0)$

Assim, usando a solução inicial em $k = 0$, podemos calcular a solução em $k + 1$ a seguir:

$$\begin{cases} x_1^{(1)} = (1 + 0 + 0) / 3 = 0.333 \\ x_2^{(1)} = (5 - 0.333 - 0) / 3 = 1.555 \\ x_3^{(1)} = (2 - 0.333 + 1.555) / 2 = 1.611 \end{cases}$$

Observe que $x_1^{(1)} = 0.333$, calculado na primeira equação $i = 1$, já foi utilizado na equação de $x_2^{(1)}$ e assim por diante, conforme a Tabela 2.5.

Tabela 2.5 – Valores da solução aproximada pelo método de Gauss-Seidel

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$ x_1^{(k+1)} - x_1^{(k)} $	$ x_2^{(k+1)} - x_2^{(k)} $	$ x_3^{(k+1)} - x_3^{(k)} $
0	0	0	0	-	-	-
1	0.333	1.555	1.611	0.333	1.555	1.611
2	1.388	0.666	0.638	1.055	0.888	0.972
3	0.768	1.197	1.214	0.620	0.531	0.575
4	1.137	0.882	0.872	0.368	0.314	0.341
5	0.918	1.069	1.075	0.218	0.186	0.202
6	1.048	0.958	0.955	0.129	0.110	0.120

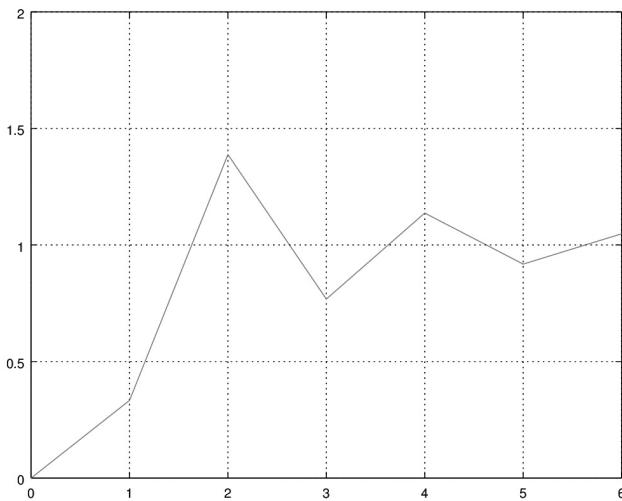
Fonte: Elaboração própria

Temos a solução aproximada $(x_1^{(6)}, x_2^{(6)}, x_3^{(6)}) = (1.048, 0.958, 0.955)$ com critério de parada $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| = 0.129$ atingido em 6 iterações.

Nesse exemplo, o processo iterativo correspondente à aplicação do método de Gauss-Seidel também é um processo oscilatório (podemos verificar que $x_i^{(k+1)} - x_i^{(k)}$ tem sinais alternados), mas agora temos um processo de convergência um pouco mais rápido porque no método de Gauss-Seidel são tomados os valores disponíveis mais atualizados.

Veja, no Gráfico 2.2, a evolução também oscilatória de x_1 (no eixo vertical) com o nível iterativo k (no eixo horizontal) usando o método de Gauss-Seidel.

Gráfico 2.2 – Evolução de x_1 (no eixo vertical) com o nível iterativo k (no eixo horizontal) usando o método de Gauss-Seidel



Fonte: Elaboração própria

A seguir, vamos apresentar condições suficientes para que um processo iterativo, de soluções de sistemas lineares, seja convergente.

2.2.3 Convergência dos Métodos Iterativos

Nas seções anteriores, desenvolvemos duas formas iterativas, eqs. (14) e (17), de construir uma sequência $\{X^{(k)}\}_{k=0}^{\infty}$ de possíveis aproximações para a solução de $A \cdot X = B$. Agora, abordaremos quando se dá a convergência dessa sequência. Para tal, necessitamos de dois conceitos adicionais:

Definição 1: no sistema $A \cdot X = B$, sejam $Z_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$, $i = 1, \dots, n$, teremos **diagonal dominante** se ocorrer:

a) $|a_{ii}| \geq Z_i$ para $i = 1, \dots, n$;

e

b) $|a_{ii}| > Z_i$, para pelo menos uma linha i de A .

Definição 2: se $|a_{ii}| > Z_i$, $\forall i = 1, \dots, n$, o sistema $A * X = B$ terá **diagonal estritamente dominante**.

2.2.3.1 Teorema Convergência – Critério de Scarborough

Segundo o teorema de convergência de Scarborough: se o sistema $A * X = B$ tiver **diagonal dominante**, ou **diagonal estritamente dominante**, tanto a sequência construída pelo método de Jacobi quanto a pelo de Gauss-Seidel convergirão para a solução S .

Veja a aplicação da Definição 1 aos **Exemplos 2.9 e 2.10**, conforme a análise a seguir:

a) Se $|a_{ii}| \geq Z_i$, $i = 1, \dots, n$ para todas as linhas i :

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 & |3| \geq |-1| + |-1| \quad V \\ x_1 + 3x_2 + x_3 = 5 & |1| \geq |+1| + |+1| \quad V \\ x_1 - x_2 + 2x_3 = 2 & |2| \geq |+1| + |-1| \quad V \end{cases}$$

temos uma condição verdadeira, pois o módulo de cada elemento da diagonal principal é maior ou igual a soma dos módulos dos demais elementos da respectiva linha.

b) Se $|a_{ii}| > Z_i$, $i = 1, \dots, n$ para alguma linha i :

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 & |3| > |-1| + |-1| \quad V \\ x_1 + 3x_2 + x_3 = 5 & |1| > |+1| + |+1| \quad V \\ x_1 - x_2 + 2x_3 = 2 & |2| \geq |+1| + |-1| \end{cases}$$

A condição também verdadeira para as duas primeiras equações.

Logo, esse sistema, resolvido nos **Exemplos 2.9 e 2.10**, satisfaz o critério de **Scarborough** da diagonal dominante e portanto tem convergência garantida.

Com relação à convergência dos métodos iterativos, algumas características importantes devem ser salientadas, entre as quais destacamos:

- a) A convergência para a solução $S = \lim \{X^{(k)}\}_{k=0}^{\infty}$ não depende do valor inicial $X^{(0)}$. Portanto, a escolha do $X^{(0)}$ adequado afeta apenas a quantidade de iterações necessárias. Quanto mais próxima a solução estimada $X^{(0)}$ estiver da solução exata S , mais rápida será a convergência (se convergir).
- b) O **teorema da convergência** contém uma condição suficiente, porém não necessária. Portanto, se esse teorema for verdadeiro, a sequência S convergirá; se não for, nada podemos afirmar.

Exemplo 2.11: verifique que, em $\begin{cases} x_1 + 2x_2 = 3 \\ x_1 - 3x_2 = -2 \end{cases}$, Gauss-Seidel fornece uma sequência convergente, mesmo não satisfazendo o teorema da convergência:

$$\begin{cases} x_1 = 3 - 2x_2 \\ x_2 = (2 + x_1)/3 \end{cases}$$

Na Tabela 2.6, temos a evolução iterativa da solução aproximada por Gauss-Seidel:

Tabela 2.6 – Evolução iterativa da solução aproximada por Gauss-Seidel

k	x_1	x_2
0	0	0
1	3	0.666667
2	1.666667	1.666667
3	-0.333333	1.222222
4	0.555556	0.555556
5	1.888889	0.851852
6	1.296296	1.296296
20	0.982658	0.982658

30	1.002284	1.002284
40	0.999699	0.999699
50	1.000040	1.000040
60	0.999995	0.999995
70	1.000001	1.000001
76	1	1

Fonte: Elaboração própria

- c) Um sistema que não tenha diagonal dominante pode, em alguns casos, ser transformado para ter diagonal dominante através de troca de linhas e/ou colunas (pivotamento parcial ou total).
- d) O critério de convergência de **Scarborough** enunciado anteriormente não é o único. Podemos verificar na literatura outros critérios de convergência.
- e) O teorema de convergência, baseado na dominância da diagonal principal da matriz de coeficientes do sistema, indica uma redução dos erros da solução aproximada em cada iteração. Por exemplo, no sistema dos **Exemplos 2.9** e **2.10**:

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 \\ x_1 + 3x_2 + x_3 = 5 \\ x_1 - x_2 + 2x_3 = 2 \end{cases}$$

temos as seguintes equações iterativas:

$$\begin{cases} x_1^{(k+1)} = (1 + 1x_2^{(k)} + 1x_3^{(k)}) / 3 \\ x_2^{(k+1)} = (5 - 1x_1^{(k+1)} - 1x_3^{(k)}) / 3 \\ x_3^{(k+1)} = (2 - 1x_1^{(k+1)} + 1x_2^{(k+1)}) / 2 \end{cases}$$

Na equação de x_1 do sistema anterior, a solução inicial $X^{(0)}$ é nula, logo $X^{(0)}$ tem um erro unitário em todas as incógnitas (pois a solução exata é 1 para cada incógnita). Observe que a solução para x_1 na primeira iteração é 0.333, logo tem um erro menor do que 1.0, ou seja, erro de 0.667. Isso decorre da

redução de erros promovida pela própria equação iterativa de x_1 , na qual os dois valores de x_2 e x_3 , cada um com seu erro interno 1.0, são multiplicados por coeficientes unitários e divididos pela diagonal principal 3.0, logo ocorre uma redução de erro aplicado na equação de x_1 , passando do erro inicial que era 1.0 para o novo erro 2/3 em $x_1 = 0.333$. Assim, **quanto maior for a diagonal principal**, mais dominante esta será e **maior** também será a **redução de erro** da solução de uma iteração para outra.

f) No sistema $\begin{cases} x_1 + 2x_2 = 3 \\ x_1 - 3x_2 = -2 \end{cases}$ do **Exemplo 2.11**, que não tem diagonal dominante e mesmo assim converge, as equações iterativas são:

$$\begin{cases} x_1 = 3 - 2x_2 \\ x_2 = (2 + x_1)/3 \end{cases}$$

Observe que na equação de x_1 ocorre uma amplificação do erro inicial de x_2 , que é multiplicado por 2 e propagado para x_1 , mas na 2ª equação, de x_2 , ocorre uma redução do erro existente em x_1 , que é multiplicado por 1/3 e depois atribuído a x_2 . Assim, ocorre uma redução global de erros promovida mais fortemente pela equação de x_2 . Note que a segunda equação tem diagonal bem dominante $| -3 | > | 1 |$ e acaba compensando a não dominância da diagonal da primeira equação $| 1 | < | 2 |$.

A seguir, vamos apresentar um artifício matemático para tentar otimizar um processo iterativo.

2.2.4 Aplicação de Coeficientes de Relaxação

Nos casos de sistemas de equações lineares com diagonal principal pouco dominante, o processo iterativo poderá convergir

oscilando, ou muito lentamente, por isso recomendamos usar fatores de sub ou sobre-relaxação, para tentar reduzir o número de iterações. Nos casos de sistemas que não tenham diagonal dominante e o Gauss-Seidel seja divergente, a sub-relaxação pode torná-lo convergente.

Por exemplo, para obter o valor mais atualizado de uma incógnita iterativa $x_i^{(k+1)}$ qualquer, podemos escrevê-la da seguinte forma:

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k+1)}$$

em que $x_i^{(k)}$ é o valor antigo, do passo iterativo k ; e $\Delta x_i^{(k+1)}$ é o valor da atualização imposta ao valor antigo $x_i^{(k)}$ para atingir o valor novo $x_i^{(k+1)}$ do passo iterativo $k + 1$.

Quando o processo iterativo não converge bem, devemos tentar impor um fator artifical na atualização do valor novo, ou seja, não aplicar 100% da atualização calculada $\Delta x_i^{(k+1)}$ sobre o valor antigo $x_i^{(k)}$, por meio de um fator de relaxação λ , com $0 < \lambda < 2$, da seguinte forma:

$$x_i^{(k+1)} = x_i^{(k)} + \lambda * \Delta x_i^{(k+1)} \quad (18)$$

Quando o processo iterativo converge **oscilando**, ou até mesmo quando diverge, devemos tentar aplicar um fator de redução, desaceleração ou amortecimento na atualização do valor novo, ou seja, impomos um fator de **sub-relaxação** λ , com $0 < \lambda < 1$ na eq. (18).

Quando o processo iterativo converge **lentamente**, podemos tentar aplicar um fator de ampliação ou aceleração na atualização do valor novo, ou seja, impomos um fator de **sobrerrelaxação** λ , com $1 < \lambda < 2$, também na eq. (18).

Observe que, se $\lambda = 1$, temos a equação evolutiva original $x_i^{(k+1)} = x_i^{(k)} + 1 * \Delta x_i^{(k+1)}$, com 100% da atualização calculada pelo método original. Alternativamente, podemos reescrever o valor da atualização total (incremento iterativo original do método) $\Delta x_i^{(k+1)} = x_i^{(k+1)} - x_i^{(k)}$ dentro da equação anterior, que gera uma forma alternativa:

$$\begin{aligned}x_i^{(k+1)} &= x_i^{(k)} + \lambda (x_i^{(k+1)} - x_i^{(k)}) \\x_i^{(k+1)} &= (1-\lambda) x_i^{(k)} + \lambda * x_i^{(k+1)}\end{aligned}\quad (19)$$

Por exemplo, a aplicação de relaxação para o método de Gauss-Seidel, dada pela eq. (17), gera a seguinte equação geral:

$$x_i^{(k+1)} = (1-\lambda)x_i^{(k)} + \lambda \frac{1}{a_{ii}} (b_i - \sum_{\substack{j=1 \\ j < i}}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{\substack{j=i+1 \\ j > i}}^n a_{ij}x_j^{(k)}) \Rightarrow i = 1, 2, \dots, n \quad (20)$$

Lembramos que a eq. (20) somente deve ser aplicada com os coeficientes a_{ij} não nulos, para evitar operações aritméticas desnecessárias.

Exemplo 2.12: resolva o sistema, a seguir, pelo método de Gauss-Seidel adotando uma sub-relaxação $\lambda = 0.5$ (lembrando que, no **Exemplo 2.10**, a solução sofreu oscilações ao longo das iterações, logo é indicada uma sub-relaxação).

$$\begin{cases} 3x_1 - x_2 - x_3 = 1 \\ x_1 + 3x_2 + x_3 = 5 \\ x_1 - x_2 + 2x_3 = 2 \end{cases}$$

Solução:

Montando as equações evolutivas para cada incógnita do sistema, conforme a eq. (20), temos:

$$\begin{cases} x_1^{(k+1)} = (1-\lambda)x_1^{(k)} + \lambda (1 + x_2^{(k)} + x_3^{(k)}) / 3 \\ x_2^{(k+1)} = (1-\lambda)x_2^{(k)} + \lambda (5 - x_1^{(k+1)} - x_3^{(k)}) / 3 \\ x_3^{(k+1)} = (1-\lambda)x_3^{(k)} + \lambda (2 - x_1^{(k+1)} + x_2^{(k+1)}) / 2 \end{cases}$$

Valor inicial: $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0, 0, 0)$ e $\lambda = 0.5$.

Na Tabela 2.7, temos os valores de 6 iterações da solução aproximada com fator de sub-relaxação $\lambda = 0.5$.

Tabela 2.7 – Valores da solução aproximada pelo método de Gauss-Seidel com fator de sub-relaxação

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$ x_1^{(k+1)} - x_1^{(k)} $	$ x_2^{(k+1)} - x_2^{(k)} $	$ x_3^{(k+1)} - x_3^{(k)} $
0	0	0	0	-	-	-
1	0.166	0.805	0.659	0.166	0.805	0.659
2	0.494	1.043	0.967	0.327	0.238	0.307
3	0.748	1.069	1.063	0.254	0.025	0.096
4	0.896	1.041	1.067	0.147	0.027	0.004
5	0.966	1.014	1.046	0.069	0.026	0.021
6	0.993	1.000	1.024	0.026	0.014	0.021

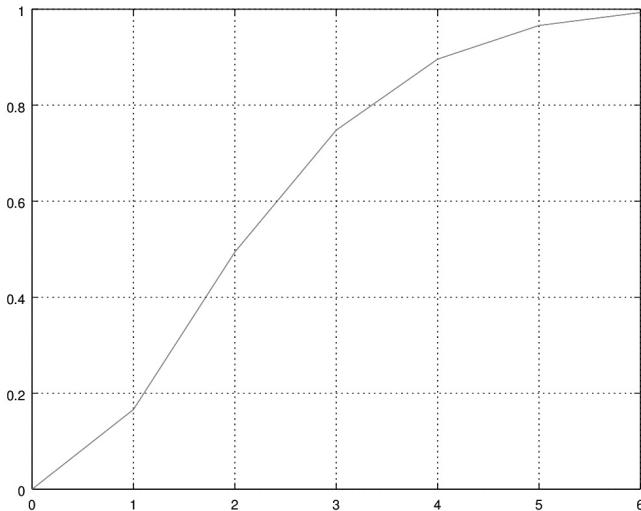
Fonte: Elaboração própria

A solução aproximada $(x_1^{(6)}, x_2^{(6)}, x_3^{(6)}) = (0.993, 1.000, 1.024)$ foi atingida com critério de parada $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| = 0.026$ em 6 iterações.

O processo iterativo com o fator de sub-relaxação ficou mais estável, gerando um processo de convergência monotônico (diferenças entre iterações sucessivas $x_i^{(k+1)} - x_i^{(k)}$ mantém os mesmos sinais para cada passo iterativo k), e consequentemente temos um processo iterativo mais rápido.

Note que, com as mesmas 6 iterações do método de Gauss-Seidel, atingimos um critério de parada máximo de 0.026 com sub-relaxação, contra 0.129 sem sub-relaxação. O Gráfico 2.3 demonstra a evolução de $x_i^{(k)}$ (no eixo vertical) e nível iterativo k (no eixo horizontal) usando o método de Gauss-Seidel e sub-relaxação $\lambda = 0.5$.

Gráfico 2.3 – Evolução de $x_1^{(k)}$ (no eixo vertical) e nível iterativo k (no eixo horizontal) usando o método de Gauss-Seidel e sub-relaxação $\lambda = 0.5$



Fonte: Elaboração própria

Recomendamos sempre aplicar previamente um teste com fatores de relaxação. Você pode experimentar pelo menos um fator de relaxação menor do que 1 (0.9) ou outro maior do que 1 (1.1) e avaliar os seus efeitos, se aumentam ou diminuem as iterações. Esses testes podem ser feitos com critérios de parada mais grosseiros, para serem mais rápidos. Também é importante buscar um fator de relaxação “ótimo”, que promova o menor número de iterações totais.

Ficou claro até este momento que as soluções obtidas para os sistemas de equações lineares são aproximadas e precisam ter um grau de confiabilidade definido. Então, quais são os erros associados a essa solução S ?

No **Exemplo 2.12**, vimos que os erros de arredondamento da solução S , obtida com apenas 4 dígitos totais, estavam presentes a partir do terceiro dígito fracionário: $X^{(6)} = \{0.993, 1.000, 1.024\}$. Lembremos

que, nos métodos iterativos, o arredondamento não se propaga para dígitos mais significativos como nos métodos eliminativos. Então, se quisermos resultados com mais dígitos significativos, devemos usar precisão maior, como a variável *double* de 16 dígitos significativos, e teríamos resultados conforme a Tabela 2.8.

Tabela 2.8 – Valores da solução aproximada pelo método de Gauss-Seidel e com fator de sub-relaxação em precisão *double*

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\text{Max } x_i^{(k+1)} - x_i^{(k)} $
0	0	0	0	-
1	0.1666666666666667	0.8055555555555555	0.6597222222222222	0.8055555555555555
2	0.494212962962963	1.043788580246910	0.967255015432099	0.327546296296296
3	0.748947080761317	1.069193940757890	1.063689222715190	0.254734117798354
4	0.896620734292838	1.041211977544270	1.067992422170450	0.147673653531521
5	0.966511100432207	1.014855401671690	1.046082286395100	0.069890366139369
6	0.993411831560569	1.000845347843240	1.024899522268220	0.026900731128362

Fonte: Elaboração própria

Observe que as soluções de sistemas de equações lineares obtidas por métodos **iterativos** são soluções **aproximadas** a cada iteração, portanto têm **erros de truncamento**, além dos erros de arredondamento presentes no último dígito significativo, logo devemos ter uma forma de avaliar esses erros de truncamento associados a cada solução aproximada obtida.

Por exemplo, qual é o erro de truncamento máximo da solução obtida anteriormente em 6 iterações?

$$X^{(6)} = \{0.9934118315605686, 1.0008453478432351, 1.0248995222682158\}$$

Considerando que os erros de arredondamento foram minimizados ao máximo (pelo uso da variável *double*), podemos estimar uma solução *S* mais próxima da exata minimizando ao máximo os erros de truncamento decorrentes das aproximações, ou seja, podemos teoricamente executar infinitas aproximações, pois no limite de iterações:

$$\lim_{k \rightarrow \infty} X^{(k+1)} = S \text{ é a solução de } A * X = B$$

Ou seja, depois de infinitas iterações, vamos atingir a solução exata do sistema. Nesse exemplo, com poucas equações, podemos fazer iterações até que o critério de parada seja o mínimo possível para as variáveis *double* utilizadas, chegando aos resultados apresentados na Tabela 2.9.

Tabela 2.9 – Valores da solução aproximada pelo método de Gauss-Seidel com fator de sub-relaxação no limite da variável *double*

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\text{Max } x_i^{(k+1)} - x_i^{(k)} $
0	0	0	0	-
1	0.1666666666666667	0.8055555555555555	0.6597222222222222	0.8055555555555555
2	0.494212962962963	1.043788580246910	0.967255015432099	0.327546296296296
3	0.748947080761317	1.069193940757890	1.063689222715190	0.254734117798354
4	0.896620734292838	1.041211977544270	1.067992422170450	0.147673653531521
5	0.966511100432207	1.014855401671690	1.046082286395100	0.069890366139369
6	0.993411831560569	1.000845347843240	1.024899522268220	0.026900731128362
12	0.999754544313626	0.999648218060907	0.999924062782415	4.66504270821e-04
55	1	1	1	0.

Fonte: Elaboração própria

Note que, na iteração 55, o critério de parada $\text{Max } |x_i^{(k+1)} - x_i^{(k)}|$ atingiu valor nulo, portanto o algoritmo chegou ao limite da precisão digital disponível, de modo que a solução obtida não vai mais variar para esta precisão *double*:

$$S = \{ 1, 1, 1 \}$$

(nesse caso é uma solução exata mesmo, pois os seus valores são inteiros)

Agora, podemos calcular os erros de truncamento da solução $X^{(6)}$ obtida com 6 iterações em relação à solução exata S :

$$\text{Erro } X^{(6)} = |X^{(6)} - S| = \{6.58816843943144e-03, 8.45347843235089e-04, 2.48995222682158e-02\}$$

$$\text{Erro de truncamento máximo } |X^{(6)} - S| = \{ 0.0248995222682158 \}$$

O **Erro de truncamento máximo** 0.0248995222682158 é da **mesma ordem de grandeza** que o **critério de parada** baseado na máxima diferença entre duas iterações sucessivas $\text{Max } |x_i^{(k+1)} - x_i^{(k)}| = 0.0269007311283617$ e então esse critério de parada poderá ser usado como estimativa de erro de truncamento.

Alternativamente, podemos estimar uma **solução mais próxima da exata** através de soluções aproximadas pelo próprio método iterativo com algumas iterações a mais, por exemplo, usando o **dobro de iterações** ou o **dobro de precisão no critério de parada**. Assim, para calcular o erro de truncamento da solução aproximada com 6 iterações, $X^{(6)}$, podemos compará-la à solução aproximada com 12 iterações, ou com o critério de parada inicial 0.0269007311283617 ao quadrado, para ter o dobro de precisão (dobro de dígitos considerados exatos), que ambas estarão mais próximas da exata do que $X^{(6)}$:

$$X^{(6)} = \{0.9934118315605686, 1.0008453478432351, 1.0248995222682158\}$$

$$X^{(12)} = \{0.999754544313626, 0.999648218060907, 0.999924062782415\}$$

Veja que $X^{(12)}$ tem um erro menor do que $X^{(6)}$, comparado com S exato:

$$\text{Erro } X^{(12)} = |X^{(12)} - S| = \{2.45455686373797e-04, 3.51781939092577e-04, 7.59372175846984e-05\}$$

(ou seja, tem 4 dígitos significativos exatos)

Assim, os erros de truncamento de $X^{(6)}$ estimados em relação a $X^{(12)}$ são:

$$\text{Erro } X^{(6)} = |X^{(6)} - X^{(12)}| = \{0.00634271275305764, 0.00119712978232767, 0.02497545948580049\}$$

$$\text{Erro de truncamento máximo } |X^{(6)} - X^{(12)}| = \{ 0.02497545948580049 \}$$

Então, o erro máximo de $X^{(m)}$ estimado, com m iterações, por comparação ao valor exato estimado $X^{(2m)}$, no caso,

$|X^{(6)} - X^{(12)}| = 0.02497545948580049$, é da mesma ordem de grandeza do erro calculado por comparação com S exato, $|X^{(6)} - S| = 0.0248995222682158$.

Essas estimativas de erros de truncamento merecem todo cuidado, pois o critério das diferenças pode ser suficientemente pequeno em alguns casos e a solução ainda pode estar longe do valor exato, como nos processos de convergência lenta.

Exemplo 2.13: calcule a solução do sistema, a seguir, com erro máximo de truncamento da ordem de 10^{-6} , com $n_1 = 3$ e $n_2 = 4$ e fator de relaxação otimizado para esse sistema:

$$\begin{cases} 2x_i - x_{i+1} = 0.1 & \longrightarrow i = 1 \\ -x_{i-1} + 2x_i - x_{i+1} = 0.1 & \longrightarrow i = 2, \dots, n_1 - 1 \\ -x_{i-2} - x_{i-1} + 2x_i = 0.2 & \longrightarrow i = n_1, \dots, n_2 - 1 \\ -x_{i-1} + x_i = 0.3 & \longrightarrow i = n_2 \end{cases}$$

Solução:

Observe que esse sistema não tem diagonal dominante (as diagonais de cada linha são iguais à soma dos módulos de seus coeficientes vizinhos, nenhuma é maior), portanto não tem convergência garantida, logo vamos implementar um algoritmo de Gauss-Seidel com uso de fatores de relaxação, partindo da solução inicial nula e fazendo iterações até que o critério máximo das diferenças entre iterações sucessivas seja $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$. Vamos usar precisão *double*, para minimizar a influência de arredondamentos.

Obtemos as equações iterativas por isolamento das incógnitas x_i em cada equação i , ou seja, isolamos as incógnitas multiplicadas pelas diagonais principais (sistema na forma matricial):

$$\begin{cases} x_i = (0.1 + x_{i+1}) / 2 & \longrightarrow i = 1 \\ x_i = (0.1 + x_{i-1} + x_{i+1}) / 2 & \longrightarrow i = 2, \dots, n_1 - 1 \\ x_i = (0.2 + x_{i-2} + x_{i-1}) / 2 & \longrightarrow i = n_1, \dots, n_2 - 1 \\ x_i = 0.3 + x_{i-1} & \longrightarrow i = n_2 \end{cases}$$

Note que, no lado direito das equações iterativas, temos as mesmas incógnitas x calculadas no lado esquerdo das equações, portanto são sempre as incógnitas mais atualizadas disponíveis.

Sem fator de relaxação, $\lambda=1.0$, conforme a Tabela 2.10, temos:

Tabela 2.10 – Valores de solução aproximada pelo método de Gauss-Seidel com critério de parada $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$	$\text{Max} x_i^{(k+1)} - x_i^{(k)} $
0	0	0	0	0	
1	0.050000	0.075000	0.162500	0.462500	0.462500
2	0.087500	0.175000	0.231250	0.531250	0.100000
3	0.137500	0.234375	0.285938	0.585938	0.059375
4	0.167187	0.276563	0.321875	0.621875	0.042188
5	0.188281	0.305078	0.346680	0.646680	0.028516
32	2.3333e-01	3.6666e-01	4.0000e-01	7.0000e-01	9.6743e-07

Fonte: Elaboração própria

Com 32 iterações, atingimos $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$ e temos:

$$X^{(32)} = \{ 0.233331807351778, 0.366664582137096, 0.399998194744437, 0.699998194744437 \}$$

$$|x_i^{(k+1)} - x_i^{(k)}| = \{ 7.08210504629e-07, \mathbf{9.67433540499e-07}, 8.37822022592e-07, 8.37822022647e-07 \}$$

Com fator de relaxação $\lambda=1.1$, $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$ em 27 iterações.

Com fator de relaxação $\lambda=1.2$, $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$ em 21 iterações.

Com fator de relaxação $\lambda=1.3$, $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$ em 24 iterações.

Então, vemos que o fator de relaxação ótimo está em torno de $\lambda=1.2$, mas qual é o **erro máximo de truncamento** dessa solução obtida com $n=21$ iterações e $\text{Max} |x_i^{(k+1)} - x_i^{(k)}| < 10^{-6}$?

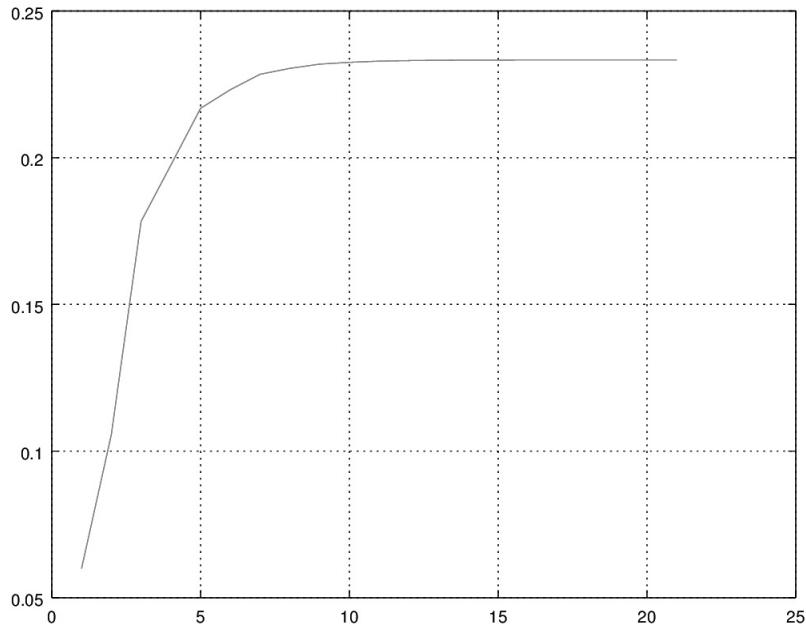
Para

$$X^{(21)} = \{ 0.233332454256926, 0.366665578771161, 0.399999140943902, 0.699999251437254 \}$$

$$|x_i^{(k+1)} - x_i^{(k)}| = \{ 7.69513875675e-07, \mathbf{9.26761935082e-07}, 7.46579149946e-07, 6.62960115027e-07 \}$$

Observe que o processo de convergência do sistema do **Exemplo 2.13**, com fator de relaxação $\lambda=1.2$, é monotônico, conforme o Gráfico 2.4, mostrando a evolução de x_1 em 21 iterações.

Gráfico 2.4 – Evolução de x_1 no eixo vertical e iterações k no eixo horizontal usando Gauss-Seidel com fator de relaxação $\lambda=1.2$



Fonte: Elaboração própria

Para calcular os erros estimados, precisaremos de uma solução mais próxima da exata, então podemos fazer iterações a mais (com o dobro de iterações, por exemplo) ou tentar atingir a solução exata S no limite da variável *double*. Nesse caso, com 61 iterações, o critério $|x_i^{(k+1)} - x_i^{(k)}| \forall i$ atinge valor zero numérico, a solução aproximada fica estacionária e é adotada como a solução exata estimada:

$$X^{(61)} = \{ 0.23333333333333, 0.36666666666667, 0.4000000000000000, 0.7000000000000000 \}$$

$$|x_i^{(k+1)} - x_i^{(k)}| = \{ 0, 0, 0, 0 \}$$

Então, o Erro máximo estimado de $X^{(21)}$ é

$$\text{Erro} \max X^{(21)} = |X^{(21)} - X^{(61)}| = 1.08789550551e-06$$

Enquanto o critério de parada é

$$\max |x_i^{(k+1)} - x_i^{(k)}| = 9.26761935082e-07$$

Novamente verificamos que o critério de parada baseado nas diferenças das soluções entre iterações sucessivas tem a mesma ordem de grandeza dos erros estimados:

$$\max |x_i^{(k+1)} - x_i^{(k)}| = 0.926761935082787e-06$$

$$\text{Erro } \max |X^{(21)} - X^{(61)}| = 1.08789550551380e-06$$

Vamos implementar um algoritmo do método de Gauss-Seidel com fator de relaxação de modo que, no lado direito das equações iterativas, temos as mesmas incógnitas x calculadas no lado esquerdo das equações, sendo sempre as incógnitas mais atualizadas disponíveis, pois, à medida que os x são calculados, já são utilizados nas próximas equações. Confira o algoritmo **Cap2Gauss_Seidel.m** no **Caderno de Algoritmos** disponível para download no link: <<http://sergiopeters.prof.ufsc.br/algoritmos-livro/>>.

Quando um sistema esparsa é gerado na forma de matriz completa, com todos os seus coeficientes nulos incluídos, é imprescindível primeiro gerar uma lista de coeficientes não nulos e aplicar os métodos iterativos operando apenas esses coeficientes não nulos.

Vimos até agora que o método de Gauss-Seidel é mais eficiente do que o de Jacobi, mas isso não é uma regra geral. O método de Jacobi se torna mais rápido no caso de processamento paralelo e/ou vetorial. No **Exemplo 2.14**, vamos apresentar um caso particular em que o método de Jacobi também é o mais rápido, mesmo em processamento sequencial normal.

Exemplo 2.14: determine a solução do sistema

$$\begin{cases} x_1 + 2x_2 - 2x_3 = 1 \\ x_1 + x_2 + x_3 = 1 \\ 2x_1 + 2x_2 + x_3 = 1 \end{cases}$$

a partir da solução inicial nula usando Jacobi, Gauss-Seidel e Gauss-Seidel com fator de sub-relaxação 0.3. Observe que esse sistema não tem convergência garantida, pois não tem diagonal dominante.

Solução:

a) Solução determinada pelo método de Jacobi:

Tabela 2.11 – Valores de solução aproximada pelo método de Jacobi

k	x_1	x_2	x_3
0	0	0	0
1	1	1	1
2	1	-1	-3
3	-3	3	1
4	-3	3	1

Fonte: Elaboração própria

Vemos que o sistema converge com três iterações.

b) Solução determinada pelo método de Gauss-Seidel:

Tabela 2.12 – Valores da solução aproximada pelo método de Gauss-Seidel

k	x_1	x_2	x_3
0	0	0	0
1	1	0	-1
2	-1	3	-3
3	-11	15	-7
4	-43	51	-15
5	-131	147	-31

Fonte: Elaboração própria

Vemos que o sistema diverge, sempre ampliando os valores.

c) Solução determinada pelo método Gauss-Seidel com fator de sub-relaxação 0.3 (a sub-relaxação é indicada porque Gauss-Seidel produziu uma sequência divergente):

Tabela 2.13 – Valores da solução aproximada pelo método de Gauss-Seidel com fator de sub-relaxação

k	x_1	x_2	x_3
0	0	0	0
1	0.3	0.21	-0.006
2	0.3804	0.33468	-0.13325
3	0.285523	0.488593	-0.25774
4	0.052064	0.703719	-0.33389
5	-0.28612	0.978607	-0.34921
12	-3.24028	3.210688	0.633227
31	-2.72791	2.792591	0.947815
70	-2.99400	2.995765	0.993771
100	-2.99938	2.99956	0.999436
189	-3	2.999999	1
190	-3	3	1

Fonte: Elaboração própria

Vemos que o sistema converge para o valor exato, mas com 190 iterações. Logo, nesse sistema sem convergência garantida, o método de Gauss-Seidel com fator de sub-relaxação 0.3 amortece as grandes variações da solução de uma iteração para outra, e também consegue obter uma sequência convergente, mas bem mais lenta do que a conseguida com o método de Jacobi (Tabela 2.11).

Considerações:

- a) Se o sistema satisfizer os critérios de convergência, mesmo que não tenha uma diagonal muito dominante, não é obrigatória a aplicação de fatores de sobre ou sub-relaxação, pois o processo iterativo já tem convergência garantida, mas é recomendável aplicá-lo para tentar acelerar o processo iterativo, conforme cada caso. Lembremos que, quanto maior for a diagonal principal, maior será a redução de erros de uma iteração para outra e mais rápida será a convergência.
- b) Se o sistema **não** tiver **convergência garantida**, recomendamos testar o efeito de fatores de sub-relaxação ou

de sobrerrelaxação, conforme o seu comportamento iterativo. O fator de relaxação pode transformar um processo iterativo divergente em convergente.

- c) A utilização de fatores de **sub-relaxação** ($0 < \lambda < 1$) pode acelerar a convergência de processos **iterativos oscilatórios** (como no **Exemplo 2.10**, quando as incógnitas x_i crescem e diminuem com certa alternância).
- d) A utilização de fatores de **sobrerrelaxação** ($1 < \lambda < 2$) pode acelerar a convergência de processos **iterativos lentos** quando a atualização total $\Delta x_i^{(k+1)} = x_i^{(k+1)} - x_i^{(k)}$ é pequena a cada iteração. Essa variação do método de Gauss-Seidel é conhecida na literatura pertinente como Sucessive Over Relaxation (SOR).
- e) As soluções aproximadas por métodos iterativos podem ser: **convergentes**, para um valor **estacionário** (quando termina o processo iterativo); **oscilatórias**; ou **divergentes**. Como vimos nos exemplos anteriores, os fatores de relaxação podem transformar um processo iterativo divergente em convergente, e um processo oscilatório em não oscilatório.
- f) A escolha adequada do fator de relaxação λ ($0 < \lambda < 2$) nos permite conduzir o processo iterativo a uma performance "ótima", atingindo a convergência com o menor número possível de iterações, mas, para descobrir esse fator "ótimo", precisamos fazer testes sucessivos (que podem ser feitos com poucas iterações usando inicialmente critério de parada maior do que o desejado).
- g) Para a solução aproximada $x_i^{(m)}$, obtida por métodos iterativos com m iterações, ou $x_i^{(\varepsilon)}$, obtida com o limite para o critério de parada ε , o erro máximo de truncamento é normalmente da mesma ordem do critério de parada baseado na máxima diferença entre as últimas iterações sucessivas, $\text{Max} |x_i^{(m)} - x_i^{(m-1)}|$. Podemos calcular o erro estimado da solução aproximada em relação a um valor de referência estimado:

- i) ou com o dobro de iterações $2m$: $x_i^{(2m)}$;
- ii) ou com limite para o critério de parada com o dobro de precisão (dobro de dígitos) ε^2 : $x_i^{(\varepsilon^2)}$.

Os valores exatos estimados nos itens (i) e (ii) têm aproximadamente a mesma grandeza. Logo, os erros de truncamentos podem ser estimados por:

$$\begin{aligned} \text{Erro}(x_i^{(m)}) &= |x_i^{(m)} - x_i^{(2m)}| \text{ ou} \\ \text{Erro}(x_i^{(\varepsilon)}) &= |x_i^{(\varepsilon)} - x_i^{(\varepsilon^2)}|, \text{ para } i = 1, 2, \dots, n. \end{aligned}$$

Esses cálculos de erros estimados precisam ser confirmados para cada tipo de sistema de equações.

2.3 Conclusões

Existem outras metodologias de resolução de sistemas lineares, mas nos propomos a apresentar, neste Capítulo, uma família de métodos que permitisse a solução de sistemas de equações lineares com poucos coeficientes nulos de médio porte, como a dos métodos eliminativos, embora estes tenham problemas de acúmulo de erros de arredondamento, especialmente os sistemas mal condicionados. E uma família que permitisse a solução de sistemas com muitos valores nulos, como a dos métodos iterativos clássicos, além de apresentarmos os algoritmos específicos para cada sistema, justamente para não operarmos esses coeficientes nulos desnecessariamente.



Complementando...

Nesta seção, vamos discutir a generalização de solução direta otimizada para sistemas esparsos através de mapeamento duplo dos índices de coeficientes não nulos.

Sistema Esparsos Genérico

Um algoritmo otimizado de Gauss operando sobre um sistema esparsos genérico, organizado ou não em faixas e armazenado na forma de matriz expandida com os coeficientes nulos originais incluídos, pode ser implementado em duas etapas.

Primeira Etapa

Na etapa de pré-processamento da matriz expandida representativa do sistema podem ser geradas e armazenadas duas listas duplas encadeadas, do tipo árvores binárias de busca (FEOFILLOFF, 2008-2009): a **primeira lista** com o **mapeamento inicial** para os índices das linhas não nulas de cada coluna; e outra para os índices das colunas não nulas de cada linha, obtidas por varredura simples de cada elemento não nulo da matriz. Esses índices das duas listas são

armazenados em árvores binárias de busca e não em vetores comuns para ter maior velocidade de acesso aos coeficientes. Essas árvores contém o valor inteiro da posição de cada coeficiente, na linha ou na coluna, conforme o caso, e os endereços dos índices vizinhos à esquerda (anterior) e à direita (posterior). Na sequência, geramos uma **segunda lista** aplicando todos os passos k da eliminação Gaussiana clássica, a cada linha i não nula abaixo de k , mas operando apenas os índices iniciais dos coeficientes não nulos das linhas e das colunas já armazenados na primeira lista, sem ainda aplicar as respectivas operações em ponto flutuante aos coeficientes reais não nulos. Nesta fase, são geradas e armazenadas as alterações no mapeamento inicial dos índices dos coeficientes não nulos, de acordo com o processo clássico de escalonamento de Gauss, ou seja, procedemos a execução de cada passo k do algoritmo gerando e armazenando uma segunda lista de linhas $i = k$ e de colunas $j = k$ que devem ser operadas pelo método de Gauss através de funções de inserção e remoção de índices encadeados na árvore de busca inicial (FEOFIL OFF, 2008-2009).

Depois dessa varredura nos passos do algoritmo, geramos e armazenamos um chamado **mapeamento instantâneo** de índices não nulos, para ser usado na aplicação de cada passo k do escalonamento da matriz de coeficientes, como se fosse uma foto dos índices dos coeficientes não nulos instantaneamente antes da aplicação do passo k do escalonamento. Essa lista dupla instantânea é armazenada na forma de duas matrizes de índices inteiros geradas a partir das árvores binárias de busca. Em cada linha k , dessas duas matrizes, são armazenadas as respectivas linhas i e colunas j de coeficientes não nulos congeladas imediatamente antes da aplicação do passo k do método de eliminação Gaussiana, de modo que contenham somente os índices das linhas e colunas que realmente precisam ser manipuladas, por conter coeficientes não nulos.

No **Caderno de Algoritmos**, confira a geração desses mapeamentos na função **fINDEX(A)** apresentada no algoritmo **GaussEsparsa.c**.

O processamento de operações de inserção e remoção de índices em árvores binárias de busca foi aplicado por ser mais rápido do que se essas operações fossem aplicadas diretamente com os vetores desses índices.

Segunda Etapa

Nesta etapa, são operadas as eliminações em ponto flutuante somente com os coeficientes não nulos, usando os índices de linhas e colunas não nulas. Portanto, a segunda etapa é independente da primeira e podemos aplicá-la quantas vezes forem necessárias, conforme a função fGaussEsparsa do mesmo algoritmo, usando os índices do mapeamento instantâneo de coeficientes não nulos pré-fixados na primeira etapa.

Na Tabela 2.14, apresentamos testes de performance no tratamento de um sistema de $N=2000$ equações, conforme o algoritmo **GaussEsparsa.c**, com largura L de faixa dupla de dispersão dos coeficientes não nulos variável entre 10 e 1000(limite), de acordo com a equação genérica a seguir:

$$\begin{cases} i = 1 & x_i + x_{i+1} = 150 \\ i = 2 \text{ até } (n/2) & x_{i-1} + 3x_i + x_{i+1} + x_{i+L} = 100 \\ i = (n/2+1) \text{ até } (n-1) & x_{i-L} + x_{i-1} + 3x_i + x_{i+1} = 200 \\ i = n & x_{i-1} + x_i = 300 \end{cases} \quad (21)$$

Em seguida, computamos o tempo para definição da indexação inicial e da indexação instantânea de cada passo k imediatamente antes da eliminação Gaussiana, o tempo para a resolução otimizada do sistema pré-mapeado e o tempo para a resolução pelo método clássico de Gauss-Seidel iterativo, otimizado por fator f de relaxação (no caso $f=1.9$), com critério de parada $\max(|\Delta x_i|) < 10^{-4}$. Observe que esse exemplo de sistema não tem convergência garantida pelo critério de convergência de Scarborough.

Tabela 2.14 – Tempos de processamento do mapeamento da matriz esparsa para solução com o método otimizado e com o Gauss-Seidel

Largura L da faixa dispersão	Indexação completa (ms^*)	Método otimizado, operando não nulos (ms)	Gauss-Seidel otimizado (ms)
$L = 0.5 \% n$	58	1	020
$L = 5 \% n$	388	12	156
$L = 10 \% n$	3509	67	337
$L = 15 \% n$	17155	213	714
$L = 20 \% n$	50507	486	1224
$L = 30 \% n$	253422	1573	2634
$L = 40 \% n$	699212	3808	4532
$L = 50 \% n$ (max)	1310824	6911	7741

Nota: $*ms = \text{mili segundo} = 10^{-3} \text{ segundos}$.

Fonte: Elaboração própria

Por esses exemplos, verificamos que sistemas com coeficientes não nulos concentrados mais próximos da diagonal principal são resolvidos mais rapidamente tanto no pré-processamento quanto nas operações aritméticas de eliminação, como é conhecido da literatura.

Nessa otimização, o mapeamento instantâneo gera valores de índices não nulos tanto mais espalhados pela matriz quanto maior for a largura inicial da faixa não nula. Sistemas com largura de dispersão L maiores geram mais operações de inserção, ou seja, preenchem mais a matriz esparsa, consequentemente geram mais operações aritméticas. Já sistemas com largura de faixa L a partir de 300 (15 % n), para os dois lados da diagonal, exigem tempo de processamento (17155 ms) superior ao método de Gauss completo (14415 ms), mas na segunda fase de eliminação tornam-se muito mais rápidos (usam apenas 213 ms de processamento). Verificamos também, nos exemplos testados, que o algoritmo otimizado proposto é sempre mais rápido do que métodos iterativos clássicos, cujas soluções são aproximadas com critério de parada relativamente grosso 10⁻⁴, enquanto as soluções obtidas pelo algoritmo apresentado são exatas, exceto pelos arredondamentos acumulados. Por ser um método

direto, ainda independe de critérios de convergência ou da classe de sistema envolvido.

Acesse o **Caderno de Exercícios e Respostas**, disponível no *link* <<http://sergiopeters.prof.ufsc.br/exercicios-e-respostas/>>, para realizar os exercícios propostos ao reforço do seu aprendizado.