UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE MATEMÁTICA

LUCAS CARVALHO SABINO

MÉTODOS DIRETOS E ITERATIVOS PARA SOLUÇÃO DE SISTEMAS LINEARES.

BELÉM-PA
2023

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE MATEMÁTICA

LUCAS CARVALHO SABINO

MÉTODOS DIRETOS E ITERATIVOS PARA SOLUÇÃO DE SISTEMAS LINEARES.

Trabalho de conclusão de curso apresentado para obtenção do título de Licenciado em Matemática da Universidade Federal do Pará.

Orientador: Prof. Dr. Manoel Silvino Batalha de Araujo

BELÉM-PA
2023

LUCAS CARVALHO SABINO

MÉTODOS DIRETOS E ITERATIVOS PARA SOLUÇÃO DE SISTEMAS LINEARES.

Trabalho de conclusão de curso apresentado para obtenção do título de Licenciado em Matemática da Universidade Federal do Pará.

Data da Defesa: 19 de dezembro de 2023
Conceito: _____

Banca Examinadora

Prof. Dr. Manoel Silvino Batalha de Araujo
Faculdade de Matemática - UFPA, campus Belém
Orientador

Prof. Dr. Anderson David de Souza Campelo
Faculdade de Matemática - UFPA, campus Belém
Membro da Banca

Prof. Dr. Samuel Levi Freitas da Luz
Faculdade de Matemática - UFPA, campus Castanhal
Membro da Banca

BELÉM-PA
2023

Este trabalho é dedicado aos meus pais, Edna do Socorro e José Sabino, que sempre acreditaram no poder da educação e sonharam com a minha formação e sucesso.

AGRADECIMENTOS

Agradeço imensamente a toda minha família pelo constante companheirismo e incentivo ao longo dos anos. Em especial, expresso minha profunda gratidão à minha mãe, Edna do Socorro, e ao meu pai, José Sabino, pela imensurável dose de amor e carinho. Agradeço também a minha irmã Catarina Sabino e ao meu primo André Heron, que apesar da distância sempre me motivaram e acreditaram na concretização deste projeto.

Agradeço a minha namorada, Julia do Lago Nascimento, por acreditar que eu seria capaz de concluir essa jornada quando eu mais quis desistir. Este trabalho não teria sido possível sem o seu amor e constante companheirismo. Obrigado por estar ao meu lado nos bons e maus momentos ao longo dos últimos 4 anos.

Expresso minha gratidão a todos os amigos que fiz durante a graduação, com destaque para o Maiko Alves, Jefferson Pedro, André Henrique, Giliardo Rodrigues e Jucivani Cavalcante. Nossos momentos de estudos e conversas foram fundamentais nesta jornada, onde compartilhamos momentos de tristeza e felicidade.

Agradeço sinceramente a todos os professores que contribuíram para minha formação ao longo da graduação, com um agradecimento especial ao meu orientador, Prof. Dr. Manoel Silvino Batalha de Araujo. Agradeço por aceitar-me como seu orientando e por sua paciência e apoio inestimáveis na realização deste trabalho.

"Nós só podemos ver um pouco do futuro, mas
o suficiente para perceber que há muito a fazer."
(Alan Turing)

RESUMO

Este trabalho tem como objetivo central realizar um estudo sobre alguns dos principais métodos para a solução de sistemas lineares, dada a frequência significativa com que esses sistemas surgem em diversas áreas, tais como engenharia, estatística, biologia, finanças, economia, entre outras. Entre os métodos existentes, nossa atenção estará voltada para duas classes amplamente reconhecidas: os métodos diretos, com destaque para a eliminação de Gauss e a fatoração LU, e os métodos iterativos, com ênfase nos métodos de Jacobi e Gauss-Seidel.

Considerando a complexidade de sistemas lineares que podem envolver centenas ou milhares de equações, a abordagem computacional se mostra fundamental para encontrar soluções exatas ou aproximadas. Nesse contexto, faremos uso do software GNU Octave como a principal ferramenta para desenvolver algoritmos capazes de calcular, de maneira direta ou iterativa, as soluções desejadas.

Ao final deste estudo, exploraremos a aplicação prática desses métodos, destacando suas vantagens, limitações e a eficiência de cada método visto através do GNU Octave. A intenção é fornecer uma compreensão aprofundada dessas técnicas, visando contribuir para a resolução eficiente de sistemas lineares em contextos diversos.

Palavras-chave: Métodos diretos. Métodos iterativos. GNU Octave.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de equações lineares são essenciais em diversas áreas da ciência e da engenharia, modelando e descrevendo fenômenos complexos presentes na nossa realidade. Resolver esses sistemas é um desafio crucial e, ao longo dos anos, diversos métodos têm sido desenvolvidos para obter soluções precisas e eficientes, como por exemplo a **eliminação de Gauss**, **fatoração LU**, **método iterativo de jacobi**, **Gauss-Seidel**, dentre outros.

O avanço tecnológico nos proporcionou uma alta capacidade de processar dados e realizar operações em frações de segundos, porém, antes de visualizarmos os resultados obtidos pelos métodos computacionais obtidos através do software GNU Octave, precisaremos entender a fundo o que de fato são os sistemas lineares e entender como funcionam os principais métodos de solução de sistemas lineares. Uma vez em que entendermos como tais métodos funcionam, teremos total capacidade de aplicar algoritmos poderosos o suficientes para solucionar os mais variados problemas envolvendo sistemas lineares.

Faremos um breve resumo do que são os sistemas lineares e suas operações de linhas, após isso, daremos destaques nos principais métodos diretos e iterativos para solução destes sistemas, finalizando com a aplicação direta com o software GNU Octave, utilizando tudo que foi visto para escolher o método de acordo com o objetivo em cada contexto.

2 SISTEMA DE EQUAÇÕES LINEARES E SUAS OPERAÇÕES ELEMENTARES

Introduziremos agora noções básicas sobre o que são **sistemas de equações lineares** e as operações elementares que podemos fazer sobre eles, e então, utilizando desses conhecimentos teremos poder suficiente para aprender sobre a **eliminação de Gauss** e a **fatoração LU**, métodos **diretos** que se utilizam de passos finitos para encontrar a solução de sistemas lineares, a menos de erros de aproximação.

Um sistema de equações lineares é um conjunto de equações lineares, ou seja, equações nas quais cada termo é uma constante ou uma variável multiplicada por uma constante, e em que as variáveis estão elevadas à primeira potência. Um sistema de equações lineares é composto por duas ou mais equações lineares que envolvem as mesmas variáveis.

Sistemas lineares são frequentemente vistos em problemas de áreas como as de engenharia elétrica, química e economia, sendo possível existir sistemas com centenas ou milhares de equações e variáveis.

Um sistema desse tipo tem a forma:

$$\begin{aligned} E_1 &: a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n = b_1 \\ E_2 &: a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n = b_2 \\ &\vdots \\ E_n &: a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n = b_n \end{aligned} \tag{2.1}$$

Neste contexto, as equações são representadas por E_i, onde $i = 1,2,...,n$. As constantes associadas são denotadas por a_{ij} para $i, j = 1,2,...,n$, enquanto os termos constantes de E_i são expressos como b_i para $i = 1,2,...,n$. As incógnitas a serem determinadas são $x_1,...,x_n$.

Dessa forma, os coeficientes da matriz $A = [a_{ij}]$ podem ser representados de maneira matricial. Aqui, x é o vetor coluna das incógnitas $x_1,...,x_n$, e $b = (b_1,b_2,...,b_n)$ é o vetor coluna que representa o termo independente das equações.

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b} \tag{2.2}$$

2.1 Operações elementares em linhas

Dado o sistema de equações 2.1, podemos realizar três tipos de operações afim de simplificar o sistema:

- Multiplicação de uma equação por uma constante não nula, substituindo a equação original pela resultante da multiplicação, ou seja $(\lambda E_i) \rightarrow E_i$.
- A equação E_j pode ser multiplicada por uma constante λ não nula e adicionada à equação E_i, substituindo a equação original pela resultante da operação, ou seja $(E_i + \lambda E_j) \rightarrow (E_i)$
- As equações podem trocar de posição, denotamos isso por $(E_i) \leftrightarrow (E_j)$

Essas operações elementares são uma das principais ferramentas usadas em métodos Diretos, buscando simplificar o sistema original, transformando-o em um sistema equivalente mais fácil de resolver como veremos a seguir.

2.2 Métodos Diretos para solução de sistemas lineares

Os métodos diretos são uma classe de técnicas utilizadas para resolver sistemas de equações lineares de forma exata, ou seja, encontrar a solução exata do sistema. No entanto, é importante ressaltar que, em cálculos numéricos, as soluções obtidas através desses métodos podem não ser exatas, mas aproximadas com uma certa margem de erro devido a fatores como arredondamentos de números, truncamentos ou limitações dos métodos utilizados. Esses erros de aproximação são inevitáveis, uma vez que as soluções numéricas são representadas em formato de ponto flutuante, com precisão limitada, o que pode afetar a exatidão da solução obtida.

Um exemplo de método direto para a solução de sistemas lineares é a **eliminação de Gauss**, que consiste em uma série de operações elementares nas equações do sistema a fim de reduzi-lo a um sistema equivalente na **forma triangular superior**, determinando então as incógnitas através do **processo de substituição regressiva**.

Apesar de fornecerem resultados precisos em termos de cálculos matemáticos, é importante estar ciente de que as soluções encontradas através de métodos diretos são aproximações da solução real a menos de erros de aproximação.

2.3 Eliminação de Gauss

A eliminação Gaussiana é utilizada para resolver sistemas lineares finitos de tamanho $n \times n$ (n linhas e n colunas), transformando o sistema em uma matriz aumentada de ordem

$n \times n+1$. Desta forma, um sistema linear como vimos em 2.1 em uma matriz aumentada:

$$E_{m\times n} = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right] = [A,b] \tag{2.3}$$

Podemos também representar a matriz da seguinte forma:

$$E_{m\times n} = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{array}\right] = [A,b] \tag{2.4}$$

onde os elementos na (n+1)-ésima coluna são os elementos de b.

Para resolver o sistema, realizamos os seguintes passos da eliminação gauss com substituição regressiva:

1. Obter matriz aumentada da forma $E = [A,b]$.
2. Transformar a matriz aumentada E em uma **matriz triangular superior**.
3. Resolver o sistema linear obtido no *Passo 2* por substituição regressiva.

ILUSTRAÇÃO: Após obter a matriz aumentada $E = [A,b]$ como vimos na equação 2.4. Devemos então escolher o primeiro elemento não nulo da primeira coluna, ou seja, o **pivô**. É importante que o pivô seja não nulo, ou seja, que $a_{ii} \neq 0$. Caso o pivô seja nulo, devemos realizar a troca de linhas visto em 2.1 afim de obter um pivô não nulo. Após identificar o **pivô** da primeira coluna, realizamos a seguinte operação:

$$(E_j - (\tfrac{a_{j1}}{a_{11}})E_1) \rightarrow (E_j) \text{ para } j = i+1, i+2, ..., n$$

Desde que $a_{11} \neq 0$ o processo irá eliminar (mudar o coeficiente para 0) x_1 na equação abaixo do elemento pivô escolhido. De modo geral para qualquer a_{ij}, para $i = 1,2...n-1$ e $j = i+1, i+2, ..., n$ afim de eliminar todos os elementos abaixo do pivô escolhido após cada eliminação de x_i, a matriz obtida então tem a forma:

$$[A,b]^{(n-1)} = E^{(n-1)} = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \vdots & a_{3n}^{(3)} & a_{3,n+1}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} & a_{n,n+1}^{(n-1)} \end{array}\right] \tag{2.5}$$

Onde a matriz $E^{(n-1)}$ para $n = 1,2,...$ representa um sistema linear com o mesmo conjunto solução do sistema original.

Após a eliminação dos coeficientes abaixo de cada pivô escolhido, a matriz A possui a forma de uma **matriz triangular superior**. Uma **matriz triangular superior** é uma matriz necessariamente quadrada, onde todos os elementos abaixo da diagonal principal são nulos, e portando, os elementos acima da diagonal são diferentes de zero. Agora que temos uma matriz triangular superior, podemos enxergar também o sistema de equações como um sistema triangular, ou seja:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= a_{1,n+1} \\ a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= a_{2,n+1} \\ a_{33}x_3 + \cdots + a_{3n}x_n &= a_{3,n+1} \\ \vdots \qquad & \quad \vdots \\ a_{nn}x_n &= a_{n,n+1} \end{aligned} \tag{2.6}$$

Com isso, podemos facilmente obter o valor de x_n na última equação isolando-o, desta forma temos que

$$x_n = \frac{a_{n,n+1}}{a_{nn}}$$

Com isso obtemos o valor de x_n na n-ésima equação, e agora podemos substituí-lo na seguinte, ou seja, na $(n-1)$-ésima equação para obter x_{n-1}, teremos então

$$x_{n-1} = \frac{a_{n-1,n+1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

Esse processo é chamado de **substituição regressiva**, e consiste em resolver as equações da última até a primeira equação de um sistema **triangular superior**. Como já conhecemos os valores de x_n e x_{n-1}, continuamos o processo de substituição para todo x_i, ou seja

$$x_i = \frac{a_{i,n-1} - a_{i,i+1} - \cdots - a_{i,i-1}x_{n-i} - a_{i,n}x_n}{a_{ii}} = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$$

para cada $i = n-1, n-2, ..., 2, 1$

Exemplo 2.3.1. Utilize a **eliminação de Gauss** no sistema linear $A_{3\times 3}$ abaixo:

$$\begin{cases} 2x_1 + 3x_2 - x_3 = 5 & (E_1) \\ 4x_1 + 4x_2 - 3x_3 = 3 & (E_2) \\ 2x_1 - 3x_2 + x_3 = -1 & (E_3) \end{cases}$$

Solução.

Passo 1: Escrever a matriz aumentada do sistema:

$$E_{3\times4} = \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 1 & -1 \end{array}\right] = [A,b]^0$$

Passo 2: Deseja-se zerar os elementos abaixo da diagonal principal, para isso escolhendo $a_{11}^{(0)}$ como pivô, calculamos os multiplicadores:
$m_{21}^{(0)} = \frac{a_{21}^{(0)}}{a_{11}^{(0)}} = \frac{4}{2}$
$m_{31}^{(0)} = \frac{a_{31}^{(0)}}{a_{11}^{(0)}} = \frac{2}{2}$
Desta Forma, temos as seguintes operações:

$$E_2^{(1)} \leftarrow E_2 - m_{21}^{(0)} \times E_1$$
$$E_3^{(1)} \leftarrow E_3 - m_{31}^{(0)} \times E_1$$

Desta forma, todos os valores abaixo do pivô a_{11} serão eliminados, obtendo-se:

$$[A,b]^{(1)} = \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & 2 & 1 & 7 \\ 0 & 6 & -2 & 6 \end{array}\right]$$

Agora devemos zerar os elementos abaixo da diagonal principal na segunda coluna, sendo o elemento a_{22} o pivô da segunda linha e $\neq 0$ escolhemos ele como o multiplicador para zerar a_{32}, então: $m_{32}^{(1)} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{6}{2} = 3$
Agora que definimos o multiplicador, faremos as seguintes transformações:

$$E_3^{(2)} \leftarrow E_3^{(1)} - m_{32}^{(1)} \times E_2^{(1)}$$

Obtendo:

$$[A,b]^{(2)} = \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & 2 & 1 & 7 \\ 0 & 0 & 5 & 15 \end{array}\right]$$

Desta forma, a matriz $[A,b]^2$ é uma matriz triangular superior equivalente a $[A,b]^{(0)}$.

Passo 3: Agora podemos realizar a substituição regressiva, para isso, representaremos a matriz em seu novo sistema triangular:

$$2x_2 + 3x_2 - x_3 = 5$$
$$2x_2 + x_3 = 7$$
$$5x_3 = 15$$

Então, realizando a substituição regressiva, temos:

$$x_3 = \frac{15}{5} = 3$$
$$x_2 = \frac{(7-3)}{2} = 2$$
$$x_1 = \frac{5-(3\times 2)-((-1)\times 3)}{2} = 1$$

Desta forma, obtemos o vetor solução $\bar{x} = [1,2,3]$ do sistema $[A,b]^{(2)}$, que é equivalente ao sistema inicial $[A,b]^{(0)}$.

A eliminação de Gauss é uma técnica fundamental para resolver sistemas lineares de equações de forma direta. No entanto, ela não se limita apenas a essa aplicação específica. Na próxima seção, exploraremos uma forma alternativa de utilizar os mesmos passos da eliminação de Gauss para fatorar uma matriz. Essa fatoração é particularmente útil quando a matriz pode ser escrita na forma A = LU, onde L é uma matriz triangular inferior e U é uma matriz triangular superior.

2.4 Fatoração LU

A fatoração LU é uma poderosa ferramenta que descompõe a matriz original em duas matrizes triangulares, permitindo simplificar e resolver problemas de forma mais eficiente. Embora nem todas as matrizes possuam essa representação específica, muitas das que encontramos com frequência no estudo de técnicas numéricas apresentam essa estrutura.

Em suma, a eliminação de Gauss, além de ser um método direto para resolver sistemas lineares, nos proporciona uma abordagem para fatorar matrizes na forma A = LU, ampliando a aplicabilidade dessa técnica em diversos contextos numéricos.

Para entendermos melhor como funciona a fatoração LU, utilizaremos dos conceitos básicos da Eliminação de Gauss já vistos anteriormente, para isso, suponha um sistema $Ax = b$, onde os termos $a_{ii}^{(i)}$ de A, para cada $i = 1,2,...,n$ são diferentes de zero.

O primeiro passo consiste em definir os multiplicadores m que transformarão o sistema em um outro, onde os elementos abaixo da diagonal principal são nulos, ou seja, para cada $j = 2,3,...,n$, temos as operações

$$(E_j - m_{j,1}E_1) \to (E_j), \text{ onde } m_{j,1} = (\frac{a_{j1}^{(0)}}{a_{11}^{(0)}})$$

Essas operações correspondem pela multiplicação à esquerda de A pela matriz

$$M^{(0)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & \cdots & 1 \end{bmatrix} \tag{2.7}$$

Chamamos $M^{(0)}$ de a **primeira matriz de transformação de Gauss.** Ao realizar $M^{(0)}A^{(0)}$, obtemos

$$M^{(0)}A^{(0)} = \begin{bmatrix} 1 & a_{12} & \cdots & 0 \\ -m_{21}^{(0)} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1}^{(0)} & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \cdots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \cdots & a_{nn}^{(0)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ a_{21}^{(0)} - m_{21}^{(0)}a_{11}^{(0)} & a_{22}^{(0)} & \cdots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} - m_{n1}^{(0)}a_{11}^{(0)} & a_{n2}^{(0)} & \cdots & a_{nn}^{(0)} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(0)} & \cdots & a_{nn}^{(1)} \end{bmatrix} = A^{(1)}$$

Observamos que após efetuar tais operações, obtemos o mesmo resultado do *passo 2* visto na Eliminação de Gauss, então temos

$$A^{(1)}x = M^{(0)}A^{(0)}x = M^{(0)}b = b^{(1)}$$

Seguimos com o processo da Eliminação de Gauss para construir então $M^{(1)}$, com os elementos abaixo da diagonal na segunda coluna substituidos pelos opostos dos multiplicadores, sendo $M^{(1)}$ da forma

$$M^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & -m_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -m_{n2} & \cdots & 1 \end{bmatrix} \tag{2.8}$$

$$m_{j,2} = \frac{a_{j2}^{(1)}}{a_{22}^{(1)}}$$

Realizando $M^{(1)}A^{(1)}$, temos então

$$A^{(2)}x = M^{(1)}A^{(1)}x = M^{(1)}M^{(0)}b = b^{(2)}$$

Para obtermos então $A^{(k+1)}x = b^{(k+1)}$ com $A^{(k)}x = b^{(k)}$ já formado, multiplicamos a **k-ésima matriz de transformação de Gauss** $M^{(k)}$ por $A^{(k)}$, desta forma

$$A^{(k+1)}x = M^{(k)}A^{(k)}x = M^{(k)}...M^{(0)}Ax = M^{(k)}b^{(k)} = b^{(k+1)} = M^{(k)}...M^{(0)} \tag{2.9}$$

O processo se encerra com a formação do sistema $A^{(n)}x = b^{(n)}$, em que $A^{(n)}$ é a matriz triangular superior

$$A^{(n)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \vdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix}$$

onde

$$A^{(n)} = M^{(n-1)}M^{(n-2)}...M^{(0)}A \tag{2.10}$$

Dessa forma, obtemos o elemento U da decomposição $A = LU$. Para calcular a matriz triangular inferior L, é necessário revisitar a multiplicação da equação $A^{(k)}x = b^{(k)}$ pela transformação de Gauss usando a matriz $M^{(k)}$, que foi empregada para derivar a equação (2.10):

$$A^{(k+1)}x = M^{(k)}A^{(k)}x = M^{(k)}b^{(k)} = b^{(k+1)}$$

Em que $M^{(k)}$ gera as operações nas linhas

$$(E_j - M_{(j,k)}E_k) \to (E_j) \qquad \text{para } \; j = k+1, ..., n.$$

Para desfazer os efeitos dessa transformação e restaurar $A^{(k)}$, é requerido que as operações $(E_j + m_{j,k}E_k) \to (E_j)$ sejam executadas para cada $j = k+1, ..., n$. Essa abordagem é análoga a multiplicar pela matriz inversa de $M^{(k)}$, a qual assume a forma:

$$K^{(k)} = [M^{(k)}]^{(-1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & m_{k+1,k} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & m_{n,k} & \cdots & \cdots & 1 \end{bmatrix}$$

Podemos determinar então a matriz triangular inferior L na fatoração de A como sendo o produto das matrizes $L^{(k)}$:

$$L = L^{(0)}L^{(1)}...L^{(n-1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & \cdots & 0 \\ m_{31} & m_{32} & 1 & \vdots & \vdots & 0 \\ \vdots & \vdots & \cdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & 1 & m_{n-1,n} \\ m_{n1} & m_{n2} & \cdots & \cdots & m_{n,n-1} & 1 \end{bmatrix}$$

Já que o produto de L com a matriz triangular superior $U = M^{(n-1)}...M^{(2)}M^{(1)}A$ dá

$$\begin{aligned} LU &= L^{(0)}L^{(2)}...L^{(n-2)}L^{(n-1)} \times M^{(n-1)}M^{(n-2)}...M^{(1)}M^{(0)}A = \\ &= [M^{(1)}]^{-1}[M^{(2)}]^{-1}...[M^{(n-1)}]^{-1}[M^{(0)}]^{-1} \times M^{(n-1)}M^{(n-2)}...M^{(1)}M^{(0)}A = A \end{aligned}$$

Teorema 2.1. Se a eliminação de Gauss puder ser realizada no sistema linear $Ax = b$ sem trocas de linhas, então a matriz A pode ser fatorada no produto de uma matriz triangular inferior de L e uma matriz triangular superior U, $A = LU$ em que $m_{(ij)} = \frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$

Exemplo 2.4.1. Determine a fatoração LU da matriz A no sistema linear $Ax = b$ em que

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 1 & 1 & 2 \\ 4 & 3 & 2 \end{bmatrix} \qquad \text{e} \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Solução. Usando o processo de Gauss para triangularizar A, temos:
Pivô = $a_{11}^{(0)} = 3$

$$m_{21} = \frac{a_{21}^{(0)}}{a_{11}^{(0)}} = \frac{1}{3} \quad \text{e} \quad m_{31} = \frac{a_{31}^{(0)}}{a_{11}^{(0)}} = \frac{4}{3}$$

Realizando as operações para zerar a_{21} e a_{31}

$$(E_2 - m_{21}E_1) \to (E_2)$$

$$(E_3 - m_{31}E_1) \to (E_3) \quad \text{então} \quad A^{(1)} = \begin{bmatrix} 3 & 2 & 4 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{1}{3} & \frac{-10}{3} \end{bmatrix}$$

Agora para zerar a_{32}
Pivô = $a_{22}^{(1)} = \frac{1}{3}$

$$m_{32} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{\frac{1}{3}}{\frac{1}{3}} = 1$$

$$(E_3 - m_{32}E_2) \to (E_3) \quad \text{então} \quad A^{(2)} = \begin{bmatrix} 3 & 2 & 4 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & -4 \end{bmatrix}$$

Os fatores L e U são

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 1 & 1 & 2 \\ 4 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{4}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 2 & 4 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & -4 \end{bmatrix} = LU$$

2.4.1 Resolução dos sistema linear $Ax = b$ pela fatoração LU

Dado um sistema de equações lineares representado por $Ax = b$ e considerando a decomposição em fatores LU da matriz A, temos o seguinte cenário:

O sistema de equações $Ax = b$ é equivalente à expressão $(LU)x = b$. Introduzindo uma nova variável y para representar o produto da matriz U com o vetor x, podemos observar que a solução para o sistema de equações lineares pode ser obtida resolvendo uma série de sistemas lineares triangulares:

i) Primeiro, resolvemos um sistema linear triangular inferior $Ly = b$ para encontrar o valor de y.

ii) Em seguida, usando o valor de y, resolvemos outro sistema linear triangular superior $Ux = y$ para encontrar a solução final x.

Isso nos permite resolver o sistema original $Ax = b$ de forma mais eficiente ao dividir o processo em etapas mais simples, aproveitando a fatoração LU da matriz A para encontrar a solução de maneira mais direta.

Vejamos de forma prática como funciona utilizando o exemplo 2.4.1, onde já encontramos as matrizes L e U

Seja

$$Ax = LUx = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{4}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 2 & 4 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Introduzimos a substituição $y = Ux$. Então, $b = L(Ux) = Ly$, então

$$Ly = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{4}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

temos que

$$\begin{aligned} y_1 &= 1 \\ \frac{1}{3}y_1 + y_2 &= 2 \qquad \rightarrow y_2 = 2 - \frac{1}{3} = \frac{5}{3} \\ \frac{4}{3}y_1 + y_2 + y_3 &= 3 \qquad \rightarrow y_3 = 3 - \frac{5}{3} - \frac{4}{3} = 0 \end{aligned}$$

Agora resolvendo para $Ux = y$ para determinar x do sistema original:

$$y = Ux = \begin{bmatrix} 3 & 2 & 4 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{5}{3} \\ 0 \end{bmatrix}$$

Utilizando a substituição regressiva, obtemos os valores $x_3 = 0$, $x_2 = 5$ e $x_1 = -3$

A eliminação de Gauss e a fatoração LU são bons métodos para resolver sistemas lineares pequenos e médios (com até algumas centenas de operações) e quando uma solução precisa é necessária, porém, em algumas áreas como física e engenharia elétrica podem surgir sistemas de equações lineares grandes (com milhares e até milhões de equações), e que muitas vezes não requerem uma precisão tão exata da solução, nesse caso o método da eliminação de gauss pode não ser a melhor opção, sendo então necessário recorrer aos métodos iterativos como os de Jacobi e Gauss-Seidel.

3 MÉTODOS ITERATIVOS

Métodos iterativos são técnicas numéricas utilizadas para obter uma solução aproximada para um problema matemático. São muito convenientes para sistemas grandes e esparsos, ou seja, sistemas com grande quantidade de zeros (A definição sobre esparsidade pode ser encontrada na referência [4], Capítulo 3), e que não requerem uma precisão exata da solução. Para problemas como estes, utilizamos técnicas iterativas como as de Jacobi e Gauss-Seidel, que geram uma sequência de soluções aproximadas que melhoram conforme as iterações são executadas. Em geral, nos métodos iterativos que veremos a seguir, faremos uma aproximação do conjunto de soluções $\bar{x}$ de um sistema linear na forma $Ax = b$, que consiste em uma série de aproximações, sendo dado uma aproximação inicial $x^{(0)}$. Além disso, precisamos sempre ter em mente os critérios de parada dos métodos e se a sequência que estamos obtendo está convergindo ou não para a solução desejada.

Resolver um sistema linear por um método iterativo envolve a criação de uma sequência de vetores que gradualmente se aproxima da solução exata para **x**. Em outras palavras, à medida que a iteração avança, a sequência de vetores $x^{(k)}$ converge para o valor real de x, o que significa que:

$$\lim_{x\to\infty} x^{(k)} = x \tag{3.1}$$

para qualquer vetor inicial dado por $x^{(0)}$. Geralmente, os métodos iterativos são construídos como uma iteração de ponto fixo. No caso de um sistema linear, reescreve-se a equação matricial em um problema de ponto fixo, ou seja:

$$x = Tx + c \tag{3.2}$$

Onde T é a matriz de iteração, e c é chamado de vetor de iteração. Após construídas T e c, o método iterativo consiste em computar a iteração

$$x^{(k+1)} = Tx^{(k)} + c, \quad k \geq 0 \tag{3.3}$$

Se $e^{(k)} = x - x^{(k)}$ é o erro da iteração k. Subtraindo 3.3 de 3.2, temos:

$$e^{(k+1)} = Te^{(k)} \tag{3.4}$$

Chamamos T de **matriz de iteração**. Se T for simétrica e definida positiva, temos que da álgebra linear, vale o resultado

$$||e^{(k+1)}|| = ||Te^{(k)}|| \leq \rho(T)||e^{(k)}||, \quad \forall k \geq 0 \tag{3.5}$$

Aqui, $\rho(T)$ representa o que chamamos de *raio espectral* de uma matriz T, ou seja, o módulo máximo dos autovalores associados a T. Iterando regressivamente 3.5, temos

$$||e^{(k)}|| \leq [\rho(T)]^k ||e^{(0)}||, \quad k \geq 0 \tag{3.6}$$

Dessa forma, $e^{(k)} \to 0$ quando $k \to \infty$ para todo $e^{(0)}$, desde que $\rho(T) < 1$. Deste resultado, podemos obter mais condições necessárias para a convergência dos métodos iterativos.

Lema 3.1. Se $\rho(T) < 1$, então existe $(I-T)^{-1}$ e:

$$(I-T)^{-1} = \sum_{k=0}^{\infty} T^k \tag{3.7}$$

Demonstração. Seja λ um autovalor de T e x um autovetor associado, isto é, $Tx = \lambda x$. Então $(I-T)x = (1-\lambda)x$. Além disso, temos $|\lambda| < \rho(T) < 1$, logo $(1-\lambda) \neq 0$, o que garante que $(I-T)$ é não singular. Agora, seja $S_m = I + T + T^2 + ... + T^m$. Então

$$(I-T)S_m = (1 + T + T^2 + ... + T^m) - (T + T^2 + ... + T^{m+1}) = I - T^{m+1}$$

como T é convergente(veja sobre matrizes convergentes na referência [1] capítulo 7), temos que

$$\lim_{m\to\infty} (I-T)S_m = \lim_{m\to\infty} (I - T^{m+1}) = I$$

$$\text{Assim, } (I-T)^{-1} = \lim_{m\to\infty} S_m = I + T + T^2 + ... = \sum_{k=0}^{\infty} T^k$$

□

Teorema 3.1. Para qualquer aproximação inicial $x^{(0)}$, a sequência recursiva $\{x^{(k)}\}_{k\in\mathbb{N}}$ dado por:

$$x^{k+1} = Tx^k + c \tag{3.8}$$

converge para solução única de $x = Tx + c$ se, e somente, se $\rho(T) < 1$.

Demonstração. Primeiro, suponha que $\rho(T) < 1$, Então:

$$\begin{aligned} x^{k+1} &= Tx^{(k)} + c \\ &= T(Tx^{(k-1)} + c) + c \\ &= T^2 x^{(k-1)} + (T+I)c \\ &\vdots \\ &= T^k x^{(0)} + (T^{k-1} + ... + T + I)c \end{aligned} \tag{3.9}$$

Como $\rho(T) < 1$, o que implica na convergência de T, e

$$\lim_{k\to\infty} T^k x^{(0)} = 0 \tag{3.10}$$

pelo lema 3.1 temos que

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} T^k x^{(0)} + (\sum_{j=0}^{\infty} T^k)c = 0 + (I-T)^{-1}c = (I-T)^{-1}c \tag{3.11}$$

Dessa forma, sequência $\{x^{(k)}\}$ converge para o vetor $x = (I-T)^{-1}c$ e $x = Tx+c$. Para demonstrar a recíproca, mostramos que, para qualquer $z \in \mathbb{R}^n$ temos que $\lim_{k\to\infty} T^k z = 0$. De acordo com as definições de convergência de matriz (consulte a referência...) isso é equivalente a $\rho(T) < 1$. Seja z um vetor arbitrário e x a solução única de $x = Tx+c$. Defina $x^{(0)} = x - z$, e, para $k \geq 1$, $x^{(k-1)} + c$. Então $\{x^{(k)}\}$ converge para x. Temos que,

$$x - x^{(k+1)} = (Tx+c) - (Tx^{(k)} = T(x - x^{(k)})$$

dessa forma,

$$\begin{aligned} x - x^{k+1} =& T(x - x^{(k)}) \\ =& T^2(x - x^{(k-1)}) \\ & \vdots \\ =& T^k(x - x^{(0)}) = T^k z \end{aligned}$$

Logo, $\lim_{k\to\infty} T^k z = \lim_{k\to\infty} T^k(x - x^{(0)}) = \lim_{k\to\infty}(x - x^{(k)} = 0$ Como z era arbitrário, então T é uma matriz convergente e $\rho(T) < 1$ □

3.1 Método iterativo de Jacobi

O método de Jacobi divide o sistema de equações lineares em n-equações, uma para cada variável desconhecida, e então usa as estimativas mais recentes de todas as outras variáveis para calcular a estimativa atual de cada variável, ou seja, dado o sistema $Ax = b$:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n = b_n \end{cases} \tag{3.12}$$

Isolando x_1 na primeira equação, x_2, e assim sucessivamente, temos:

$$\begin{cases} x_1 = \frac{b_1 - (a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)}{a_{11}} \\ x_2 = \frac{b_2 - (a_{21}x_1 - a_{23}x_3 + \cdots - a_{2n}x_n)}{a_{22}} \\ \vdots \\ x_n = \frac{b_n - (a_{n1}x_1 - a_{n2}x_2 + \cdots - a_{n,n-1}x_{n-1})}{a_{nn}} \end{cases} \tag{3.13}$$

De modo geral, para $a_{ii} \neq 0$, $i = 1, 2, \ldots n$

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{j=1, j\neq 0}^{n-1} a_{nj}x_j) \quad \text{para } k \geq 0 \tag{3.14}$$

No método de Jacobi, realizamos uma aproximação inicial conhecida como $x^{(0)}$, para então encontrar uma nova aproximação para $x^{(1)}, x^{(2)}, ..., x^{(n)}$, ou seja:

$$x_1^{(1)} = \frac{1}{a_{11}}(b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(0)})$$

Repetindo a ideia para $x_2^{(1)}, ..., x_n^{(1)}$

$$x_i^{(1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(0)}) \;\; i = 1, 2, ..., n \tag{3.15}$$

Com isso, obtemos todos os $x^{(1)} = [x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}]$, utilizamos novamente a equação 3.13, para encontrar as novas aproximações $x^{(2)} = [x_1^{(2)}, x_2^{(2)}, ..., x_n^{(2)}]$. Resumidamente, o método de Jacobi consiste em todos os componentes dos vetores de $\bar{x} = [x^{(1)}, x^{(2)} ..., x^{(n)}]$ utilizando a equação 3.14. O método será bem sucedido se a sequência de vetores sucessivamente convergir para solução do problema, ou seja, se $\max_{1\leq i\leq n} |x_i^{(k+1)} - x_i^{(k)}|$ se aproxima de zero enquanto k cresce, isso gera um dos critérios de parada para o método, ou seja, se

$$d^{(k)} = \max_{1\leq i\leq n} |x_i^{(k+1)} - x_i^{(k)}| \leq \varepsilon \tag{3.16}$$

ou ainda, podemos efetuar o teste do erro relativo dado por

$$d_r^{(k)} = \frac{d^{(k)}}{||x^{(k+1)}||_\infty} \leq \varepsilon \tag{3.17}$$

Onde ε é a tolerância que fixa o grau de precisão das soluções, ou o processo irá parar se $k > M$, sendo k a iteração atual e N o número máximo de iterações.
Podemos reescrever o método de Jacobi numa forma mais concisa da seguinte maneira:

Decompondo a matriz de A em $Ax = b$ em $A = D + L + U$, onde

- L é a matriz triangular **estritamente** inferior, com $l_{ij} = a_{ij}$ se $i > j$, e $l_{ij} = 0$ se $i \geq j$ (os elementos da diagonal principal são zeros);

- D uma matriz diagonal com $d_{ii} = a_{ii}$;

- U uma matriz triangular **estritamente** superior com $u_{ij} = a_{ij}$ se $i < j$, e $l_{ij} = 0$ se $i \geq j$ (os elementos da diagonal principal são zeros);

Ou seja:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{U}$$

Com isso podemos definir $Ax = b$ substituindo A da seguinte forma:

$$\begin{aligned} Ax = b &\Leftrightarrow (D+L+U)x = b \\ &\Leftrightarrow Dx = -(L+U)x + b \end{aligned}$$

Como D é uma matriz diagonal, podemos facilmente calcular sua inversa, dessa forma temos:

$$x = -D^{-1}(L+U)x + D^{-1}b$$

Dessa forma, podemos definir as iterações de Jacobi pela equação matricial

$$x^{(k+1)} = -D^{-1}(L+U)x^{(k)} + D^{-1}b, \quad D^{(-1)} = [\frac{1}{d_{ii}}]$$

Agora, faremos $T_j = D^{(-1)}(L+U)$ e $c_j = D^{(-1)}b$, sendo T_j a matriz de iteração e c_j o vetor da iteração do método de Jacobi, com a isso a técnica de assume a forma

$$x^{(k+1)} = T_j x^k + c_j \tag{3.18}$$

que também pode ser representado por

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^k) \tag{3.19}$$

onde x_i^{k+1} é a estimativa atual da i-ésima variável na k-ésima iteração, a_{ij} é o coeficiente da i-ésima equação para a j-ésima variável, b_i é o termo independente da i-ésima equação e n é o número total de equações no sistema.

Exemplo 3.1.1. Resolver pelo método de Jacobi o sistema:

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

com $\varepsilon \leq 10^{-2}$ e $N = 10$

Solução.podemos reescrever o sistema de equações como:

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

Vamos começar com as seguintes estimativas iniciais para x_1, x_2:
$x_1^{(0)} = x_2^{(0)} = 0$ para $k = 0$ tem-se:

$$\begin{cases} x_1^{(1)} = \frac{1}{2}(1 + x_2^{(0)}) = \frac{1}{2}(1+0) = 0,5 \\ x_2^{(1)} = \frac{1}{2}(3 - x_1^{(0)}) = \frac{1}{2}(3-0) = 1,5 \end{cases}$$

para $k = 1$

$$\begin{cases} x_1^{(2)} = \frac{1}{2}(1 + x_2^{(1)}) = \frac{1}{2}(1 + 1,5) = 1,25 \\ x_2^{(2)} = \frac{1}{2}(3 - x_1^{(1)}) = \frac{1}{2}(3 - 0,5) = 1,25 \end{cases}$$

usaremos como critério de parada a equação 3.17, e continuando a iteração para $k = 2, 3, ...$ temos os seguintes valores:

k	**(0)**	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**
$x_1^{(k)}$	0	0.5	1.25	1.125	0.938	0.969	1.016	1.008	0.996	0.998
$x_2^{(k)}$	0	1.5	1.25	0.875	0.938	1.031	1.016	0.992	0.996	1.002
ε	-	1.5	0.75	0.375	0.187	0.093	0.047	0.024	0.012	0.006

Como $0,006 \leq 10^{-2}$ atingimos um dos critérios de parada, então os cálculos são encerrados e o vetor solução é definido por $\bar{x} = [0.998, 1.002]$ após 9 iterações. A precisão da solução pode ser melhorada aumentando o número de iterações e ajustando a tolerância de convergência.

Exemplo 3.1.2. Resolva o sistema linear:

$$\begin{cases} 10x_1 + 2x_2 + x_3 = 7 \\ x_1 + 5x_2 + x_3 = -8 \\ 2x_1 + 3x_2 + 10x_3 = 6 \end{cases}$$

pelo método de Jacobi com $x^{(0)} = \begin{pmatrix} 0,7 \\ -1,6 \\ 0,6 \end{pmatrix}$, e $\varepsilon = 0,05$.

Solução.Desta forma, o processo iterativo é

$$\begin{cases} x_1^{(k+1)} = (7 - 2x_2^{(k)} - x_3^{(k)})\frac{1}{10} = \frac{7}{10} - \frac{2}{10}x_2^{(k)} - \frac{1}{10}x_3^{(k)} \\ x_2^{(k+1)} = (-8 - x_1^{(k)} - x_3^{(k)})\frac{1}{5} = -\frac{8}{5} - \frac{1}{5}x_1^{(k)} - \frac{1}{5}x_3^{(k)} \\ x_3^{(k+1)} = (6 - 2x_1^{(k)} - 3x_2^{(k)})\frac{1}{10} = \frac{6}{10} - \frac{2}{10}x_1^{(k)} - \frac{3}{10}x_2^{(k)} \end{cases}$$

Para $k = 0$, temos

$$\begin{cases} x_1^{(1)} = 0,7 - 0,2x_2^{(0)} - 0,1x_3^{(0)} = 0,7 - 0,2(-1,6) - 0,1(0,6) = 0,96 \\ x_2^{(1)} = -1,6 - 0,2x_1^{(0)} - 0,2x_3^{(0)} = -1,6 - 0,2(0,7) - 0,2(0,6) = -1,86 \\ x_3^{(1)} = 0,6 - 0,2x_1^{(0)} - 0,3x_2^{(0)} = 0,6 - 0,2(0,7) - 0,3(1,6) = 0,94 \end{cases}$$

Desta forma, $x^{(1)} = \begin{pmatrix} 0,96 \\ -1,86 \\ 0,94 \end{pmatrix}$, Calculando $d_r^{(1)}$ como visto em 3.17, temos:

$|x_1^{(1)} - x_1^{(0)}| = 0,26$

$|x_2^{(1)} - x_2^{(0)}| = 0,26 \Rightarrow d_r^{(1)} = \dfrac{0,34}{\max_{1\leq i\leq n}|x_i^{(1)}|} = \dfrac{0,34}{1,86} = 0,1828 > \varepsilon$

$|x_3^{(1)} - x_3^{(0)}| = 0,34$

Prosseguindo para $k = 1$, temos

$$x^{(2)} = \begin{pmatrix} 0,978 \\ -1,98 \\ 0,966 \end{pmatrix} \Rightarrow d_r^{(2)} = \frac{0,12}{1.98} = 0,0606 > \varepsilon$$

Para $k = 2$

$$x^{(3)} \begin{pmatrix} 0,9994 \\ -1,9888 \\ 0,9984 \end{pmatrix} \Rightarrow d_r^{(3)} = \frac{0,0324}{1.9888} = 0,0163 < \varepsilon$$

Como $0,0163 < \varepsilon$, as iterações são encerradas, e o vetor solução obtido pelo método de Jacobi é

$$\overline{x} = x^{(3)} = \begin{pmatrix} 0,9994 \\ -1,9888 \\ 0,9984 \end{pmatrix}$$

3.2 Método iterativo de Gauss-Seidel

O método de Gauss-Seidel é uma extensão do método de Jacobi. O método de Gauss-Seidel é mais eficiente do que o anterior, pois utiliza os valores atualizados das incógnitas assim que eles estão disponíveis, em vez de esperar até que todas as incógnitas sejam atualizadas em cada iteração. O método também consiste em utilizar o sistemas de equações $Ax = b$ assim como vimos na equação 3.13, também isolamos cada variável $x_1, x_2, ..., x_n$, porém, para o método de Gauss-Seidel temos:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}) \\ x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)}) \\ x_3^{(k+1)} = \frac{1}{a_{33}}(b_2 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - a_{34}x^{(k)} - \cdots - a_{3n}x^{(k)}) \\ \vdots \\ x_n^{(k+1)} = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{k+1} - a_{n2}x_2^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)}) \end{cases} \tag{3.20}$$

ou então:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j<i} a_{ij}x_j^{(k)}), \;\; i = i,2,...,n \tag{3.21}$$

Assim como fizemos com o método de Jacobi, também podemos representar o método de Gauss-Seidel em sua forma matricial, com as definições de D, L e U já apresentadas anteriormente, podemos também escrever A como $A = (D - L - U)$ temos:

$$Ax = b \leftrightarrow (D-L-U)x = b \leftrightarrow Dx = (L+U)x + b \leftrightarrow$$
$$\leftrightarrow x = D^{-1}(L+U)x + D^{-1}b$$

Agora podemos aplicar o método iterativo de Gauss-Seidel, que consiste em encontrar o vetor $x^{(k+1)}$, então

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b$$

que da origem à fórmula de Gauss-Seidel

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b \tag{3.22}$$

podemos ainda fazer $T_g = (D-L)^{-1}U$ e $c_g = (D-L)^{-1}b$, a técnica fica então com a forma

$$x^{(k+1)} = T_g x^{(k)} + c_g \tag{3.23}$$

Exemplo 3.2.1. Resolver pelo método de Gauss-Seidel com $\varepsilon < 10^{-2}$

$$\begin{cases} 20x_1 + x_2 + x_3 + 2x_4 = 33 \\ x_1 + 10x_2 + 2x_3 + 4x_4 = 38,4 \\ x_1 + 2x_2 + 10x_3 + x_4 = 43,5 \\ 2x_1 + 4x_2 + x_3 + 20x_4 = 45,6 \end{cases}$$

Pelo método de Gauss-Seidel, temos:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{20}(33 - x_2^{(k)} - x_3^{(k)} - 2x_4^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{10}(38,4 - x_1^{(k+1)} - 2x_3^{(k)} - 4x_4^{(k)}) \\ x_3^{(k+1)} &= \frac{1}{10}(43,5 - x_1^{(k+1)} - 2x_2^{(k+1)} - x_4^{(k)}) \\ x_4^{(k+1)} &= \frac{1}{20}(45,6 - 2x_1^{(k+1)} - 4x_2^{(k+1)} - x_3^{(k+1)}) \end{aligned} \tag{3.24}$$

Sendo $x_1 = x_2 = x_3 = x_4 = 0$ como uma aproximação inicial arbitrária, temos os seguintes valores em cada iteração

k	**(0)**	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
$x_1^{(k)}$	0	1,6500	1,1730	1,1951	1,1996	1,2000
$x_2^{(k)}$	0	3,6750	2,5497	2,4110	2,4006	2,4000
$x_3^{(k)}$	0	3,4500	3,6020	3,6001	3,6001	3,6000
$x_4^{(k)}$	0	1,2075	1,4727	1,4982	1,4999	1,5000
ε	-	3,6750	1,1253	0,1387	0,0104	0,0005

Os valores de x_1, x_2, x_3 e x_4 estão convergindo para valores constantes iguais a aproximadamente $1,2000$, $2,4000$, $3,6000$ e $1,5000$, respectivamente. Isso sugere que o sistema linear tem uma única solução (ou pelo menos uma solução que o Método de Gauss-Seidel está convergindo).

Exemplo 3.2.2. Resolver pelo método de Gauss-Seidel o sistema do exemplo 3.1.1:

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

com $\varepsilon \leq 10^{-2}$ e $N = 10$

Solução. Vimos que utilizando o método de Jacobi, foram necessárias 9 iterações para atingir o critério de convergência, utilizaremos agora o método de Gauss-Seidel para fazer um comparativo dos métodos no mesmo sistema e mesmos critérios de parada. Então, seja $k = 0$ e $x_1^{(0)} = x_2^{(0)} = 0$, temos que:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}(1 + x_2^{(k)}) = \frac{1}{2}(1+0) = 0,5 = x_1^{(1)} \\ x_2^{(k+1)} = \frac{1}{2}(3 - x_1^{(k+1)}) = \frac{1}{2}(3 - \frac{1}{2}) = 1,25 = x_2^{(1)} \end{cases}$$

Continuando a iteração para $k = 1,2...n$ atingimos o critério de parada para $k = 5$, obtendo os resultados:

k	**(0)**	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
$x_1^{(k)}$	0	0,500	1,1250	0,9688	1,0078	0,9980
$x_2^{(k)}$	0	1,2500	0,9375	1,0156	0,9961	1,0010
ε	-	1,2500	0,6250	0,1562	0,0390	0.009

Ao compararmos os resultados obtidos com o método de Gauss-Seidel com os da tabela 3.1 do método de Jacobi, vemos claramente que o método de Gauss-Seidel converge mais rapidamente para a solução do sistema utilizando os mesmos critérios em ambos os casos

3.3 Condições suficientes para convergência dos métodos iterativos

As condições suficientes para a convergência dos métodos iterativos, como o Método de Jacobi e o Método de Gauss-Seidel, variam de acordo com o método específico e com as características do sistema linear que está sendo resolvido. No entanto, algumas condições gerais podem garantir a convergência. Uma condição que garante a convergência dos métodos de Gauss-Seidel e Jacobi é que a matriz seja estritamente diagonal dominante.

Isso significa que, em cada linha da matriz, o elemento na diagonal principal deve ser maior do que a soma dos elementos fora da diagonal principal na mesma linha. No entanto, é

importante observar que essa condição é mais forte do que o necessário para a convergência, o que significa que os métodos podem convergir mesmo em casos em que a matriz não seja estritamente diagonal dominante, mas essa condição é uma garantia segura de convergência quando atendida.

Teorema 3.2. Seja o sistema linear dado por $Ax = b$ e $\alpha_k = \sum_{j=1, j\neq k}^{n} \frac{|a_{kj}|}{|a_{kk}|}$ se

$$\alpha = \max_{1\leq k\leq n} \alpha_k < 1$$

então o método iterativo é convergente, e $x^{(k)}$ converge para a solução do sistema, independente da aproximação inicial $x^{(0)}$

Exemplo 3.3.1. Analisando a matriz A dos sistemas dos exemplos 3.1.1 e 3.1.2.

Solução.a)

$$A_1 = \begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix} \Rightarrow \alpha_1 = \frac{1}{2} = 0,5 < 1$$

$$\alpha_2 = \frac{1}{2} = 0,5 < 1$$

Solução.b)

$$A_2 = \begin{bmatrix} 10 & 2 & 1 \\ 1 & 5 & 1 \\ 2 & 3 & 10 \end{bmatrix} \Rightarrow \alpha_1 = \frac{2+1}{10} = 0,3 < 1$$

$$\alpha_2 = \frac{1+1}{5} = 0,4 < 1$$

$$\alpha_3 = \frac{2+3}{10} = 0,5 < 1$$

Como podemos ver, o $\max_{1\leq k\leq n} \alpha_k = 0,5 < 1$ em ambos os casos é verdadeiro, desta forma, pelo critério das linhas temos garantia da convergência dos métodos de Jacobi e Gauss-Seidel.

4 APLICAÇÕES

A obtenção da solução de um sistema linear através dos métodos diretos ou iterativos nem sempre é fácil. Alguns sistemas podem conter dezenas ou até milhares de equações e incógnitas, tornando a utilização desses métodos extremamente trabalhosa para aqueles que desejam obter a solução de sistemas de grande escala. Por esse motivo, o apoio computacional se torna indispensável, viabilizando a obtenção da solução de sistemas complexos de forma eficiente e precisa.

Através da utilização de software especializado e algoritmos de resolução, é possível automatizar o processo de solução de sistemas lineares, acelerando significativamente o cálculo e reduzindo a probabilidade de erros humanos. Além disso, o poder computacional atual permite lidar com sistemas de dimensões impressionantes que, de outra forma, seriam praticamente impossíveis de resolver manualmente.

Portanto, a integração da computação no processo de resolução de sistemas lineares é fundamental para a resolução de problemas complexos em diversas áreas, como engenharia, ciência da computação, física e economia, facilitando a análise e a tomada de decisões embasadas em dados precisos.

Um dos softwares amplamente utilizados para a criação de rotinas e algoritmos é o **GNU Octave**, um programa de livre acesso e com uma vasta gama de recursos para análise numérica, simulação e resolução de equações matemáticas. Ele oferece uma interface amigável e poderosa que permite aos usuários desenvolverem algoritmos personalizados, realizar cálculos complexos e explorar dados de forma eficiente, tornando-o uma ferramenta essencial para pesquisadores, engenheiros e cientistas em diversas áreas.

Na referência 1, o capítulo 6 e 7, os algoritmos *6.1*, *7.1* e *7.2* nos fornecem respectivamente os pseudocódigos para a **eliminação de Gauss**, **método iterativo de Jacobi** e **método iterativo de Gauss-Seidel** em qualquer linguagem ou editor de programação. Através destes, fui capaz de desenvolver meus próprios códigos utilizando o **GNU Octave**, com capacidade de solucionar qualquer sistema linear (se houver solução) utilizando os métodos mencionados anteriormente.

Gauss.m

```
function [x] = Gauss(A, b)
E = [A,b]
n = size(E,1);
for i=1:n-1 %inicia o processo de eliminação
  for p=i:n % percorrer todas as linhas da diagonal para abaixo
    if E(p,i)!=0 %fixei a coluna da diagonal i
      break %sai do for em p
    endif
  endfor
L = E(p,:); %Armazena a linha E(p,:) em L
E(p,:) = E(i,:); %E(p,:) recebe E(i,:)
E(i,:) = L; %Troca de linha efetuada
  for j = i+1:n
    if E(i,i) != 0 %Se o pivô for não nulo...
      m(j,i) = E(j,i)/E(i,i); %Define os multiplicadores m
      E(j,:)-= m(j,i)*E(i,:); %Se o pivô é não nulo, ocorre então a eliminação
    else %Caso não exista um pivô diferente de zero
      E %Exibe a matriz aumentada impossível de ser resolvida
      error('Não existe uma solução única')
    endif
  endfor
endfor
if E(n,n) == 0 %Caso impossível
  error('Não existe uma solução única')
endif
display('E após a troca')
E %Exibe a matriz aumentada transformada em matriz triangular superior
x(n) = E(n,n+1)/E(n,n); %Começa a substituição regressiva
for i = n-1:-1:1
  S=0; %Define uma variável para o somatório
  for j=i+1:n
    S=S+E(i,j)*x(j); %Define um somatório
  endfor
  x(i) = (E(i,n+1) - S)/E(i,i); %vetor solução do sistema
endfor
x(:,i)
endfunction
```

Figura 1 – Código da eliminação de Gauss no Octave
Fonte: O autor - 2023

A função no Octave chamada de **Gauss** recebe dois parâmetros, a matriz A, com os elementos a_{ij}, para $i, j = 1, 2, ..., n$, e o vetor b com os elementos b_i, para $i = 1, 2, ..., n$. Logo em seguida se obtem a matriz aumentada $E = [A, b]$, e então é iniciado o processo de eliminação de acordo como vimos na seção 2.3.

Jacobi.m

```
function [x] = Jacobi(A, b, x, tol, N) %Recebe os parametros
  clc %Limpa a tela
  E = [A,b] %Matriz aumentada E = [A|b]
  n = size(A,1); %Recebe quantas incógnitas existem em A
  k = 0; %Valor inicial da iteração
  x0 = x; %x0 recebe a aproximação inicial
  while k <= N %Condicional de parada da iteração
    for i = 1:n %Começa a iteração das linhas
      s = 0;  %Define uma variável para o somatório
      for j = 1:n %Começa a iteração das colunas
        if j != i %Condicional para o somatório
          s = s + A(i,j)*x0(j); %Somatório com o ultimo vetor sol. obtido
        endif
      endfor
      x(i) = (b(i) - s)/A(i,i); %Armazena o vetor solução para k+1
    endfor
    eps = norm(x-x0, 'inf') %Armazenando a norma em uma variável
    if eps < tol %Condicional de parada
      display('O procedimento foi bem-sucedido')
      display(["Foram realizados ", num2str(k+1), " iterações"])
      return
    endif
    x0 = x %Salvando o novo vetor solução x em x0
    k += 1; %Incrementa k para a nova iteração
  endwhile  %Fim da iteração caso k>N
  error('Número máximo de iterações excedido')
endfunction
```

Aqui a função recebe 5 parâmetros, a matriz A dos coeficientes a_{ij}, os componentes de b_i de b, a aproximação inicial $x^{(0)}$, a tolerância tol e o número máximo de iterações N. Desta forma, o algoritmo itera até atingir a convergência (definida pela tolerância tol) ou até atingir o número máximo de iterações N.

GSeidel.m

```
1  function [x] = GSeidel(A, b, x, tol, N) %recebe os parametros
2    clc %Limpa a tela
3    E = [A,b] %Matriz aumentada E = [A|b]
4    n = size(A,1); %Recebe quantas incógnitas existem em A
5    k = 0; %Valor inicial da iteração
6    x0 = x; %x0 recebe a aproximação inicial
7    while k <= N %Condicional de parada da iteração
8      for i = 1:n %Começa a iteração das linhas
9        s = 0;  %Define uma variável para o somatório
10       for j = 1:n %Começa a iteração das colunas
11         if j != i %Condicional para o somatório
12           s = s + A(i,j)*x(j);  %Somatório com os valores sol. mais atuais
13         endif
14       endfor
15       x(i) = (b(i) - s)/A(i,i); %Armazena o vetor solução para k+1
16     endfor
17     eps = norm(x-x0, 'inf') %Armazenando a norma em uma variável
18     if eps < tol  %Condicional de parada
19       display('O procedimento foi bem-sucedido')
20       display(["Foram realizados ", num2str(k+1), " iterações"])
21       return
22     endif
23     x0 = x; %Salvando o novo vetor solução x em x0
24     k += 1; %Incrementa k para a nova iteração
25   endwhile  %Fim da iteração caso k>N
26   error('Número máximo de iterações excedido')
27 endfunction
```

Figura 2 – Código do método iterativo de Gauss-Seidel no Octave
Fonte: O autor - 2023

Note que o algoritmo do método de Jacobi na figura **??** e o algoritmo do método de Gauss-Seidel em 2 são de fato bastante similares, o que de fato os torna diferentes é o fato de que no de Gauss-Seidel utilizamos os valores mais atuais de x, o que pode ser visto na linha **12** de ambas as imagens. Enquanto no método de Jacobi multiplicamos o elemento a_{ij} pela última aproximação de $x0$, no método de Gauss-Seidel utilizamos os valores de x recém adquiridos, tornando com que o método de Gauss-Seidel convirja mais rapidamente na maioria dos casos, como vimos ao comparar o exemplo 3.1.1 utilizando ambos os métodos.

Mas apesar disso, é importante ressaltar que existirão casos em que o método de Jacobi será mais eficiente que o método de Gauss-Seidel, fazendo com que seja necessário uma análise por iteração para verificar qual dos métodos convergirá mais rapidamente, como veremos a seguir.

4.1 Exercícios aplicados com apoio do software Octave

Exemplo 4.1.1. Utilize do a eliminação de Gauss para solucionar os problemas a seguir

a) $3,333x_1 + 15920x_2 - 10,333x_3 = 15913$
$2,222x_1 + 16,71x_2 + 9,612x_3 = 28,544$
$1,5611x_1 + 5,1791x_2 + 1,6852x_3 = 8,4254$

b) $\pi x_1 + \sqrt{2}x_2 - x_3 + x_4 = 0$
$ex_1 - x_2 + x_3 + 2x_4 = 1$
$x_1 + x_2 - \sqrt{3}x_3 + x_4 = 2$
$-x_1 - x_2 + x_3 - \sqrt{5}x_4 = 3$

***Solução.*a)** Para utilizarmos a função *Gauss* que vemos na figura 1, devemos primeiramente fornecer os parâmetros que a função pede, ou seja, A e b, desta forma:

$$\underbrace{\begin{bmatrix} 3,333 & 15920 & 10,333 \\ 2,222 & 16,71 & 9,612 \\ 1,5611 & 5,1791 & 1,6852 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 15913 \\ 28,544 \\ 8,4254 \end{bmatrix}}_{b}$$

Para isso, devemos chamar a função na janela de comandos do Octave utilizando seu nome e fornecendo as matrizes correspondentes, então:

» Gauss([3.333,15920,-10.333; 2.222,16.71,9.612;1.5611,5.1791,1.6852] , [15913;28.544;8.4254])
Desta forma, o resultado obtido na janela de comandos será:

Podemos observar que em alguns casos, o Octave costuma representar números que possuem muitas casas decimais utilizando notação ciêntifica. Vemos facilmente como ele critou

```
>> Gauss([3.333,15920,-10.333; 2.222,16.71,9.612;1.5611,5.1791,1.6852],[15913
;28.544;8.4254])
E =

   3.3330e+00   1.5920e+04  -1.0333e+01   1.5913e+04
   2.2220e+00   1.6710e+01   9.6120e+00   2.8544e+01
   1.5611e+00   5.1791e+00   1.6852e+00   8.4254e+00

E após a troca
E =

   3.3330e+00   1.5920e+04  -1.0333e+01   1.5913e+04
            0  -1.0597e+04   1.6501e+01  -1.0580e+04
            0            0  -5.0781e+00  -5.0781e+00

ans = 1.0000
ans =

   1.0000   1.0000   1.0000
```

Figura 3 – Solução de 4.1.1 a)
Fonte: O autor - 2023

a matriz aumentada E, após isso representou a matriz E na forma triangular superior, e então realizou a substituição regressiva, retornando o vetor solução para $x_1 = x_2 = x_3 = 1$

Solução.**b)** Utilizaremos novamenten o mesmo comando no octave, fornecendo novamente os parâmetros A e b para a função no Octave

$$\underbrace{\begin{bmatrix} \pi & \sqrt{2} & -1 & 1 \\ e & -1 & 1 & 2 \\ 1 & 1 & -\sqrt{3} & 1 \\ -1 & -1 & 1 & -\sqrt{5} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}}_{b}$$

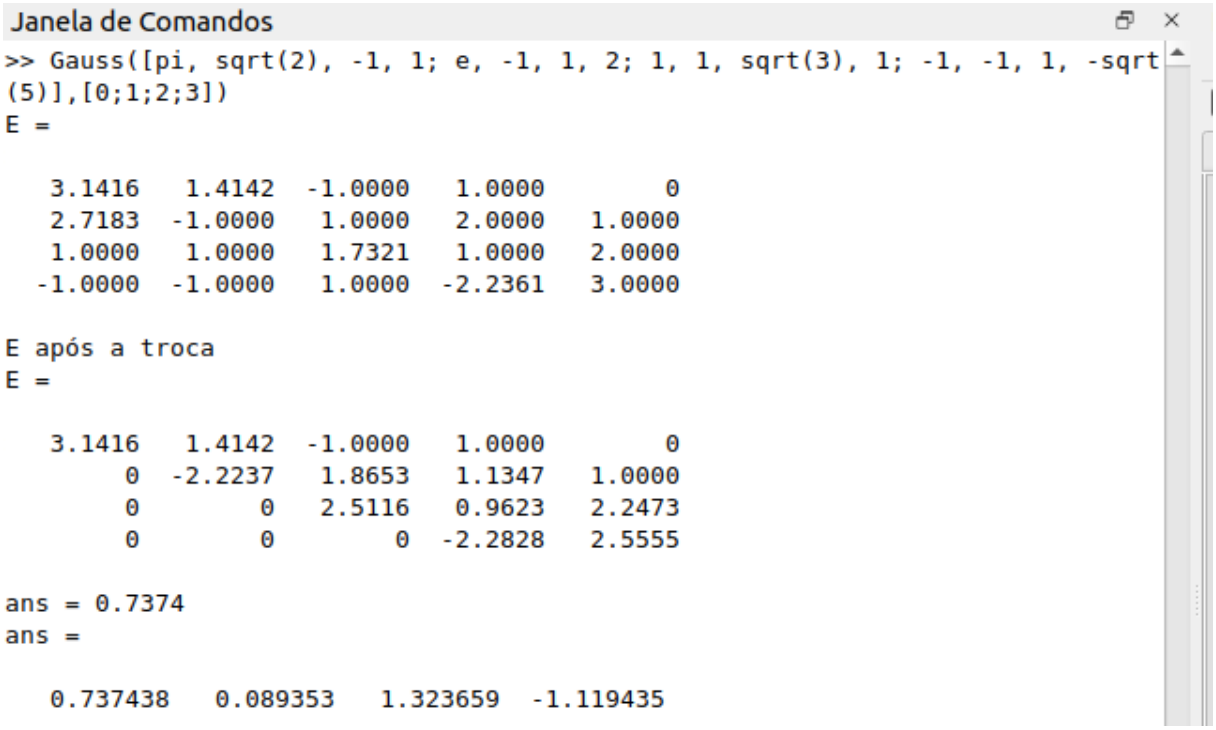

Figura 4 – Solução de 4.1.1 b)
Fonte: O autor - 2023

Facilmente obtemos os vetor solução $\bar{x} = [0.737438, 0.089353, 1.323659, -1.119435]$, considerando os erros de aproximação.

Exemplo 4.1.2. Encontre as duas primeiras iterações do método de Jacobi e Gauss-Seidel para os seguintes sistemas lineares, utilizando $x^{(0)} = 0$:

a)
$$\begin{aligned} +4x_1 + x_2 - x_3 \quad\quad + x_5 &= 6 \\ -x_1 - 3x_2 + x_3 + x_4 \quad\quad &= 6 \\ +2x_1 + x_2 + 5x_3 - x_4 - x_5 &= 6 \\ -x_1 - x_2 - x_3 + 4x_4 \quad\quad &= 6 \\ +2x_2 - x_3 + x_4 + 4x_5 &= 6 \end{aligned}$$

b)
$$\begin{aligned} +4x_1 - x_2 \quad\quad\quad\quad\quad\quad &= 0 \\ -x_1 + 4x_2 - x_3 \quad\quad\quad\quad &= 5 \\ -x_2 + 4x_3 \quad\quad\quad\quad &= 0 \\ +4x_4 - x_5 \quad\quad &= 6 \\ -x_4 + 4x_5 - x_6 &= -2 \\ -x_5 + 4x_6 &= 6 \end{aligned}$$

Solução. **a)** Para o primeiro sistema, temos os seguintes componentes A e b

$$\underbrace{\begin{bmatrix} 4 & 1 & -1 & 0 & 1 \\ -1 & -3 & 1 & 1 & 0 \\ 2 & 1 & 5 & -1 & -1 \\ -1 & -1 & 1 & 4 & 0 \\ 0 & 2 & -1 & 1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \\ 6 \end{bmatrix}}_{b}$$

Chamaremos então a função Jacobi, como apresenta a figura **??**, preenchendo seus parâmetros para que possa então ser executado:

» Jacobi([4,1,1,0,1;-1,-3,1,1,0;2,1,5,-1,-1;-1,-1,-1,4,0;0,2,-1,1,4],[6;6;6;6;6],[0,0,0,0,0],0.01,1)

Como o exercício não especifica uma tolerância, daremos um valor arbitrário de *0.01*. Note também que 1 é o valor máximo de iteração, porém k parte de 0, totalizando duas iterações como pede o exercício. Após executar este comando, o resultado obtido na janela de comandos será:

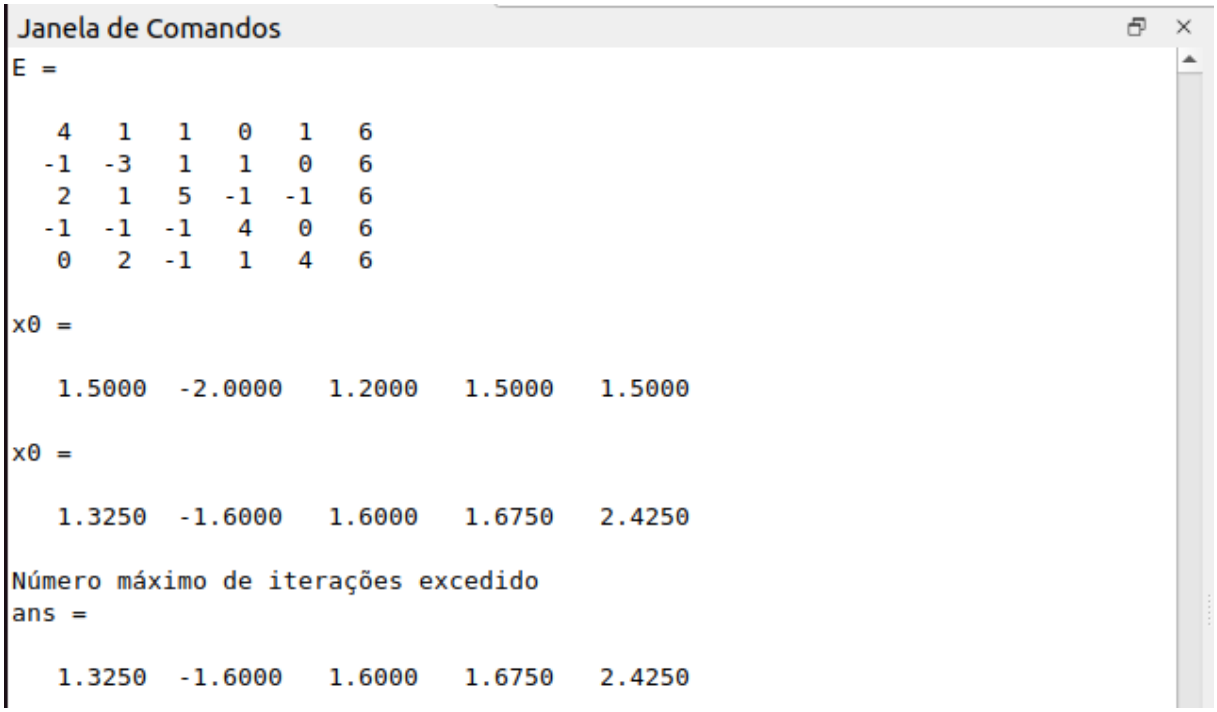

Figura 5 – Solução de 4.1.2 a) pelo método de Jacobi
Fonte: O autor - 2023

Portanto, no método de Jacobi o vetor solução após duas iterações é:
$\bar{x} = [1.325, -1.6, 1.6, 1.67, 2.425]$

Agora utilizando o método de Gauss-Seidel, visto na figura 2, novamente forneceremos os parâmetros necessários para a função funcionar, da mesma forma que fizemos com o método de Jacobi

» GSeidel([4,1,1,0,1;-1,-3,1,1,0;2,1,5,-1,-1;-1,-1,-1,4,0;0,2,-1,1,4],[6;6;6;6;6],[0,0,0,0,0],0.01,1)

Novamente fornecemos as matrizes A e b, o chute inicial x, uma tolerância arbitrária e o número de iterações, obtendo:

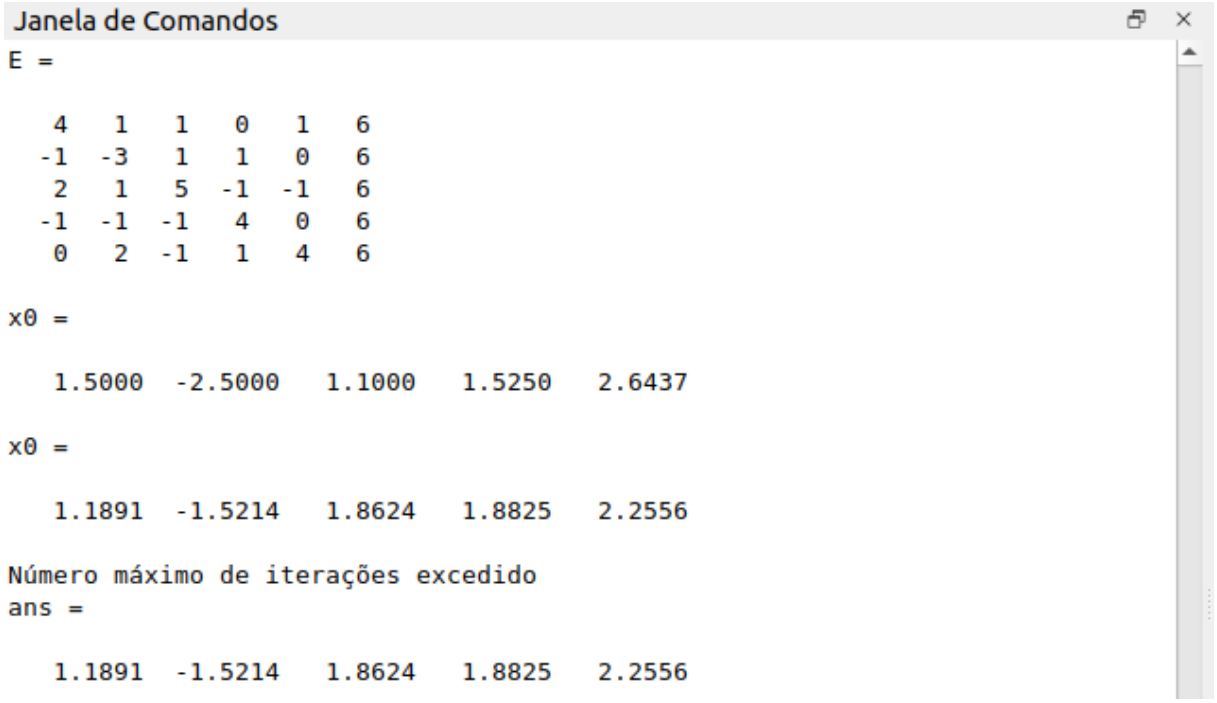

Figura 6 – Solução de 4.1.2 a) pelo método de Gauss-Seidel
Fonte: O autor - 2023

Sendo o vetor solução dado por 2 iterações do método de Gauss-Seidel:
$\bar{x} = [1.1891, -1.5214, 1.8624, 1.8825, 2.2556]$
***Solução.*b)** Para o segundo sistema, temos as seguintes matrizes

$$\underbrace{\begin{bmatrix} 4 & -1 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 0 \\ 5 \\ 0 \\ 6 \\ -2 \\ 6 \end{bmatrix}}_{b}$$

Fornecendo novamente os parâmetros para a função de Jacobi e Gauss-Seidel no octave, temos respectivamente:

» Jacobi([4,-1,0,0,0,0;-1,4,-1,0,0,0;0,-1,4,0,0,0;0,0,0,4,-1,0;0,0,0,-1,4,-1;0,0,0,0,-1,4],[0;5;0;6;-2;6],[0,0,0,0,0,0],0.01,1)

» GSeidel([4,-1,0,0,0,0;-1,4,-1,0,0,0;0,-1,4,0,0,0;0,0,0,4,-1,0;0,0,0,-1,4,-1;0,0,0,0,-1,4],[0;5;0;6;-2;6],[0,0,0,0,0,0],0.01,1)

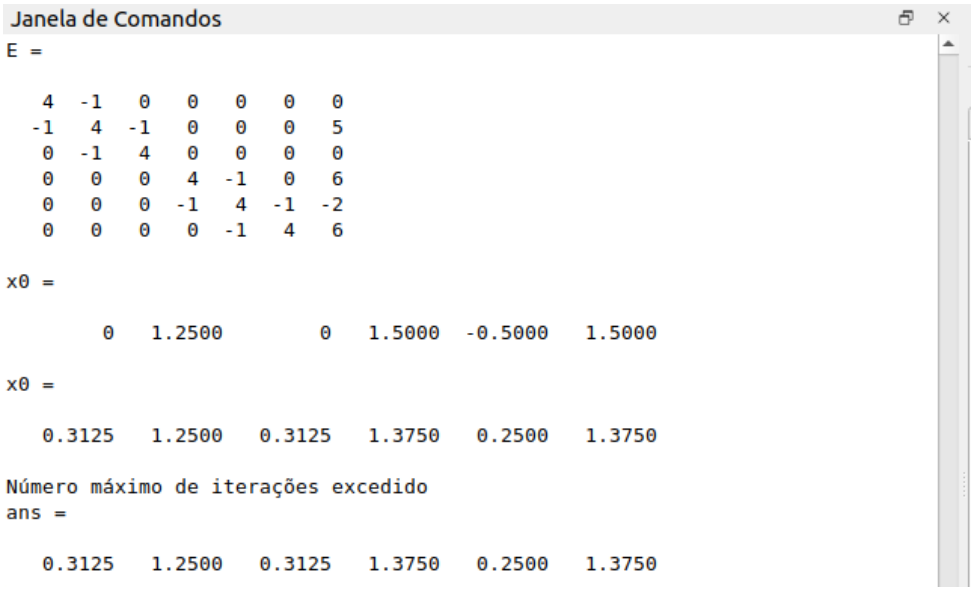

Figura 7 – Solução de 4.1.2 b) pelo método de Jacobi
Fonte: O autor - 2023

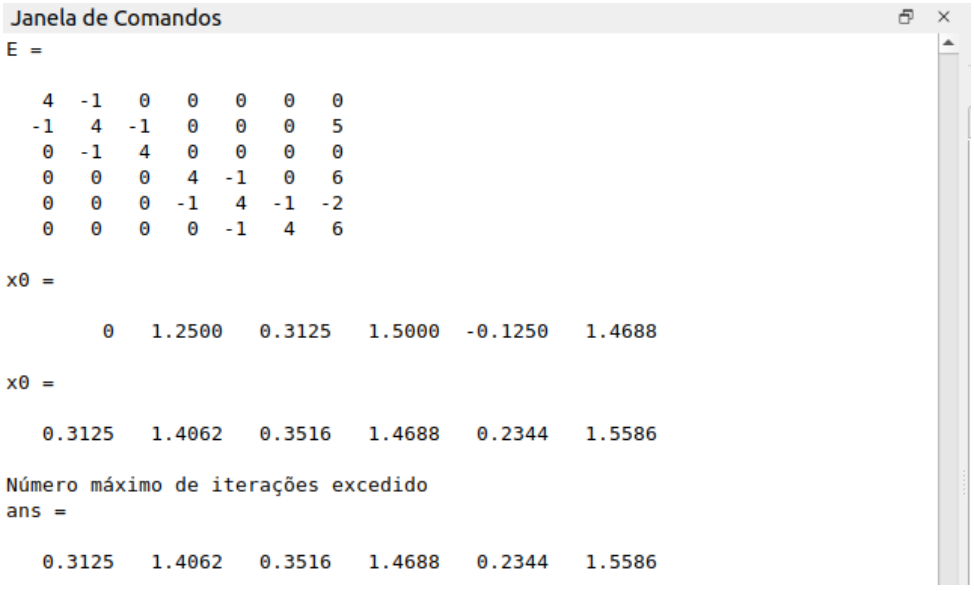

Figura 8 – Solução de 4.1.2 b) pelo método de Gauss-Seidel
Fonte: O autor - 2023

Temos que os vetores soluções para os métodos de Jacobi e Gauss-Seidel são, respectivamente:

$\bar{x}_j = [0.3125, 1.25, 0.3125, 1.375, 0.25, 1.375]$
$\bar{x}_g = [0.3125, 1.4062, 0.3516, 1.4688, 0.2344, 1.5586]$

Lembrando que para obter resultados mais precisos, basta apenas ajustar o número de iterações e a tolerância.

Exemplo 4.1.3. O sistema linear

$$x_1 + 2x_2 - 2x_3 = 7$$
$$x_1 + x_2 + x_3 = 2$$
$$2x_1 + 2x_2 + x_3 = 5$$

tem a solução $\bar{x} = [1, 2, -1]$

a)Use o método de Jacobi com $x^{(0)} = 0$ para obter uma aproximação da solução do sistema linear com precisão de 10^{-5} na normal l_∞

b)Mostre que o método de Gauss-Seidel aplicado como no item (a) falha em dar uma boa aproximação em 25 iterações.

Solução.**a)** Como o exemplo não especifica o número de iterações que deverá ser feito, daremos a N o valor de 100 para que possamos testar o código. Ao aplicar a matriz no Octave, podemos verificar que o resultado coincide com o que nos é dado pela questão: Vemos que foram

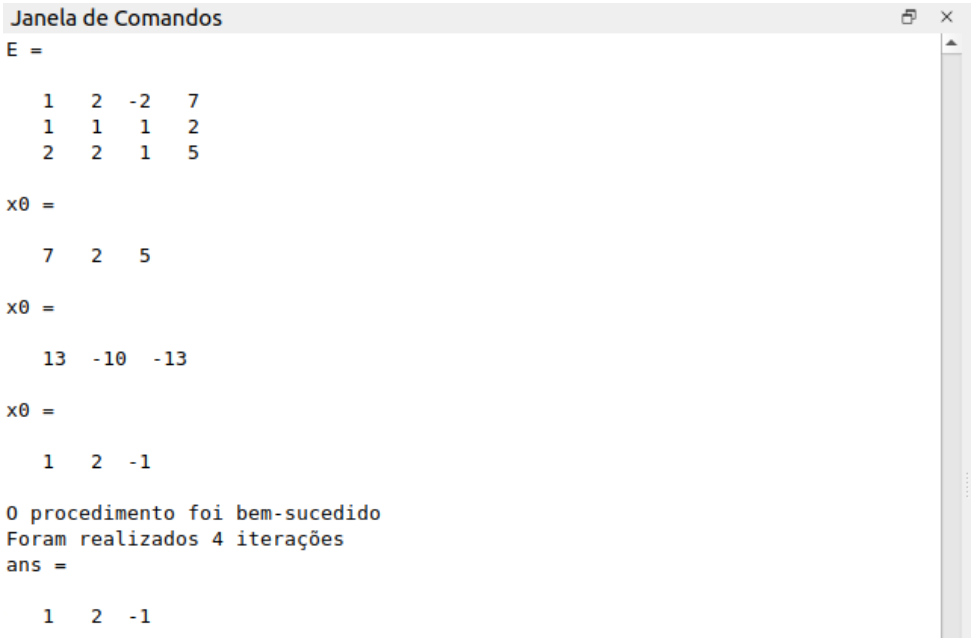

Figura 9 – Solução de 4.1.3 a) pelo método de Jacobi
Fonte: O autor - 2023

necessárias apenas 4 iterações.

Solução.**b)**Agora iremos resolver o mesmo sistema com o método de Gauss-Seidel aplicado no Octave com 25 iterações:

Figura 10 – Solução de 4.1.3 b) pelo método de Gauss-Seidel
Fonte: O autor - 2023

Com isso é provado que após 25 iterações, o método de Gauss-Seidel falha em obter um resultado preciso do sistema, tornando o método de Jacobi o mais eficiente para esse caso. Porém é bom ressaltar novamente que a escolha do método irá variar de sistema para sistema, o método de Jacobi falhará em alguns casos assim como o método de Gauss-Seidel falhará em outros, fazendo-se necessário um estudo do sistema a cada iteração para definir qual é o melhor.

Exemplo 4.1.4. Um cabo coaxial é feito de um condutor interno de 0,1 polegada quadrada e de um condutor externo de 0,5 polegada quadrada. O potencial em um ponto na seção transversal do cabo é descrito pela equação de Laplace. Suponha que o condutor interno seja mantido a 0 volt e que o condutor interno seja mantiro a 110 volts. A aproximação do potencial entre os dois

condutores requer a resolução do seguinte sistema linear:

$$\begin{bmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 4 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 4 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 4 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 4 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ w_{11} \\ w_{12} \end{bmatrix} = \begin{bmatrix} 220 \\ 110 \\ 110 \\ 220 \\ 110 \\ 110 \\ 110 \\ 110 \\ 220 \\ 110 \\ 110 \\ 220 \end{bmatrix}$$

a)Resolva o sistema linear usando o método de Jacobi com $x^{(0)} = 0$ e $TOL = 10^{-2}$

b) Repita a parte *b)* usando o método de Gauss-Seidel

Solução.**a)**Utilizando o método de Jacobi implementado no Octave, encontramos: O vetor solu-

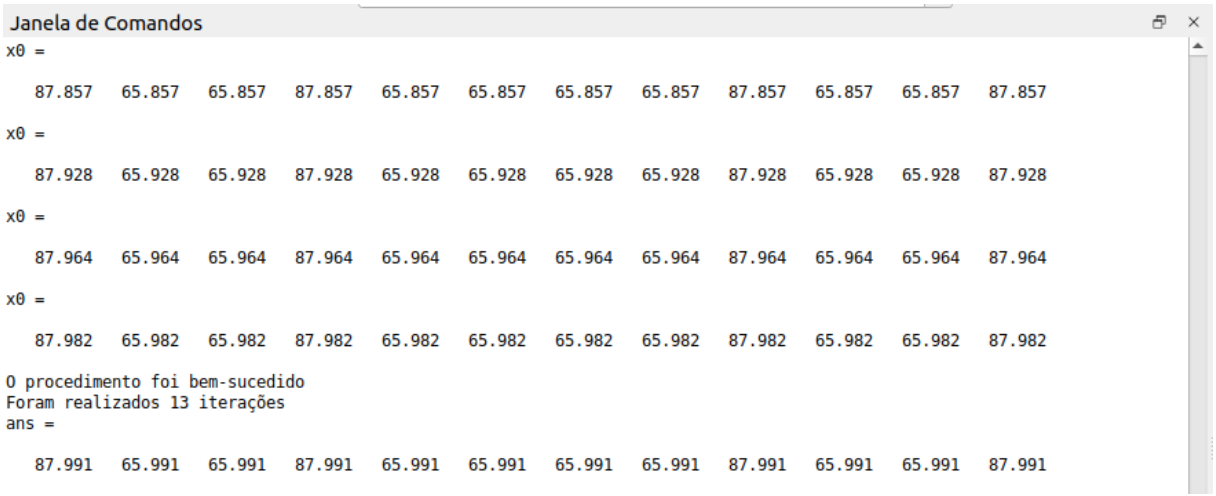

Figura 11 – Solução de 4.1.4 b) pelo método de Jacobi
Fonte: O autor - 2023

ção é dado por

$$\bar{x} = [87.991, 65.991, 65.991, 87.991, 65.991, 65.991, 65.991, 65.991, 87.991, 65.991, 65.991, 87.991]$$

Solução.**b)** Agora utilizando o método de Gauss-Seidel implementado no Octave, temos: Vemos

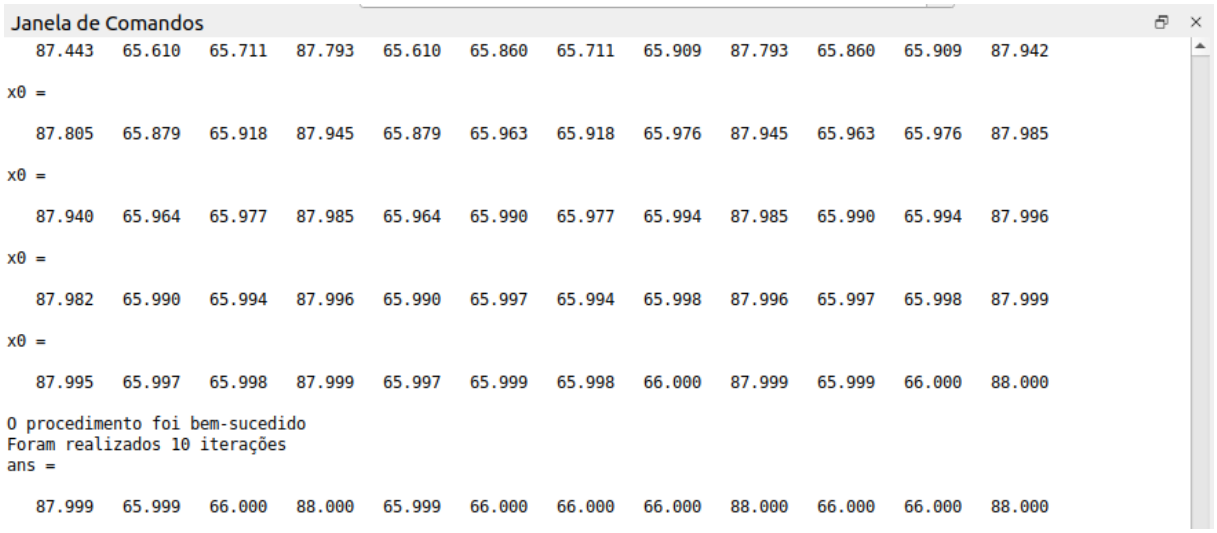

Figura 12 – Solução de 4.1.4 c) pelo método de Gauss-Seidel
Fonte: O autor - 2023

que com o método de Gauss-Seidel foram realizadas menos iterações até atingir a convergência, tornando o método de Gauss-Seidel mais eficiente nesse caso, sendo o vetor solução dado por:

$$\bar{x} = [87.999, 65.999, 66, 88, 65.999, 66, 66, 66, 88, 66, 66, 88]$$

5 CONCLUSÃO

Os métodos diretos e iterativos são poderosas ferramentas para solução de sistemas lineares, como foi mostrado neste trabalho. Seu uso pode ainda ser amplificado com apoio computacional, tornando possível solucionar uma vasta variedade de sistemas de equações diferentes, com possibilidade de ajustar com precisão os critérios de parada dos métodos. O software matemático GNU Octave é uma ferramenta fundamental para o estudo de métodos numéricos, nos possibilitando a criação de rotinas e funções capazes de realizar os cálculos de cada método em fração de segundos. Vale mencionar que os códigos de Gauss, Jacobi, e Gauss-Seidel foram criados a partir da interpretação deste autor com base nos algoritmos vistos na referência [1] deste trabalho, a criação de funções no Octave para as mesmas finalidades podem diferenciar de autor para autor, podendo haver simplificações e melhorias dependendo do conhecimento de cada um.
Mas entre os métodos diretos e iterativos, qual escolher dado um sistema linear na forma $Ax = b$? Bem, isso depende de uma série de fatores, como condições de convergência, esparsidade da matriz ou erros de arredondamento. Conforme vimos, os métodos diretos são processos finitos e, portanto, teoricamente, obtêm a solução de qualquer sistema não singular de equações. Já os métodos iterativos tem convergência assegurada apenas sob determinadas condições.

E quanto ao programa computacional utilizado, Octave é a única opção? Uma forte opção similar ao Octave é o **MatLab**, que também possui um alto desempenho e uma linguagem similar ao Octave, mas as opções não se limitam apenas a softwares, também existem uma variedade de linguagens de programação, como **C#**, **R**, e **Python**, amplamente utilizado para o processamento de grande volumes de dados, com uma sintaxe amigável e com uma sólida comunidade na internet, possuindo bibliotecas próprias para a criação de algoritmos numéricos como os vistos nesse trabalho.

REFERÊNCIAS

BURDEN, Richard. FAIRES, Douglas. BURDEN, Annette **Análise Numérica.** 10ª ed. São Paulo: Cengage Learning, 2020.

A. Quarteroni, F. Saleri **Cálculo científico com MATLAB e Octave**. Milano: Springer-Verlag Italia, 2006

BARROSO, Leônidas Conceição et al. **Cálculo Numérico (Com Aplicações)**. 2ª ed. São Paulo: HARBRA, 1987.

FRANCO, Neide Barroso. **Cálculo numérico**. São Paulo: Pearson Prentice Hall, 2006.

RUGGIERO, Márcia A. Gomes. **Cálculo numérico: aspectos teóricos e computacionais**. 2ª ed. São Paulo: Pearson Makron Books, 1996.