UNIVERSIDADE FEDERAL DO TOCANTINS
CÂMPUS UNIVERSITÁRIO PROF. DR. SÉRGIO JACINTHO LEONOR
CURSO DE LICENCIATURA EM MATEMÁTICA

PAULO CÉZAR RODRIGUES OLIVEIRA

MÉTODOS NUMÉRICOS PARA SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES UTILIZANDO O MATLAB

ARRAIAS-TO
2019

PAULO CÉZAR RODRIGUES OLIVEIRA

MÉTODOS NUMÉRICOS PARA SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES UTILIZANDO O MATLAB

Monografia apresentada ao curso de Licenciatura em Matemática da Universidade Federal do Tocantins Câmpus Universitário Prof. Dr. Sérgio Jacintho Leonor, para obtenção do título de graduado e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Prof. Me. Dirlei Ruscheinsky.

ARRAIAS-TO
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Federal do Tocantins

O48m Oliveira, Paulo Cézar Rodrigues.

Métodos Numéricos para Solução de Sistemas de Equações Lineares Utiliando o MATLAB. / Paulo Cézar Rodrigues Oliveira. – Arraias, TO, 2019.

60 f.

Monografia Graduação - Universidade Federal do Tocantins – Câmpus Universitário de Arraias - Curso de Matemática, 2019.

Orientador: Dirlei Ruscheinsky

1. Sistemas de Equações Lineares. 2. Métodos Numéricos. 3. MATLAB. 4. Software. I. Título

CDD 510

Elaborado pelo sistema de geração automática de ficha catalográfica da UFT com os dados fornecidos pelo(a) autor(a).

FUNDAÇÃO UNIVERSIDADE FEDERAL DO TOCANTINS - UFT
CAMPUS UNIVERSITÁRIO PROF. DR. SÉRGIO JACINTHO LEONOR - ARRAIAS-TO
CURSO DE LICENCIATURA EM MATEMÁTICA

ATA DE DEFESA DE MONOGRAFIA

Aos dez dias do mês de dezembro, do ano de dois mil e dezenove, às 10:00 horas, na Sala de Defesa (06T) do ProfMat, realizou-se a Defesa da Monografia intitulada "Métodos Numéricos para Solução de Sistemas de Equações Lineares Utilizando o MATLAB", como Trabalho de Conclusão de Curso (TCC) do acadêmico **Paulo Cezar Rodrigues Oliveira**, do Curso de Licenciatura em Matemática do Câmpus Universitário Prof. Dr. Sérgio Jacintho Leonor, sob orientação do Professor Me. Dirlei Ruscheinsky, tendo como demais membros da banca avaliadora o Professor Me. Fernando Soares de Carvalho e o Professor Esp. Claudio Ulisse.

Após a defesa e as considerações da Banca, o trabalho foi considerado APROVADO.

Nada mais havendo a relatar, assinam esta Ata o professor orientador e os demais componentes da banca.

	Sim	Não
O trabalho está apto a ser publicizado no Repositório Institucional da Universidade Federal do Tocantins através da Biblioteca Digital de Monografias	X	

Professor Me. Dirlei Ruscheinsky
Orientador

Professor Me. Fernando Soares de Carvalho
Examinador 1

Professor Esp. Claudio Ulisse
Examinador 2

Arraias (TO)

Ao meu pai Evangelista Oliveira Costa e à minha mãe Irani Rodrigues Oliveira

Agradecimentos

Agradeço primeiramente ao meu Deus por conceder esta oportunidade em minha vida, por me amparar nos momentos difíceis de saúde e cansaço, que não foram poucos.

Agradeço imensamente às pessoas que carrego comigo no fundo do coração: Irani Rodrigues, Evangelista Oliveira, Iasmim Carvalho e José Bonifácio Oliveira. Foram de suma importância nesta longa e dura caminhada, que sem eles não teria conseguido percorrer.

Aos meus amigos e colegas de profissão: Cleidiane Martins, Dearley Fernandes, Jean Carlos, Jorge Luiz, Luiz Eduardo, Luiz Marles, Manoel Neto, Mathaus Layne, Murilo Alberto. Mesmo que de forma indireta, contribuíram significativamente nesta minha caminhada, seja nos momentos de trabalho ou descontração.

A todos os meus professores, pelo conhecimento construído ao longo do curso de Licenciatura em Matemática.

Um agradecimento especial ao professor Fernando Soares que iniciou comigo o trabalho com esta monografia e ao professor Dirlei Ruscheinsky, meu orientador, pela paciência e conhecimento a mim proporcionados.

"A Matemática apresenta invenções tão sutis que poderão servir não só para satisfazer os curiosos como, também para auxiliar as artes e poupar trabalho aos homens."
(Descartes, 1596 - 1650)

RESUMO

O presente trabalho aborda alguns métodos numéricos usados para resolver sistemas de equações lineares, sendo eles: Eliminação Gaussiana, Fatoração LU, Jacobi, Gauss Seidel, Sobre-Relaxação-Sucessiva e Gradiente Conjugado. Para uma melhor usabilidade dos métodos será utilizado o software MATLAB para que os mesmos possam ser implementados. O trabalho com o software destaca-se pela facilidade que este oferece ao tipo de problema aqui abordados, com funções pré-definidas que auxiliam no momento da implementação. No decorrer deste trabalho será apresentada a descrição dos métodos, expondo seus funcionamentos seguidos de exemplos para aplicar os conceitos. Será feita a análise dos dados obtidos por cada código levando em consideração o tempo gasto e quantidade de iterações(para métodos iterativos).

Palavras-chave: Sistemas de equações lineares. Métodos numéricos. MATLAB.

ABSTRACT

The present work addresses some numerical methods used to solve systems of linear equations, being them: Gaussian Elimination, LU Factorization, Jacobi, Gauss Seidel, Successive Over-Relaxation, and Conjugate Gradient. For a better method usability the software will be used MATLAB so that they can be implemented. The work with software stands out for the ease that it offers to the kind of problem approached here, with predefined functions that help at the moment of implementation. During this work will be presented the description of the methods, exposing your working followed by examples to apply the concepts. It will be done the analysis of data obtained by each code leading in consideration the time spent and the quantity of iterations (for iterative methods).

Keywords: Systems of linear equations. Numerical methods. MATLAB.

Lista de Ilustrações

Lista de Tabelas

Lista de Códigos

Lista de símbolos

a_{ij}	Coeficiente que compõe a matriz A
x_j	Variável que compõe o vetor x
b_i	Termo independente que compõe o vetor b
$(A\|b)$	Matriz expandida do sistema
$\max_{1<i<n}$	Maior valor de 1 a n
l_i	Linha da matriz
$\|a_{ij}\|$	Módulo do coeficiente
A^t	Matriz transposta de A
$det(A)$	Determinante da matriz A
$\gamma(A)$	Número condição de A
$\|\|A\|\|$	Norma de A
A^{-1}	Inversa da matriz A
$\rho(R)$	Raio espectral da Matriz de iteração R
λ_i	Autovalores
ε	Precisão do erro
Σ	Somatório
$\prod$	Produtório
ξ_i	Parâmetro para Critério de Linhas
β_i	Parâmetro para Critério de Sassenfeld
ω	Parâmetro de relaxação
(Ax,x)	Produto interno
∇	Gradiente
$\frac{\partial}{\partial x_i}$	Derivada parcial

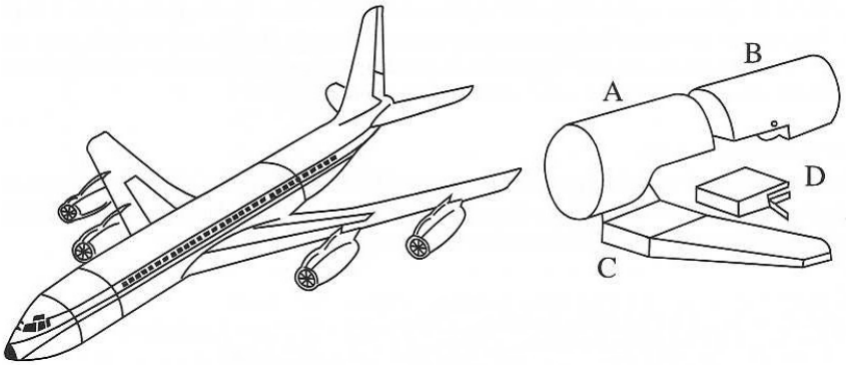
Sumário

1 INTRODUÇÃO

Neste trabalho buscaremos abordar os diferentes métodos para resolução de sistemas de equações lineares $Ax = b$, visto que estes são encontrados em problemas das mais diversas áreas, sendo algumas delas a matemática, engenharia, biologia e economia, por exemplo. Tais sistemas de equações podem ser desde os mais simples e pequenos a até sistemas de grande porte com milhares de equações.

Os sistemas de equações lineares surgem da necessidade de modelar problemas práticos como por exemplo, analisar a estrutura de uma aeronave por meio do método de elementos finitos que conduzirá a sistemas lineares de grandes dimensões.

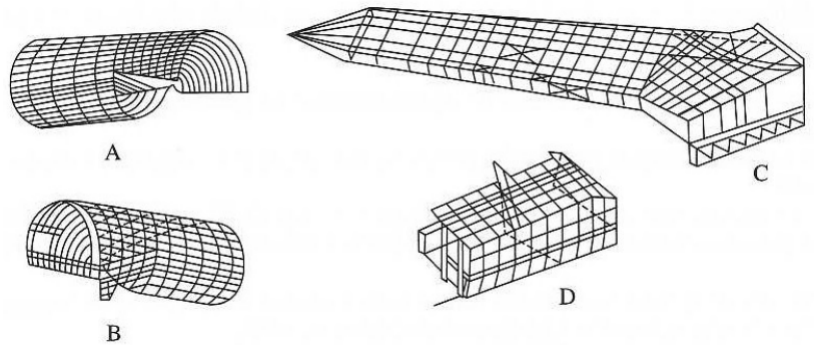
Figura 1 – Estruturas da aeronave



Fonte: Sperandio, Mendes e Silva (2003, p. 67)

As estruturas A, B, C e D da figura 1 são discretizadas por elementos finitos (figura 2), o que conduz a sistemas de equação lineares de grande porte (SPERANDIO; MENDES; SILVA, 2003, p. 67).

Figura 2 – Discretização das subestruturas por elementos finitos



Fonte: Sperandio, Mendes e Silva (2003, p. 68)

Diante da grande variedade de sistemas lineares que surgem, foram sendo criados ao longo dos anos diferentes métodos numéricos para que fossem alcançados resultados mais precisos de forma mais eficiente. Atualmente os métodos numéricos podem ser classificados em dois grupos: diretos e iterativos. Os métodos diretos podem ser classificados como aqueles que, após determinado número de operações, conduzem à solução exata do sistema desconsiderando os erros de arredondamento que podem ser cometidos pela máquina/computador.

No grupo de métodos diretos podemos citar alguns que são mais conhecidos e utilizados, dentre eles: o próprio cálculo da inversa de A, definido por $x = A^{-1}b$; a Regra de Crammer; eliminação de Gauss; fatoração LU. Neste trabalho o primeiro será desconsiderado, visto que calcular a inversa e logo após multiplicá-la por b envolve um grande número de operações tornando o método não competitivo. Da mesma forma para a Regra de Crammer, que num sistema $n \times n$ efetua o cálculo de $(n+1)$ determinantes, tornando a solução trabalhosa por exigir um grande número de operações como no anterior(RUGGIERO; LOPES, 1996).

Assim, os métodos diretos aqui trabalhados serão a Eliminação de Gauss e a Fatoração LU. Visto que nas bibliografias estudadas há um direcionamento positivo para a eficiência destes dois métodos diretos.

Os métodos iterativos produzem uma sequência de vetores $x^{(k)}$ que, com uma dada precisão, convergem para a solução do sistema. Os métodos deste tipo aqui usados serão: Gauss Seidel, Jacobi[1], SOR e Gradiente Conjugado.

Para Cunha (2000, p. 29), a escolha do método numérico

> "[...] depende de cada caso. Se os métodos diretos tem a vantagem de fornecer solução após um número finito de passos e não dependem de condições de convergência, eles podem ser inviáveis quando o sistema é muito grande ou mal condicionado."

Nesse sentido, Cunha (2000, p. 57) afirma que "Os métodos iterativos são convenientes para sistemas grandes e esparsos que aparecem com frequência na discretização de equações diferenciais."

A recente aquisição da licença do *software* MATLAB pelo campus Arraias, tornou possível um trabalho como este, possibilitando uma melhor análise dos dados que serão obtidos pelos códigos com as rotinas de cada método numérico. Através destas rotinas poderemos testar diversos sistemas de equações lineares a fim de comparar os dados obtidos e assim poder apontar o método numérico que é mais eficiente para determinado sistema de equações.

Este trabalho está organizado em três capítulos. No capítulo 2 trataremos dos métodos numéricos, descrevendo-os e apresentando suas definições e teoremas. Usaremos ainda, exemplos para aplicarmos tais conhecimentos. O capítulo 3 será destinado a comparações entre os métodos,

[1] Este método também aparece em algumas bibliografias como Gauss Jacobi e Jacobi Richardson

onde estaremos analisando a quantidade de iterações e o tempo gasto por cada método para chegar à solução exata ou aproximada de determinado sistema linear.

1.1 MATLAB

O *software* MATLAB (abreviatura do inglês *MATrix LABoratory* - Laboratório de Matrizes) é um programa de computador otimizado para computação numérica de análise e visualização de dados. Surgiu como um programa para operações matemáticas sobre matrizes, mas com o passar dos anos foi sendo aperfeiçoado para um sistema computacional bastante útil e flexível (BECKER *et al.*, 2010).

O *software* é do tipo **interpretador**, ou seja, as instruções definidas na linguagem são executadas diretamente linha por linha. Diferentemente de um compilador, que a partir de um programa de entrada produz outro, que equivale ao original, mas em uma linguagem executável como por exemplo a linguagem de máquina.

O programa em sua versão R2019a, utiliza por idioma padrão em seu ambiente de trabalho o inglês. Assim, um usuário com nível de inglês iniciante e que saiba o mínimo de programação, não terá dificuldades em programar. Visto que, na internet temos à disposição várias apostilas/tutoriais de universidades, grupos de pesquisa, etc. que podem auxiliar o programador. Além do próprio site da *MathWorks* que apresenta uma vasta gama de trechos com códigos implementados e comentados de funções pré-definidas do programa.

A linguagem de programação MATLAB, que o programa executa, foi desenvolvida na década de 70 por Clever Moler, então presidente do departamento de Ciências da Computação da Universidade do Novo México.

Na década de 80, através de uma parceria com Steve Bangert e o engenheiro jack Litte, a MathWorks (detentora dos direitos autorais) foi fundada e o MATLAB foi reescrito em linguagem C. Tornando-se um ambiente integrado de modelagem de sistemas e algoritmos, ideal para implementação de projetos complexos, a ferramenta é um produto líder em computação numérica e científica (CHAIA; DAIBERT,).

A linguagem MATLAB é considerada uma linguagem multi-paradigma. Para Lopes, Rangel e Martha (2019, p. 5) "Um paradigma de programação é a forma que se apresenta a estrutura e execução do código."

Assim, no MATLAB o programador pode utilizar o paradigma que julgar ser mais eficiente, sendo eles: Programação Estruturada, Programação Orientada a Objetos e Programação Orientada a Eventos.

Comparada a outras linguagens de programação de alto nível como o Fortran, C, C++ e Python, muito utilizadas na ciência e na engenharia, a linguagem MATLAB sai na frente quando o problema envolve cálculos matemáticos dos mais triviais aos mais elaborados. Um exemplo

simples é a multiplicação de duas matrizes, no MATLAB obtemos o resultado após aplicarmos a operação de multiplicação representada por "$*$".

Código 1.1 – Produto de matrizes em MATLAB

```
1
2 A = [1 2 3 ; 4 3 2 ; 1 4 9]  % Matriz A 3x3
3 T = [1 5 3 ; 1 3 2 ; 1 4 9]  % Matriz T 3x3
4
5 Q = A*T          % Armazena o produto das matrizes
6                  % em outra matriz 3x3
```

Fonte: elaborado pelo autor

Já em outras linguagens é necessário o uso de laços de repetição, demandando mais linhas de código para execução, como é o caso da linguagem C. No código feito em C logo abaixo, temos o mesmo exemplo de multiplicação entre as duas matrizes vistas no código 1.1.

Código 1.2 – Produto de matrizes em C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int matriz_A[3][3]={{1,2,3},{4,3,2},{1,4,9}}; // Matriz A 3x3
6      int matriz_T[3][3]={{1,5,3},{1,3,2},{1,4,9}}; // Matriz T 3x3
7      int matriz_Q[3][3];
8      int i=0, j=0, k=0, sum=0;
9
10     // Calcula a matriz Q=AxT
11     for(i=0; i<3; i++){
12         for(j=0; j<3; j++){
13             for(k=0; k<3; k++){
14                 sum += matriz_A[i][k] * matriz_T[k][j];}
15             matriz_Q[i][j]=sum;
16             sum=0;}}
17
18     // Imprime a matriz A
19     printf("\nMatriz A\n");
20     for(i=0; i<3; i++){
21         for(j=0; j<3; j++){
22             printf("%i  ", matriz_A[i][j]);}
23         printf("\n");}
```

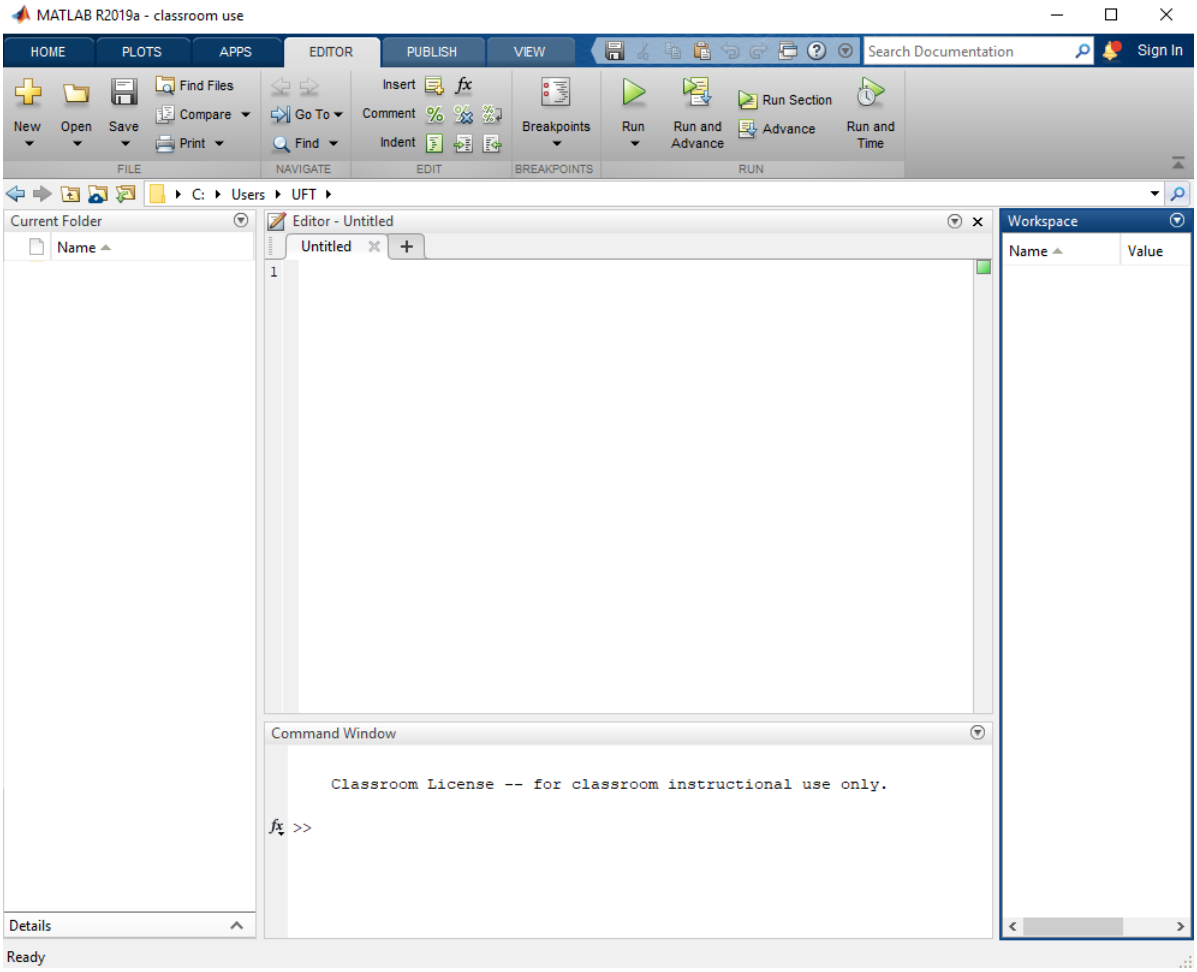
```
24
25      // Imprime a matriz T
26      printf("\nMatriz T\n");
27      for(i=0; i<3; i++){
28          for(j=0; j<3; j++){
29              printf("%i  ", matriz_T[i][j]); }
30          printf("\n"); }
31
32      // Imprime a matriz Q
33      printf("\nMatriz Q=AxT\n");
34      for(i=0; i<3; i++){
35          for(j=0; j<3; j++){
36              printf("%i  ", matriz_Q[i][j]); }
37          printf("\n"); }
38
39      printf("\n");
40      return 0;
41  }
```

Fonte: elaborado pelo autor

Ao observarmos os dois códigos escritos nas duas linguagens percebemos a facilidade que o MATLAB proporciona no momento da implementação e consequentemente torna a velocidade de execução do código muito maior.

Na figura 3, apresentamos a tela inicial do MATLAB:

Figura 3 – Interface inicial



Fonte: elaborada pelo autor

Nesta tela inicial podemos ver o quadro de título **Editor**, é onde podemos usar a maior parte dos recursos presentes no *software*. Aqui podemos desenvolver as rotinas programáveis que desejarmos e salvá-las em arquivo padrão do MATLAB, na extensão ".m". O quadro com título **Workspace** nos mostra as variáveis que estão salvas na memória à medida em que vão sendo criadas pelo programador.

No quadro com título **Command Window** é mostrado o resultado do código executado no Editor. Uma outra opção deste é usá-lo como uma calculadora, ou seja, podemos executar cálculos rápidos onde não são necessárias longas rotinas programadas, não havendo necessidade de salvar o código para executar.

Um outro ponto forte da linguagem é o trabalho com gráficos. A título de exemplo da utilização de um gráfico bidimensional, faremos a representação gráfica do seguinte sistema de

equações:

$$\begin{cases} 2x_1 + x_2 = 3 \\ x_1 - 3x_2 = -2 \end{cases} \tag{1.1}$$

Abaixo apresentamos as linhas de código que são necessárias para exibir o gráfico através do comando `plot`:

Código 1.3 – Representação gráfica do sistema 1.1

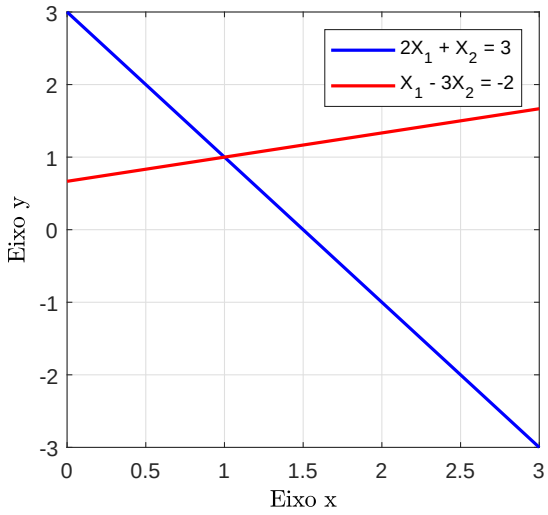
```
clear all
clc

x = 0:0.5:3                          % Intervalo em que o grafico aparece
x_1 = 3-2*x                          % Primeira equacao
x_2 = (2+x)/3                        % Segunda equacao
plot(x,x_1,x,x_2)                    % Comando que cria o grafico
legend('2X1 + X2 = 3','X1 - 3X2 = -2') % Legenda indicando os graficos
                                     % de cada equacao
grid on                              % Adiciona malhas ao grafico
xlabel('Eixo x')                     % Indica o eixo x
ylabel('Eixo y')                     % Indica o eixo y
```

Fonte: elaborado pelo autor

Na figura 4 temos o gráfico gerado após a execução do código.

Figura 4 – Representação Gráfica do Sistema 1.1



Fonte: elaborada pelo autor

2 MÉTODOS NUMÉRICOS

Neste capítulo iremos descrever os métodos numéricos utilizados e resolver exemplos para que os conhecimentos possam ser aplicados. Serão usados como base alguns referenciais teóricos como (CUNHA, 2000), (BURDEN; FAIRES, 2003), (CAMPOS, 2007), (RUGGIERO; LOPES, 1996)

Os Sistemas de Equações Lineares os quais iremos resolver com os métodos citados, serão do tipo

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

mais precisamente em sua forma matricial

$$Ax = b,$$

sendo A matriz de coeficientes, x o vetor solução e b o vetor de termos independentes:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{2.2}$$

2.1 Métodos Diretos

Como dito na introdução deste trabalho, os métodos diretos aqui tratados serão a Eliminação Gaussiana e a Fatoração LU. A seguir iremos descrever cada um dos métodos, apresentando suas definições e exemplos para que possamos aplicar a teoria.

2.1.1 Eliminação Gaussiana

Iniciaremos apresentando o método de Eliminação Gaussiana, que leva esse nome devido a seu criador Carl Friedrich Gauss (1777-1855). Este método consiste em transformar a matriz dos coeficientes A, numa outra que seja equivalente, na forma triangularizada. Tal processo é desenvolvido com base no seguinte teorema:

Teorema 2.1. *Dado um sistema de equações lineares, podemos submetê-lo a uma sequência de operações elementares:*

i. Multiplicação de uma equação por uma constante $m \neq 0$:

$$l_i \leftarrow ml_i;$$

ii. Soma do múltiplo de uma equação à outra equação:

$$l_i \leftarrow l_i + m_{ik} l_k.$$

iii. Permutação das linhas do sistema:

$$l_i \leftrightarrow l_k$$

Para descrever o processo de triangularização usaremos a notação de *matriz expandida* $A|b$, de tamanho $n \times (n+1)$. Assim, ao efetuarmos uma operação numa linha de A, automaticamente estaremos alterando a linha correspondente em b.

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right] \quad \begin{array}{c} l_1 \\ l_2 \\ \vdots \\ l_n \end{array} \tag{2.3}$$

O processo será feito em etapas, sendo que em cada uma serão eliminados os coeficientes a_{ij} das incógnitas que estão abaixo da diagonal, ou seja $i > j$. Em cada etapa serão definidos *multiplicadores*, representados por:

$$m_{ij} = \frac{a_{ij}}{a_{ii}}$$

com a_{ii} chamado de *pivô* de cada coluna (RUGGIERO; LOPES, 1996).

1ª Etapa

Nesta etapa eliminaremos na primeira coluna os elementos que se encontram abaixo da diagonal. Supondo que $a_{11} \neq 0$ vamos subtrair l_1, multiplicada por m_{i1}, de todas as $n-1$ equações, algebricamente teremos em 2.3:

$$\begin{array}{rl} l_2^{(2)} \leftarrow & l_2 - m_{i1} l_1 \\ \vdots & \vdots \\ l_n^{(2)} \leftarrow & l_n - m_{i1} l_1 \end{array}$$

O índice superior entre parênteses indica que a equação da linha correspondente receberá um valor atualizado após a operação. Assim o sistema 2.3 é atualizado para:

$$[\tilde{A}|\tilde{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right] \tag{2.4}$$

2ª Etapa

Seguindo de 2.4, nesta etapa iremos eliminar da segunda coluna os elementos que estão abaixo da diagonal. Desta vez, supondo $a_{22} \neq 0$, iremos subtrair l_2, multiplicada por m_{i2}, de todas as $n-2$ equações:

$$\begin{array}{rcl} l_3^{(3)} \leftarrow & l_3^{(2)} - m_{i2} l_2^{(2)} \\ \vdots & \vdots \\ l_n^{(3)} \leftarrow & l_n^{(2)} - m_{i2} l_2^{(2)} \end{array}$$

o sistema resulta em

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{array}\right] \tag{2.5}$$

Os processos se repetem, seguindo a mesma ideia acima, até que todos os elementos abaixo da diagonal sejam eliminados. Se os pivôs não se anularem, o sistema estará na forma triangular:

$$\begin{array}{rcrcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ & + & a_{22}^{(2)}x_2 & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & & & & & \vdots & & \vdots \\ & & & & & & a_{nn}^{(n)}x_n & = & b_n^{(n)} \end{array} \tag{2.6}$$

Caso algum dos pivôs se anularem no processo de eliminação, é preciso que façamos a troca das linhas de modo que os elementos não sejam nulos na posição dos pivôs.

Para resolver o sistema triangular superior, usaremos uma substituição retrocedida das variáveis em 2.6 da seguinte forma:

$$\left\{\begin{array}{rcl} x_n & = & b_n/a_{nn} \\ x_{n-1} & = & (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \\ \vdots & & \vdots \\ x_1 & = & (b_1 - a_{1n}x_n - \cdots - a_{13}x_3 - a_{12}x_2)/a_{11} \end{array}\right. \tag{2.7}$$

Estratégia do Pivoteamento

Voltando ao caso dos multiplicadores $m_{ij} = a_{ij}/a_{ii}$, temos como denominador o pivô (elemento da diagonal). Há casos em que este pivô é zero, tornando a operação impossível de continuar.

Para os casos como este é conveniente utilizar o *pivoteamento parcial* que, conforme afirma Cunha (2000, p. 37) consiste na "troca sistemática de linhas, de modo que o pivô seja o

maior elemento, em valor absoluto, da coluna que estamos eliminando.", ou seja,

$$|a_{rk}| = \max_{k \leq i \leq n} |a_{ik}| .$$

Nos sistemas em que a matriz dos coeficientes é:

- diagonalmente dominante

$$|a_{ii}| > \sum_{i \neq j} |a_{ii}|$$

ou

- simétrica e positiva definida

$$A^T = A \quad \text{e} \quad x^T A x > 0$$

o processo de eliminação pode ser aplicado sem o pivoteamento.

Exemplo 2.1. *Resolva o sistema linear abaixo, utilizando o Método de Eliminação de Gauss.*

$$\begin{cases} 5x_1 - 2x_2 + 4x_3 + x_4 = 4 \\ 2x_1 + 3x_3 + 7x_4 = 9 \\ -8x_1 + x_2 + 5x_3 + 3x_4 = 3 \\ 6x_1 + 2x_2 - x_3 + 9x_4 = 2 \end{cases}$$

Como o elemento da diagonal na segunda coluna do sistema é nulo, nos trará problemas nas operações com o multiplicador m_{i2}, pois teremos denominador igual a 0. Para evitar problemas do tipo, aplicaremos o Método de Eliminação de Gauss com Pivoteamento.

Usando a mesma representação de 2.3, teremos o sistema na forma extendida:

$$[A|b] = \left[\begin{array}{cccc|c} 5 & -2 & 4 & 1 & 4 \\ 2 & 0 & 3 & 7 & 9 \\ -8 & 1 & 5 & 3 & 3 \\ 6 & 2 & -1 & 9 & 2 \end{array}\right]$$

Aplicando o pivoteamento, faremos agora a permutação das linhas, para evitar o pivô nulo na linha 2, bem como organizarmos na diagonal os maiores elementos em valor absoluto.

$$[A|b] = \left[\begin{array}{cccc|c} -8 & 1 & 5 & 3 & 3 \\ 5 & -2 & 4 & 1 & 4 \\ 2 & 0 & 3 & 7 & 9 \\ 6 & 2 & -1 & 9 & 2 \end{array}\right] \begin{array}{l} l_1 \leftrightarrow l_3 \\ l_2 \leftrightarrow l_3^{(1)} \\ \\ \\ \end{array}$$

Na primeira troca $(l_1 \leftrightarrow l_3)$, a l_1 passa a ser a l_3 e a l_3 assume a posição da l_1. O processo é análogo na segunda troca $(l_2 \leftrightarrow l_3^{(1)})$.

Tabela 1 – Eliminação de Gauss no sistema linear do exemplo 2.1

(k)	l	Multiplicador	**A**				**b**	Operação
	1		-8	1	5	3	3	
1	2	-0,25	5	-2	4	1	4	$l_2 - m_{21}l_1$
	3	-0,625	2	0	3	7	9	$l_3 - m_{31}l_1$
	4	-0,75	6	2	-1	9	2	$l_4 - m_{41}l_1$
	2		0	2,75	2,75	11,25	4,25	
2	3	0,0909	0	0,25	4,25	7,75	9,75	$l_3 - m_{32}l_2$
	4	-0,5	0	-1,375	7,125	2,875	5,875	$l_4 - m_{42}l_2$
3	3		0	0	8,5	8,5	8	
	4	0,4706	0	0	4	6,7273	9,3636	$l_4 - m_{43}l_3$
	4		0	0	0	2,7273	5,5989	

Fonte: elaborada pelo autor

O processo de eliminação está descrito na tabela 1.

Na primeira coluna temos as etapas (k) do processo. Na coluna l temos as linhas em que as operações estão ocorrendo. A coluna do Multiplicador armazena os multiplicadores envolvidos em cada etapa. Na última coluna estão as operações que foram utilizadas para eliminar os termos.

É possível observarmos na tabela 1 que a cada passo k foram efetuados novos pivoteamentos após a eliminação da coluna correspondente à etapa.

Concluídas as eliminações a matriz estendida estará triangularizada.

$$\left[\begin{array}{cccc|c} -8 & 1 & 5 & 3 & 3 \\ 0 & 2,75 & 2,75 & 11,25 & 4,25 \\ 0 & 0 & 8.5 & 8,5 & 8 \\ 0 & 0 & 0 & 2,7273 & 5,5989 \end{array}\right]$$

Para resolver o sistema triangular usaremos a substituição retrocedida, como mostrado em 2.7, o resultado é apresentado no vetor linha abaixo:

$$\bar{x} = \left[\begin{array}{cccc} -1,0176 & -5,7412 & -1,1118 & 2,0529 \end{array}\right]^t$$

2.1.2 Fatoração LU

Trataremos agora do método de Fatoração LU que, resumidamente, consiste em decompor a matriz dos coeficientes A em dois fatores:

- *matriz triangular inferior L* composta pelos multiplicadores m_{ij} da etapa de triangularização do método de eliminação;
- *matriz triangular superior U*, cujos elementos u_{ij} são os mesmos a_{ij} que compõe a matriz A ao final da triangularização.

Na forma matricial L e U terão a seguinte forma

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ m_{n1} & m_{n2} & \cdots & 1 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} l_{ij} = 0; & i < j \\ l_{ii} = 1; & i = j \\ l_{ij} = m_{ij}; & i > j \end{cases}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \quad \Rightarrow \quad \begin{cases} u_{ij} = a_{ij}^{(k)}; & i \leq j \\ u_{ij} = 0; & i > j \end{cases}$$

Para encontrarmos as duas matrizes utilizaremos o próprio método de eliminação de Gauss. Armazenaremos os multiplicadores m_{ij} na matriz L, triangular inferior.

Para desenvolvermos o método, utilizaremos a matriz na forma genérica A:

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

Do Método de Eliminação, os multiplicadores que atuam na anulação da primeira coluna são os seguintes:

$$\begin{aligned} m_{21} &= \tfrac{a_{21}}{a_{11}} \\ &\vdots \\ m_{n1} &= \tfrac{a_{n1}}{a_{11}} \end{aligned}$$

Armazenando o negativo desses multiplicadores numa matriz M de diagonal unitária, teremos

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

Observe que ao multiplicarmos $M^{(1)}$ por A o resultado será a matriz $A^{(1)}$:

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

sendo $A^{(1)}$ a mesma matriz obtida ao final da 1ª etapa da Eliminação Gaussiana.

Os multiplicadores que atuam na anulação da segunda coluna abaixo da diagonal, são:

$$\begin{aligned} m_{32} &= \tfrac{a_{32}}{a_{22}} \\ &\vdots \\ m_{n2} &= \tfrac{a_{n2}}{a_{22}} \end{aligned}$$

Da mesma forma que representamos em $M^{(1)}$ façamos também em $M^{(2)}$:

$$M^{(2)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & -m_{n2} & \cdots & 1 \end{bmatrix}.$$

Tal processo se repete até que tenhamos $M^{(k-1)}$, ou seja, uma matriz que contenha o último multiplicador usado na eliminação da penúltima coluna de A:

$$m_{n,n-1} = \frac{a_{n,n-1}}{a_{n-1,n-1}}$$

$$M^{(k-1)} = \begin{bmatrix} 1 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & & 1 & \vdots \\ 0 & \cdots & -m_{n,n-1} & 1 \end{bmatrix}.$$

Ao efetuarmos o produto $M^{(2)} \ldots M^{(k-1)}.A^{(1)}$, teremos

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

Pode-se perceber que a matriz $A^{(k)}$ triangular superior é a mesma obtida ao final do processo de eliminação. Das matrizes de multiplicadores, segue que o produto

$$M^{(1)}M^{(2)} \ldots M^{(k-1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -m_{21} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -m_{n-1,1} & -m_{n-1,2} & \cdots & 1 & 0 \\ -m_{n1} & -m_{n2} & \cdots & -m_{n,n-1} & 1 \end{bmatrix}$$

dá origem à matriz que efetua a triangularização de $A^{(k)}$.

Nosso objetivo é obter $A = LU$, temos que $M^{(1)}M^{(2)}\dots M^{(k-1)}.A = A^{(k)}$. Fazendo

$$A^{(k)} = U$$

$$\Rightarrow M^{(1)}M^{(2)}\dots M^{(k-1)}.A = U \Rightarrow A = [M^{(1)}]^{-1}[M^{(2)}]^{-1}\dots[M^{(k-1)}]^{-1}.U.$$

Sendo

$$L = [M^{(1)}]^{-1}[M^{(2)}]^{-1}\dots[M^{(k-1)}]^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ m_{21} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n-1,1} & m_{n-1,2} & \cdots & 1 & 0 \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{bmatrix},$$

temos que $A = LU$, ou seja,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ m_{n1} & m_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

O teorema 2 logo abaixo nos garante a unicidade das matrizes L e U no processo de fatoração da matriz dos coeficientes A.

Teorema 2.2. *Sejam $A = a_{(ij)}$ uma matriz quadrada de ordem n, e A_k a matriz construída das k primeiras linhas e colunas de A. Suponha que $det(A_k) \neq 0$ para k = 1, 2,..., (n - 1). Então, existe uma única matriz triangular inferior $L = (m_{ij})$, com $m_{ii} = 1$, $1 \leq i \leq n$ e uma única matriz triangular superior $U = u_{ij}$ tais que LU = A. Como também $det(A) = u_{11}u_{22}...u_{nn}$.*

Para a demonstração faremos indução sobre n.

Para $n = 1$, teremos

$$a_{11} = l_{11}.u_{11} = 1.u_{11}$$

e $det(A) = u_{11}$.

Seja A uma matriz de ordem k, dividimos a mesma em submatrizes:

$$L = \begin{bmatrix} L_{k-1} & 0 \\ m & 1 \end{bmatrix},$$

$$U = \begin{bmatrix} U_{k-1} & p \\ 0 & u_{kk} \end{bmatrix}.$$

De $A = LU$, segue que

$$\begin{bmatrix} L_{k-1}U_{k-1} & L_{k-1}p \\ mU_{k-1} & mp+u_{kk} \end{bmatrix}.$$

Agora, pela hipótese de indução, L_{k-1} e U_{k-1} são unicamente determinados e $L_{k-1}U_{k-1} = A_{k-1}$. Assim $A = LU$ é equivalente a $L_{k-1}p = x$; $mU_{k-1} = y$ e $mp+u_{kk} = a_{kk}$, ou seja, $p = L_{k-1}^{-1}x$; $m = yU_{k-1}^{-1}$ e $u_{kk} = a_{kk} - mp$. Então p, m e u_{kk} são determinados univocamente nesta ordem, e L e U são determinados unicamente.

Por fim, temos o determinante da matriz A:

$$\begin{aligned} det(A) = det(L).det(U) &= 1.det(U_{k-1}).u_{kk} \\ &= u_{11}u_{22}...u_{k-1,k-1}u_{kk} \end{aligned}$$

Resolvendo o sistema $LUx = b$

Após fatorarmos a matriz dos coeficientes A, no produto de duas matrizes triangulares LU, obtemos $A = LU$. Assim

$$Ax = b \qquad \Rightarrow \qquad LUx = b.$$

O novo sistema é dividido e resolvido na seguinte ordem:

$$Ly = b \qquad \text{e} \qquad Ux = y$$

Exemplo 2.2. *Resolva o sistema linear abaixo, utilizando o Método de Fatoração LU.*

$$\begin{cases} 2x_1 + 8x_2 + 7x_3 = 4 \\ 3x_1 + 9x_2 + x_3 = 2 \\ 5x_1 + 4x_2 + 6x_3 = 5 \end{cases}$$

Iremos aplicar o mesmo procedimento da Eliminação Gaussiana para transformar A em LU. Usaremos a tabela abaixo para organizar o passo a passo.

Tabela 2 – Fatoração LU do sistema linear do exemplo 2.2

(k)	l	L			A			b	Operação
	1				2	8	7	4	
1	2	m_{21}	=	3/2	3	9	1	2	$l_2 - m_{21}l_1$
	3	m_{31}	=	5/2	5	4	6	5	$l_3 - m_{31}l_1$
	1				2	8	7		
2	2				0	-3	-9,5		
	3	m_{32}	=	16/3	0	-16	-11,5		$l_3 - m_{32}l_2$
	1	1	0	0	2	8	7		
	2	1,5	1	0	0	-3	-9,5		
	3	2,5	5,333	1	0	0	39,166		

Fonte: elaborada pelo autor

Definidos L e U, iremos resolver o sistema $Ly = b$ e depois $Ux = y$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1,5 & 1 & 0 \\ 2,5 & 5,333 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 5 \end{bmatrix}$$

Aplicando a substituição progressiva no sistema, já que L é triangular inferior, a solução é

$$y = \begin{bmatrix} 4 \\ -4 \\ 16,333 \end{bmatrix}.$$

Encontrado y o substituiremos no próximo sistema:

$$\begin{bmatrix} 2 & 8 & 7 \\ 0 & -3 & -9,5 \\ 0 & 0 & 39,166 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \\ 16,333 \end{bmatrix}.$$

O sistema é resolvido por substituição retroativa, à qual obtemos o vetor solução

$$x = \begin{bmatrix} 0,489 \\ 0,012 \\ 0,417 \end{bmatrix}.$$

Estratégia do Pivoteamento

A permutação de linhas no método de Fatoração LU tem o mesmo objetivo que na eliminação de Gauss, ou seja, evitar a incidência de pivôs nulos, tornando o cálculo possível para a maioria dos sistemas lineares.

A troca de linhas se dá pela operação feita sobre A que associada a uma matriz identidade, dá origem à **matriz de permutação** P. Vejamos o seguinte exemplo:

Exemplo 2.3. *Encontre a matriz P associada à troca de linhas de A*

$$A = \begin{bmatrix} 2 & 4 & 5 \\ 1 & 9 & 3 \\ 5 & 1 & 2 \end{bmatrix} \quad sendo \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Iremos efetuar a troca de linhas em A e I simultaneamente, observe abaixo:

$$IA = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 5 \\ 1 & 9 & 3 \\ 5 & 1 & 2 \end{bmatrix} \begin{matrix} l_1^{(0)} \leftrightarrow l_3^{(0)} \\ l_2^{(0)} \leftrightarrow l_3^{(1)} \end{matrix}$$

$$\Rightarrow \quad PA = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 1 & 2 \\ 2 & 4 & 5 \\ 1 & 9 & 3 \end{bmatrix}.$$

Assim, a matriz de permutação associada à troca de linhas de A é

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Para resolver um sistema $Ax = b$ pelo método de Fatoração LU com pivoteamento parcial, procedemos da seguinte forma:

$$\begin{aligned} PA &= LU \\ PAx &= Pb \\ LUx &= Pb, \end{aligned}$$

logo,

$$Ly = Pb \quad \Rightarrow \quad y = Ux$$

Exemplo 2.4. *Resolva o sistema linear abaixo, utilizando o Método de Fatoração LU.*

$$\begin{cases} 5x_1 - 2x_2 + 4x_3 + x_4 = 4 \\ 2x_1 + 3x_3 + 7x_4 = 9 \\ -8x_1 + x_2 + 5x_3 + 3x_4 = 3 \\ 6x_1 + 2x_2 - x_3 + 9x_4 = 2 \end{cases}$$

Para resolver o sistema devemos efetuar a troca de linhas na matriz A, já que a segunda linha possui o elemento da diagonal igual a zero. Com isso faremos a permutação de linhas em A com a matriz I associada da seguinte forma:

$$IA = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -2 & 4 & 1 \\ 2 & 0 & 3 & 7 \\ -8 & 1 & 5 & 3 \\ 6 & 2 & -1 & 9 \end{bmatrix}$$

$$PA = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -8 & 1 & 5 & 3 \\ 5 & -2 & 4 & 1 \\ 2 & 0 & 3 & 7 \\ 6 & 2 & -1 & 9 \end{bmatrix} \begin{matrix} l_1 \leftrightarrow l_3 \\ l_2 \leftrightarrow l_3^{(1)} \end{matrix}$$

Com as linhas permutadas seguimos ao passo da eliminação, o qual é feito na seguinte tabela:

Tabela 3 – Fatoração LU do sistema linear do exemplo 2.4

(k)	l	L				**A**				Operação
	1					-8	1	5	3	
1	2	m_{21}	=	-5/8		5	-2	4	1	$l_2 - m_{21}l_1$
	3	m_{31}	=	-2/8		2	0	3	7	$l_3 - m_{31}l_1$
	4	m_{41}	=	-6/8		6	2	-1	9	$l_4 - m_{41}l_1$
	2					0	-1,375	7,125	2,875	
2	3	m_{32}	=	$\frac{-0.25}{1.375}$		0	0,25	4,25	7,75	$l_3 - m_{32}l_2$
	4	m_{42}	=	$\frac{-2.75}{1.375}$		0	2,75	2,75	11,25	$l_4 - m_{42}l_2$
3	3					0	0	5,5455	8,2727	
	4	m_{43}	=	$\frac{17}{5.5455}$		0	0	17	17	$l_4 - m_{43}l_3$
	4					0	0	0	-8,3607	
		1	0	0	0	-8	1	5	3	
		-0,625	1	0	0	0	-1,375	7,125	2,875	
		-0,25	-0,1818	1	0	0	0	5,5455	8,2727	
		-0,75	-2	3,0656	1	0	0	0	-8,3607	

Fonte: elaborada pelo autor

Com as matrizes L e U definidas, resolvemos os sistemas que cada uma compõe:

$$Ly = Pb \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0,625 & 1 & 0 & 0 \\ -0,25 & -0,1818 & 1 & 0 \\ -0,75 & -2 & 3,0656 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 9 \\ 2 \end{bmatrix}$$

$$\Rightarrow \quad y = \begin{bmatrix} 3 \\ 5,875 \\ 10,8182 \\ -17,1639 \end{bmatrix}$$

$$Ux = y \Rightarrow \begin{bmatrix} -8 & 1 & 5 & 3 \\ 0 & -1,375 & 7,125 & 2,875 \\ 0 & 0 & 5,5455 & 8,2727 \\ 0 & 0 & 0 & -8,3607 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 5,875 \\ 10,8182 \\ -17,1639 \end{bmatrix}$$

$$\Rightarrow \quad x = \begin{bmatrix} -1,0176 \\ -5,7412 \\ -1,1118 \\ 2,0529 \end{bmatrix}$$

2.1.3 Mal Condicionamento

Considere o seguinte sistema de equações:

$$\begin{cases} x_1 + x_2 = 1 \\ 9.87x_1 + 10x_2 = 10 \end{cases}$$

Ao ser resolvido pelo método de Eliminação Gaussiana, são obtidos os valores $x_1 = 0$ e $x_2 = 1$. Já o sistema

$$\begin{cases} x_1 + x_2 = 1 \\ 9,98x_1 + 9,82x_2 = 9,5 \end{cases},$$

onde fizemos pequenas alterações nos coeficientes, os resultados encontrados foram $x_1 = -2$ e $x_2 = 3$, ou seja, resultados muito diferentes em relação ao primeiro sistema, já que a mudança nos coeficientes foi mínima. Este é um pequeno exemplo de sistema mal condicionado.

Uma maneira de saber se um sistema é mal condicionado é determinando o número de condição da matriz dos coeficientes, definido pelo produto da norma da matriz[1] e sua inversa.

$$\gamma(A) = \|A\| \left\|A^{-1}\right\| \tag{2.8}$$

Tomando $\widetilde{A}$ como uma aproximação da matriz exata A, seja $Ax = b$ e $\widetilde{A}y = b$ (com y sendo solução aproximada), observamos que

$$x = A^{-1}b = A^{-1}\widetilde{A}y = A^{-1}(A + \widetilde{A} - A)y = A^{-1}Ay + A^{-1}(\widetilde{A} - A)y = y + A^{-1}(\widetilde{A} - A)y$$

Fazendo $\widetilde{A} - A = \delta A$, teremos

$$x - y = A^{-1}\delta A y$$

Da propriedade de normas de produto de matrizes, sabemos que $\|Ax\| \leq \|A\| \|x\|$ e $\|AB\| \leq \|A\| \|B\|$.

Ao usarmos duas vezes a propriedade no último termo desenvolvido acima, veremos que

$$\|x - y\| \leq \|A^{-1}\| \|\delta A\| \|y\| = \|A\| \|A^{-1}\| \frac{\|\delta A\|}{\|A\|} \|y\|.$$

[1] Norma euclidiana de matrizes: $\|A\|_E = \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}$

De 2.8 sabemos que $\gamma(A) = \|A\| \left\|A^{-1}\right\|$, nos dando a possibilidade reescrever a desigualdade da seguinte forma:

$$\frac{\|x-y\|}{\|y\|} \leq \gamma(A) \frac{\|\delta A\|}{\|A\|},$$

nos levando a observar que esta desigualdade estabelece uma delimitação para o erro relativo da solução.

Caso o vetor independente b contenha erro, a desigualdade poderá ter a forma (CUNHA, 2000)

$$\frac{\|x--y\|}{\|x\|} \leq \frac{cond(A)}{1-cond(A)\frac{\|\delta A\|}{\|A\|}} \left\{ \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right\},$$

com δb representando o erro absoluto do vetor b.

Algumas observações (FRANCO, 2006):

- Se o número de condição $\gamma(A)$ for grande, pequenos arredondamentos em A acarretarão grandes alterações no resultado x, tornando $Ax = b$ mal condicionado.
- O número de condição é considerado grande quando o resultado for maior que 10^4.

Nos exemplos 2.1 e 2.2, os números de condição $\gamma(A)$ são, respectivamente, $\gamma(A) = 22,3565$ e $\gamma(A) = 5,5899$.

Exemplo 2.5. *A matriz de Hilbert (David Hilbert, 1862-1943) é um exemplo de matriz mal condicionada, seus elementos são definidos por*

$$H_{ij} = \frac{1}{i+j-1}, \qquad 1 \leq i, j \leq n.$$

Fazendo n = 15 a matriz de Hilbert tem um número de condição aproximadamente 3.9627×10^{17}. Com um número de condição tão grande torna-se difícil resolver numericamente um sistema com esta matriz, pois influencia drasticamente na precisão da máquina.

No Matlab podemos calcular o número de condição da matriz de Hilbert com a seguinte rotina:

Código 2.1 – Matriz de Hilbert

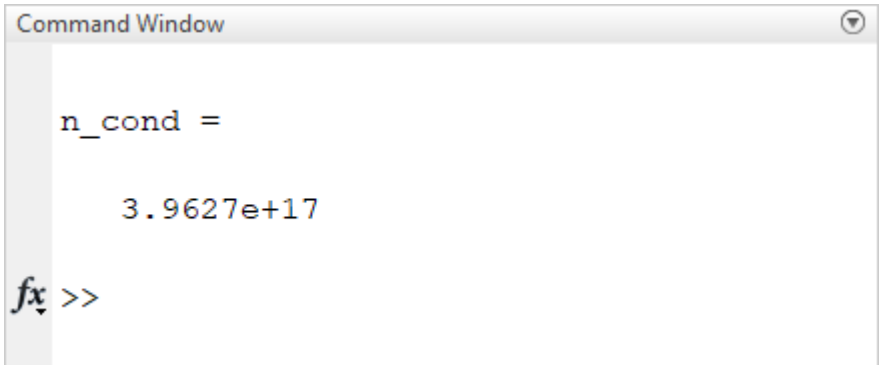
```
1 clear all                    % Limpa a memoria
2 clc                          % Limpa a tela
3
4 n = 15;                      % Numero de dimensoes da matriz
5
```

```
6  for i=1:n
7      for j=1:n
8          h(i,j)=1/(i+j-1);    % Matriz de Hilbert
9      end
10 end
11
12 n_cond = cond(h)              % Numero condicao da matriz h.
```

Fonte: elaborado pelo autor

O resultado após a execução da rotina é:

Figura 5 – Número condição da matriz de Hilbert com $n = 15$



Fonte: elaborada pelo autor

2.2 Métodos Iterativos

A partir desta seção começamos a apresentar os outros tipos de métodos que compõe este trabalho, os iterativos. O funcionamento básico de tais métodos consiste em dado um vetor inicial x^0, são geradas sequências de vetores aproximações $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$.

Para a aplicação no Matlab usaremos notação matricial na forma geral dos métodos iterativos:

$$x = Rx + s, \tag{2.9}$$

ou ainda,

$$x^{(k+1)} = Rx^{(k)} + s,$$

onde R é chamada matriz de iteração e s o vetor de iteração. Nesta última equação, temos representada a iteração em si.

Convergência dos Métodos Iterativos

Tendo em vista que os métodos iterativos nos dão soluções aproximadas após um número finito de iterações, é preciso que tenhamos garantia de que estas convirjam, ou não, para a solução.

O teorema abaixo estabelece a condição necessária e suficiente para a convergência dos métodos de Jacobi e Gauss-Seidel.

Teorema 2.3. *Se os autovalores da matriz de iteração satisfazem* $\rho(R) = \max\limits_{1\leq i\leq n} |\lambda_i| < 1$ *o método iterativo* $\boldsymbol{x}^{(k+1)} = Rx^{(k)} + s$ *converge, independente do vetor inicial* $x^{(0)}$.

Suponha que R, não simétrica nem positiva definida, possui n autovetores $x_1, x_2, ..., x_n$ com autovalores $\lambda_1, \lambda_2, ..., \lambda_n$. O vetor erro $E^{(0)}$ pode, portanto, ser representado como combinação linear dos autovetores

$$E^{(0)} = \sum_{i=1}^{n} s_i x_i.$$

Para o m-ésimo vetor de erro, por recorrência temos que

$$E^{(m)} = \sum_{i=1}^{n} s_i \lambda_i^m x_i \qquad m = 1, 2, ... \tag{2.10}$$

i) Condição necessária. Se $\lim\limits_{m\to\infty} E^{(m)} = 0$, para todo vetor inicial $x^{(0)}$, pela equação 2.10 o raio espectral $\rho(R) = \max\limits_{1\leq i\leq n} |\lambda_i|$ deve necessariamente ser inferior a 1.

ii) Condição suficiente. Se $\rho(R) < 1$, a convergência de $E^{(m)}$ em relação ao vetor inicial segue da equação 2.10 para todo vetor $E^{(0)}$.

Observação: Essa demonstração foi baseada em Schwarz, Rutishauser e Stiefel (1973, p.47).

No Matlab os autovalores de R são calculados pelo comando `eig(R)`. As duas linhas de código abaixo são as mesmas usadas nos códigos dos métodos iterativos que serão vistos nas 3 subseções seguintes.

Código 2.2 – Autovalor

```
1
2 at = eig(R);          % Determina os autovalores da matriz de iteracao
3 raio = norm(at,inf);  % Seleciona o raio espectral (maior autovalor)
```

Fonte: elaborado pelo autor

O teorema 2.3 estabelece condição necessária e suficiente, porém o custo para calcular os autovalores da matriz de iteração é alto, mais trabalhoso que a própria solução do sistema (CUNHA, 2000).

Neste trabalho temos o apoio computacional do MATLAB para determinação dos autovalores. Para os casos em que não há essa possibilidade existe outros critérios que estabelecem condições apenas suficientes para convergência dos métodos. Os teoremas a seguir apresentam tais condições.

Teorema 2.4. *(Critério das linhas) O método iterativo de Jacobi gera uma sequência $x^{(k)}$ que converge para a solução do sistema quando*

$$\xi_i = \frac{1}{|a_{ii}|}\left(\sum_{j\neq i}|a_{ij}|\right) < 1.$$

Em palavras o método converge quando o quociente, da soma dos coeficientes da linha k pelo elemento da diagonal principal, for menor que 1.

A demonstração pode ser encontrada em Ruggiero e Lopes (1996).

Teorema 2.5. *(Critério de Sassenfeld) Seja A uma matriz quadrada de ordem n. Sejam β_i, dados por:*

$$\beta_i = \frac{\sum_{j=1}^{i-1}|a_{ij}|\beta_j + \sum_{j=i+1}^{n}|a_{ij}|}{|a_{ii}|}$$

Se $\beta = \max_{1\leq i\leq n}\beta_j < 1$, então o método de Gauss-Seidel gera uma sequência de vetores $x^{(k)}$ convergente qualquer que seja a aproximação $x^{(0)}$. Além do mais, quanto menor for β mais rápida será a convergência do método.

A demonstração pode ser encontrada em Arenales e Darezzo (2008).

2.2.1 Método de Jacobi

O *método de Jacobi* utiliza as equações do tipo

$$\begin{aligned} x_1 &= \left(b_1 - \sum_{j=2}^{n} a_{1j}x_j\right)/a_{11}, \\ &\vdots \\ x_i &= \left(b_i - \sum_{j\neq i}^{n} a_{ij}x_j\right)/a_{ii}, \\ &\vdots \\ x_n &= \left(b_n - \sum_{j=1}^{n-1} a_{nj}x_j\right)/a_{nn} \end{aligned} \tag{2.11}$$

para gerar as sequências de vetores aproximações, a partir de uma solução inicial $x^{(0)}$. Tal método recebe este nome devido ao seu criador Carl Gustav Jacobi (1804-1851), que o apresentou em 1845 (CUNHA, 2000).

Usando a notação matricial para definir o método de Jacobi, teremos a matriz A decomposta em $A = D - L - U$, onde:

- $-L$: Matriz triangular estritamente inferior $-a_{ij}$ se $i < j$ e $a_{ij} = 0$ se $i \leq j$;
- D : Matriz diagonal a_{ii};
- $-U$: Matriz triangular estritamente superior $-a_{ij}$ se $i < j$ e $a_{ij} = 0$ se $i \geq j$

Da equação $Ax = b$ teremos $(D - L - U)x = b$ que será transformada em

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

Assim o método iterativo de Jacobi estará representado na fórmula geral de 2.9 onde $R_j = D^{-1}(L+U)$ e $s_j = D^{-1}b$.

Quando se há garantia de que o método iterativo irá convergir (teorema 2.3), a sequência de vetores solução $x^{(k)}$ se aproxima da solução x do sistema:

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

Como critério de parada do método usaremos o erro absoluto. Assim, as iterações prosseguem até que o erro absoluto seja menor ou igual que a precisão ε dada, ou seja,

$$\|x^{(k)} - x^{(k-1)}\| \leq \varepsilon.$$

No exemplo abaixo iremos aplicar o que foi visto do método de jacobi, bem como verificar se o sistema atende ao critério de convergência visto anteriormente.

Exemplo 2.6. *Considerando o sistema linear*

$$\begin{cases} 11.8x_1 + 7x_2 + 3.2x_3 - x_4 = 6.65 \\ 5x_1 - 10.7x_2 - 2x_3 + 3x_4 = -8 \\ 1.2x_1 + 4.8x_2 + 9x_3 + 2.3x_4 = 5 \\ -5x_1 + 2x_2 + 3.5x_3 + 8.75x_4 = 3 \end{cases}$$

resolva utilizando o método de Jacobi com solução inicial $x^{(0)} = (0,0,0,0)^t$ *e precisão* $\varepsilon = 0.0002$.

Utilizando a rotina programada para o método de Jacobi, o raio espectral é calculado:

$$\rho(R) = 0.4432 < 1,$$

assim, a convergência é garantida.

Prosseguindo com a solução do sistema pelo método, armazenaremos os vetores obtidos por cada iteração na tabela abaixo, onde cada linha corresponde a um vetor.

Tabela 4 – Iterações do exemplo 2.6

Iterações	x_1	x_2	x_3	x_4	Erro absoluto
0	0	0	0	0	
1	0,5636	0,7477	0,5556	0,3429	1
2	-0,0016	1,0033	-0,0060	0,2718	0,2234
3	-0,0070	0,8242	-0,0488	0,1150	0,2352
4	0,0976	0,7858	0,0875	0,1700	0,0019
5	0,0881	0,8246	0,0800	0,1840	0,0455
6	0,0683	0,8255	0,0570	0,1727	0,0054
7	0,0731	0,8173	0,0621	0,1704	0,0090
8	0,0763	0,8180	0,0663	0,1730	0,0021
9	0,0750	0,8194	0,0649	0,1730	0,0014
10	0,0745	0,8191	0,0643	0,1724	0,0006
11	0,0748	0,8188	0,0647	0,1725	0,0002
12	0,0749	0,8189	0,0648	0,1726	0,0001
Sol. exata	0,0748	0,8189	0,0647	0,1725	

Fonte: elaborada pelo autor

Com a aproximação inicial $x^{(0)} = (0,0,0,0)^t$ e precisão $\varepsilon = 0.0002$ o método convergiu na 12ª iteração, quando atinge a precisão desejada.

2.2.2 Método de Gauss-Seidel

O método de Gauss-Seidel pode ser visto como um melhoramento do método de Jacobi, no que diz respeito à velocidade de convergência, já que dado um vetor inicial $x^{(0)}$ calcula-se a primeira iteração da seguinte forma:

$$\begin{aligned}
x_1^{(1)} &= \tfrac{1}{a_{11}}(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} \cdots - a_{1n}x_n^{(0)}) \\
x_2^{(1)} &= \tfrac{1}{a_{22}}(b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)} \cdots - a_{2n}x_n^{(0)}) \\
x_3^{(1)} &= \tfrac{1}{a_{33}}(b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} \cdots - a_{3n}x_n^{(0)}) \\
&\vdots \quad \vdots \\
x_n^{(1)} &= \tfrac{1}{a_{nn}}(b_n - a_{n1}x_1^{(1)} - a_{n2}x_2^{(1)} \cdots - a_{n,n-1}x_{n-1}^{(1)})
\end{aligned}$$

Este pequeno exemplo ilustra a primeira iteração do método. Em palavras, o método de Gauss-Seidel utiliza os valores atualizados da própria iteração para o cálculo das demais variáveis.

Utilizando a mesma notação matricial de 2.9, podemos representar o método da mesma forma, fazendo $A = D - L - U$:

$$(D-L-U)x = b$$

$$\Rightarrow (D-L)x = b + Ux$$

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

Sendo $R_{gs} = (D-L)^{-1}U$ e $s_{gs} = (D-L)^{-1}b$

Tendo-se garantida a convergência do sistema, pela condição estabelecida (teorema2.3), a sequência de vetores solução $x^{(k)}$ se aproxima da solução x do sistema:

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

As iterações prosseguem até que o erro seja menor ou igual que a precisão ε dada, ou seja,

$$\|x^{(k)} - x^{(k-1)}\| \leq \varepsilon.$$

No exemplo abaixo iremos aplicar o que foi visto acima, bem como verificar se o sistema atende ao critério de convergência visto anteriormente.

Exemplo 2.7. *Considerando o sistema linear visto no exemplo 2.6, resolva-o utilizando o método de Gauss Seidel com solução inicial* $x^{(0)} = (0,0,0,0)^t$ *e precisão* $\varepsilon = 0.0002$.

O cálculo do raio espectral é efetuado e obtemos:

$$\rho(R) = 0.3045 < 1,$$

assim, de acordo com o teorema 2.3, a convergência é garantida.

Prosseguindo com a solução do sistema pelo método, armazenaremos os vetores obtidos por cada iteração na tabela abaixo, onde cada linha corresponde a um vetor.

Tabela 5 – Iterações do exemplo 2.7

Iterações	x_1	x_2	x_3	x_4	Erro absoluto
0	0	0	0	0	
1	0,5636	1,011	-0,0588	0,4573	1
2	0,0185	0,8955	-0,0414	0,1653	0.1955
3	0,0576	0,8286	0,0637	0,1609	0,0656
4	0,0684	0,8128	0,0718	0,1674	0,0150
5	0,0761	0,8167	0,067	0,1728	0,0062
6	0,0755	0,8189	0,0646	0,173	0,0022
7	0,0749	0,8191	0,0645	0,1727	0,0001
Sol. exata	0,0748	0,8189	0,0647	0,1725	

Fonte: elaborada pelo autor

Na tabela acima registramos em cada linha as respectivas iterações obtidas através da rotina programada no *software*. Na última linha podemos ver que chegamos a solução do sistema com um grau de exatidão satisfatório, já que a tolerância estabelecida era de $\varepsilon = 0.0002$ para o erro.

2.2.3 Método de Sobre-Relaxação Sucessiva (SRS)

O método Sobre-Relaxação Sucessiva, do inglês *Successive Over-Relaxation (SOR)*, consiste num melhoramento do método iterativo de Gauss-Seidel, de forma a acelerar a convergência.

Este método consiste em utilizarmos um parâmetro ω de sobre-relaxação nas iterações que passam a ter a forma

$$\mathbf{x}^{(k+1)} = (1-\omega)\mathbf{x}^{(k)} + \omega\mathbf{x}^{(k+1)},$$

ou seja,

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

Para reescrever o método em notação matricial, multiplicaremos por ω os dois lados da equação $(D-L-U)x = b$, onde

$$\omega(D-L-U)x = \omega b.$$

Ao fazermos a soma do vetor nulo $(D-D)x$ no primeiro termo, teremos

$$(D-D)x + \omega(D-L-U)x = \omega b$$

$$\Rightarrow (D-\omega L)x = [(1-\omega)D + \omega U]x + \omega b$$

$$\Rightarrow x^{(k+1)} = (D-\omega L)^{-1}[(1-\omega)D + \omega U]x + (D-\omega L)^{-1}\omega b$$

Assim o método SOR estará representado na forma geral

$$x^{(k+1)} = R_{srs}x^{(k)} + s_{srs},$$

onde $R_{srs} = (D-\omega L)^{(-1)}[(1-\omega)D + \omega U]$ e $s_{srs} = (D-\omega L)^{-1}\omega b$.

Como dito no início desta subseção, este método é um melhoramento ou aceleração da convergência do método de Gauss-Seidel. Mas para que isso ocorra o parâmetro ω deve estar bem definido. É aí que entra a dificuldade em utilizar a *SRS*, ou seja, definir o melhor valor possível de ω para que o método convirja.

Para isso os seguintes teoremas nos ajudam, em grande parte dos casos, a determinar o valor do parâmetro de sobre-relaxação.

Teorema 2.6. *Se $a_{ii} \neq 0$ para cada $i = 1, 2, ..., n$, então $\rho(R_{srs}) \geq |\omega - 1|$. O que significa que o método SRS converge somente se $0 < \omega < 2$.*

Sejam $\lambda_1, \ldots, \lambda_n$ os autovalores de R_{srs}. Então

$$\begin{aligned}\prod_{i=1}^{n} \lambda_i &= detR_{srs} = det\left((D-\omega L)^{-1}[(1-\omega)D+\omega U]\right) \\ &= det(D-\omega L)^{-1} det\left((1-\omega)D+\omega U\right) = det(D^{-1}) det\left((1-\omega)D\right) \\ &= \left(\frac{1}{a_{11}a_{22}...a_{nn}}\right)[(1-\omega)^n(a_{11}a_{22}...a_{nn})] = (1-\omega)^n\end{aligned}$$

Portanto

$$\rho(R_{srs}) = \max_{1 \leq i \leq n} |\lambda_i| \geq |\omega - 1|$$

e $|\omega - 1| < 1$ se e só se $0 < \omega < 2$.

Quando $\omega = 1$ teremos o próprio método de Gauss-Seidel. É importante ressaltar que a escolha $1 < \omega < 2$ define os métodos de sobre-relaxação e quando $0 < \omega < 1$ caracterizam-se os métodos de sub-relaxação (CUNHA, 2000).

No exemplo abaixo iremos resolver o sistema linear atribuindo ao parâmetro ω valores que estejam no intervalo determinado pelo teorema 2.6.

Exemplo 2.8. *Considerando o sistema linear*

$$\begin{cases} 12x_1 + 3x_2 + x_3 + 5x_4 + x_5 + x_6 = 36 \\ 10x_2 + x_3 + 4x_4 - 5x_5 = -34 \\ x_1 + 5x_2 - 15x_3 + 3x_4 + 6x_5 + x_6 = 58 \\ 2x_1 + 6x_2 + x_3 + 9x_4 + x_6 = 14 \\ 8x_1 + 4x_2 - 5x_3 + x_4 + 12x_5 + x_6 = 74 \\ 5x_1 + x_2 + 3x_3 + 7x_5 + 10x_6 = 90 \end{cases}$$

resolva utilizando o método SRS com solução inicial $x^{(0)} = (0,0,0,0,0,0)^t$ e precisão $\varepsilon =$ 0.00001.

Na tabela abaixo estão registrados os dados referentes às iterações do método, de acordo com os valores atribuídos à ω.

Tabela 6 – Iterações do exemplo 2.8

ω	0,2	0,4	0,6	1	1,2	1,4	1,6	1,8
Iterações	71	33	19	10	17	54	>600	>600
$\rho(R)$	0,9034	0,793	0,6634	0,2833	0,4984	0,8497	1,2124	1,5884

Fonte: elaborada pelo autor

Ao analisar as primeiras 4 colunas de números da tabela, pode-se perceber que à medida em que os valores do parâmetro ω aumentam, as iterações diminuem e o raio espectral $\rho(R)$

também. Quando os valores do parâmetro são maiores que 1, as iterações aumentam e o raio também. Observando às duas últimas colunas, vemos que o método não converge quando $\omega = 1,6$ e $\omega = 1,8$, já que $\rho(R) > 1$ contrariando o teorema 2.3.

Como citado anteriormente, quando $\omega = 1$ temos o próprio método de Gauss-Seidel. Assim, como pode ser observado na tabela 6, o sistema converge mais rapidamente por Gauss-Seidel do que pelo método de SRS.

2.2.4 Método Gradiente

Nesta seção trataremos do Método Gradiente, porém este não será implementado no *software*, já que optamos por trabalhar com o método dos Gradientes Conjugados, tido como mais eficiente que o primeiro. Mas para uma melhor compreensão do leitor iniciaremos aqui descrevendo o Método Gradiente.

Assim sendo, dado um sistema linear $Ax = b$ iremos admitir que a matriz seja simétrica ($A^t = A$) e positiva definida, onde $x^tAx > 0$, para todo $x \neq 0$. Adotaremos ainda, a notação $(x,y) = x^t, y$ que diz respeito ao produto interior de vetores n-dimensionais.

O método gradiente tem como ideia básica minimizar a função de

$$F(x) = \frac{1}{2}(Ax,x) + (b,x). \tag{2.12}$$

O mínimo da função é atingido quando seu gradiente

$$\nabla F(x) = b - Ax \tag{2.13}$$

é zero, isto é,

$$b - Ax = r$$

sendo r o vetor residual.

Teorema 2.7. *Determinar a solução do sistema linear $Ax = b$, onde A é simétrica positiva definida, é equivalente a determinar o ponto de mínimo de 2.12.*

Seja $x' = (x_1, x_2, ..., x_n)^t$ o ponto em que todas derivadas parciais de F se anulam e $(x_1, x_2, ..., x_n)^t$ é solução de $Ax - b = 0$, então $\nabla F = 0 \Rightarrow r = 0 \Rightarrow x'$ é a solução de $b - Ax = 0$.

Temos que *x'* é ponto estacionário de F se e somente se $\nabla F(x') = 0$, ou seja,

$$\sum_{j=1}^{n} b_i - a_{ij}x'_j = 0.$$

O ponto estacionário é único, pois o sistema admite uma única solução. Como:

$$\frac{\partial^2 F(x')}{\partial x_1^2} = a_{11}, \quad \frac{\partial^2 F(x')}{\partial x_1 x_2} = a_{12}, \quad \ldots, \quad \frac{\partial^2 F(x')}{\partial x_i x_j} = a_{ij},$$

temos que:

$$A = a_{ij} = \frac{\partial^2 F(x')}{\partial x_i x_j}.$$

Por hipótese, A é definida positiva. Assim, x é ponto de mínimo.

O vetor gradiente determina a direção de crescimento máximo da função. Assim, para alcançarmos o mínimo usaremos a direção contrária:

$$x^{(k+1)} = x^{(0)} + s_k p^k, \tag{2.14}$$

sendo s_k o parâmetro que regula o tamanho do passo, de busca do mínimo, na k-ésima iteração. E p a direção em que devemos procurar o mínimo de F em 2.12, definido por $p^k = -r^{k-1}$.

$$\begin{aligned} F(x^{(k+1)}) &= \tfrac{1}{2}(Ax^{(k+1)}, x^{(k+1)}) + (b, x^{(k+1)}) \\ &= \tfrac{1}{2}(A(x^{(0)} + s_k p), x^{(0)} + s_k p) + (b, x^{(0)} + s_k p) \\ &= \tfrac{1}{2}[(Ax^{(0)}, x^{(0)}) + 2s_k(Ax^{(0)}, p) + s_k^2(Ap, p) + 2(b, x^{(0)}) + 2s_k(b, p)] \\ &= F(x^{(0)}) + \tfrac{s_k^2}{2}(Ap, p) + s(b - Ax^{(0)}, p) \end{aligned}$$

sendo que $\frac{1}{2}[(Ax^{(0)}, x^{(0)}) + 2s_k(b, x^{(0)})] = F(x^{(0)})$. Logo,

$$F(x^{(k+1)}) = F(x^{(0)}) + \frac{s_k^2}{2}(Ap, p) + s_k(r, p).$$

A condição necessária para que s_k minimize F é

$$\frac{\partial F(x^{(k+1)})}{\partial s} = s(Ap, p) + (r, p) = 0 \Rightarrow s = -\frac{(r, p)}{(Ap, p)},$$

que é ponto estacionário de F. Além do mais, $\dfrac{\partial^2 F(x^{(k+1)})}{\partial s^2} = (Ap, p) > 0$, já que A é positiva definida e, assim, s é mínimo na direção p.

Logo,

$$s_{min} = -\frac{(r, p)}{(Ap, p)} = s = -\frac{(r^{(k-1)}, p^k)}{(Ap^k, p^k)} \Rightarrow s_{min} = \frac{(r^{(k-1)}, r^{(k-1)})}{(Ar^{(k-1)}, r^{(k-1)})}.$$

$$\Rightarrow s_{min} = \frac{(r^{(k-1)}, r^{(k-1)})}{(Ar^{(k-1)}, r^{(k-1)})}. \tag{2.15}$$

Assim, no método dos Gradientes, temos:

$$x^{(k)} = x^{(k-1)} + sp^k$$

$$x^{(k)} = x^{(k-1)} - s_{min} r^{(k-1)}. \tag{2.16}$$

Com isso

$$r^{(k)} = b - Ax^{(k)} \Rightarrow r^{(k)} = b - A(x^{(k-1)} - s_{min}r^{(k-1)})$$
$$r^{(k)} = b - Ax^{(k-1)} + s_{min}Ar^{(k-1)}$$

$$r^{(k)} = r^{(k-1)} + s_{min}Ar^{(k-1)} \tag{2.17}$$

Os métodos do tipo Gradiente são construídos por meio das equações 2.15, 2.16 e 2.17.

2.2.5 Método Gradiente Conjugado

O passo $k = 1$ no método dos Gradientes Conjugados é o mesmo do método dos Gradientes, ou seja, dado $x^{(0)}$, encontramos $r^{(0)} = b - Ax^{(0)}$ e:

$$p^{(1)} = -r^{(0)},$$
$$x^{(1)} = x^{(0)} - sr^{(0)},$$

onde,

$$s = q_1 = -\frac{(r^{(0)}, p^{(1)})}{(Ap^{(1)}, p^{(1)})} = \frac{(r^{(0)}, r^{(0)})}{(Ar^{(0)}, r^{(0)})}.$$

Assim,

$$x^{(1)} = x^{(0)} - \frac{(r^{(0)}, r^{(0)})}{(Ar^{(0)}, r^{(0)})}r^{(0)}. \tag{2.18}$$

As direções $p^{(k)}$ e $p^{(k-1)}$ serão usadas como direções conjugadas, isto é, $p^{(k)}$ será tal que:

$$(Ap^{(k)}, p^{(k-1)}) = (p^{(k)}, Ap^{(k-1)}) = 0.$$

Ao tomarmos $p^{(k)}$ como combinação linear de $r^{(k-1)}$ e $p^{(k-1)}$, com $r^{(k-1)}$ não nulo, podemos tomá-lo igual a -1.

Portanto,

$$p^{(k)} = -r^{(k-1)} + \alpha_{k-1}p^{(k-1)} \tag{2.19}$$

sendo α_{k-1} o coeficiente a ser determinado.

Assim sendo, para $k = 2, 3, ...$

$$(p^{(k)}, Ap^{(k-1)}) = 0$$
$$(-r^{(k-1)} + \alpha_{k-1}p^{(k-1)}, Ap^{(k-1)}) = 0$$
$$(-r^{(k-1)}, Ap^{(k-1)}) + \alpha_{k-1}(p^{(k-1)}, Ap^{(k-1)}) = 0$$

Ao isolarmos α_{k-1}, obtemos:

$$\alpha_{k-1} = \frac{(-r^{(k-1)}, Ap^{(k-1)})}{(p^{(k-1)}, Ap^{(k-1)})}, k = 2, 3, ... \tag{2.20}$$

Assim que identificamos a direção $p^{(k)}$, procuramos pelo ponto de mínimo.

De $x^{(k)} = x^{(k-1)} + sp^k$, temos:

$$x^k = x^{(k-1)} + q_k p^k, \tag{2.21}$$

sendo que

$$q_k = -\frac{(r^{(k-1)}, p^{(k)})}{Ap^{(k)}, p^{(k)}}$$

Do resíduo $r^{(k)} = b - Ax^{(k)}$, temos que

$$r^{(k)} = b - A(x^{(k-1)} + q_k p^k)$$

$$r^{(k)} = b - Ax^{(k-1)} + q_k A p^k$$

$$r^{(k)} = r^{(k-1)} + q_k A p^k \tag{2.22}$$

O método dos Gradientes Conjugados é construído por meio das equações 2.18, 2.19, 2.20, 2.21 e 2.22.

Para este método estaremos usando uma função já pronta do MATLAB. Assim, no exemplo abaixo e no próximo capítulo, será usada a função `cgs` para resolver sistemas via método dos Gradientes Conjugados.

Exemplo 2.9. *Considere o sistema linear $Ax = b$, onde A é simétrica positiva definida e composta por: diagonal principal $a_{ii} = 6$, $i = 1 : n$; diagonais secundárias $a_{21} = a_{n,n-1} = 3$, $a_{i,i+1} = a_{i-1,i} = 3$, $i = 2 : n-1$; diagonais afastadas $a_{i,i+l} = 1$, $i = 1 : n-l$ e $a_{i-l,i} = 1$, $i = l+1 : n$. O vetor b definido por: $b_1 = b_n = 1$; $b_i = -1$, $i = 2 : l$; $b_i = 0$, $i = l+1 : n-1$; $b_i = -1$, $i = n-l+1 : n-1$. Com a solução inicial sendo $x_i^{(0)} = 0$, $i = 1 : n$ e precisão $\varepsilon = 0.0002$, resolva o sistema pelo método do Gradiente Conjugado.*

Vamos fixar $l = 6$ e variar o tamanho de n entre 10 e 80, ao executar a função `cgs` do MATLAB, o programa resolve o sistema de equações pelo método do Gradiente Conjugado, com isso registramos na tabela abaixo a quantidade de iterações que o método usa para atingir a solução.

Tabela 7 – Iterações do exemplo 2.9

n	10	20	30	40	50	60	70	80
Iterações	5	10	13	17	20	26	28	32

Fonte: elaborada pelo autor

3 COMPARAÇÕES ENTRE OS MÉTODOS

Este capítulo é destinado às comparações entre os métodos numéricos. Estaremos analisando o tempo de convergência dos métodos no geral, quantidade de iterações serão analisadas nos métodos iterativos. O computador utilizado para os cálculos tem as seguintes configurações: Processador Intel Core i7 de 4ª Geração, 64 *Gigabytes* de memória RAM, 1 *Terabyte* de armazenamento e Windows 10 instalado.

Estaremos utilizando o comando `rand`, responsável por gerar matrizes de números aleatórios, nos dando a possibilidade de trabalhar com sistemas de grandes dimensões que seriam trabalhosos de serem criados manualmente.

Código 3.1 – Comando `rand`

```
1
2 n=;                       % Tamanho do sistema
3 A=rand(n)+n*eye(n);       % Matriz aleatoria nxn com diagonal dominante
4                           % O comando eye retorna uma matriz identidade
5 A=(A+A')/2;               % A se torna simetrica
6 b=rand(n,1);              % Vetor coluna de numeros aleatorios
```

Fonte: elaborado pelo autor

A utilização do comando `eye`, que gera uma matriz identidade, nos permitirá tornar a matriz A diagonal dominante através da soma de `n*eye(n)`.

O estudo para determinar a solução dos sistemas de equações lineares de pequeno porte teve seu ápice a partir do século XVIII. Com trabalhos de grandes matemáticos como o do suíço Gabriel Cramer (1704 - 1752), que mostrou a solução de sistemas de equações com o cálculo dos determinantes. Um século mais tarde o alemão Carl Friedrich Gauss publica o método de eliminação (SANTOS, 2007).

Como já citamos no início, os métodos diretos aqui trabalhados são a Eliminação de Gauss e a Fatoração LU. Na tabela abaixo registramos os tempos oriundos das soluções de sistemas de números aleatórios. Na primeira coluna estão expressos os tamanhos que n assume, nas demais colunas estão registrados os tempos de cada método, variando a cada linha conforme o sistema $n \times n$ aumenta de tamanho.

Tabela 8 – Tempo gasto: Métodos diretos

n	Eliminação Gaussiana	Elimação Gaussiana com Pivoteamento	LU	LU com Pivoteamento
1000	5,6150337	5,6859271	5,5979659	5,6481978
2000	63,236028	63,502294	62,859587	63,321445
3000	243,50061	244,27152	243,38440	243,92547
4000	246,62650	247,71509	245,89137	248,88075
5000	490,30256	497,20089	491,36025	496,69899
6000	870,19679	862,71813	864,74930	869,21927
7000	1408,1167	1399,3337	1411,8924	1410,6465

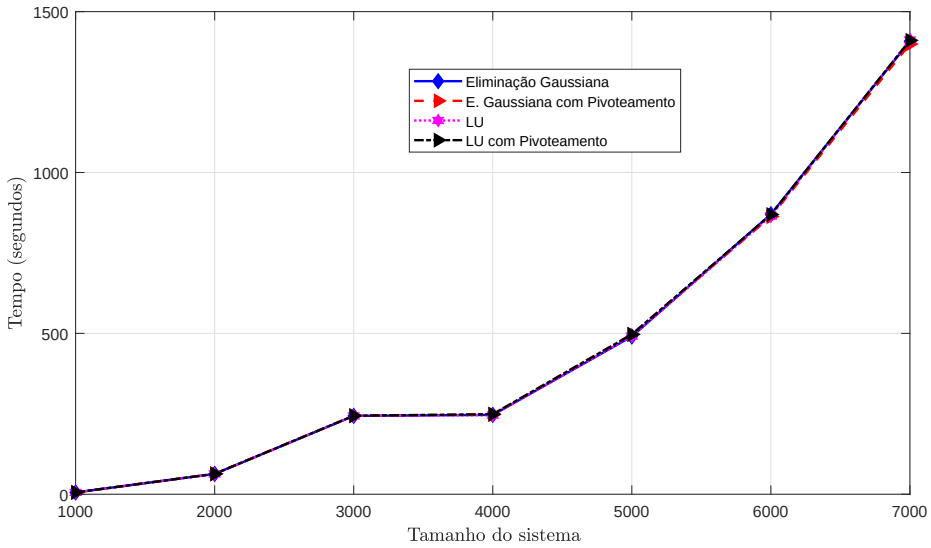
Fonte: elaborada pelo autor

As matrizes geradas aleatoriamente tem diagonal dominante, sendo assim as rotinas com pivoteamento parcial terão tempos parecidos com as rotinas que não tem. Ainda que o pivoteamento não seja necessário, nas colunas ”Eliminação Gaussiana com Pivoteamento” e ”LU com Pivoteamento” o tempo ainda é frações de segundos superior ao das rotinas sem pivoteamento, isso se deve ao fato de possuírem um laço de repetição a mais, responsável por procurar o maior elemento e permutar as linhas.

Observando os dados podemos perceber que a Fatoração LU sem pivoteamento é, frações de segundos, mais rápida que os demais. Isso se deve ao fato de que no processo de fatoração o vetor b não é alterado como no processo de eliminação de Gauss, sendo usado somente no momento da solução do sistema do primeiro $Ly = b$.

Na figura abaixo temos a representação dos gráficos dos métodos apresentados na tabela.

Figura 6 – Gráfico referente aos dados da tabela 8



Fonte: elaborada pelo autor

Pelo fato de os tempos serem bem próximos uns aos outros, os gráficos são também parecidos. É possível observarmos o crescimento dos mesmos à medida em que as dimensões do sistema aumentam, demandando mais tempo para a solução.

Os métodos iterativos surgiram da necessidade de acelerar o processo para encontrar a solução de sistemas sobre certas circunstâncias. Com o surgimento dos computadores a capacidade de resolver sistemas cada vez maiores ficou mais acessível e o desenvolvimento de métodos cada vez mais eficientes tornou-se necessário.

Para os métodos iterativos vistos no capítulo anterior, foram desenvolvidas três rotinas, sendo elas: Jacobi, Gauss Seidel e Sobre-Relaxação Sucessiva. Na tabela abaixo estão registrados os tempos das soluções calculadas pelas rotinas dos três métodos. Optamos por não adicionar a coluna referente aos tempos obtidos pelo método do Gradiente Conjugado, também iterativo, pelo fato de o mesmo ser uma função própria do MATLAB, tendo uma eficiência muito superior às rotinas aqui desenvolvidas.

Tabela 9 – Tempo gasto: Métodos iterativos

n	Jacobi	Iter. Jacobi	Gauss Seidel	Iter. G.S.	SRS	Iter. SRS
1000	0,01313	10	0,05065	5	0,13289	5
2000	0,03955	9	0,20364	5	0,46835	5
3000	0,07718	9	0,52694	5	1,40092	5
4000	0,13452	9	0,58294	5	2,20838	5
5000	0,20315	9	1,04541	5	3,52888	5
6000	0,33033	9	1,45368	5	5,37987	5
7000	0,40126	8	1,95276	5	8,01303	5

Fonte: elaborada pelo autor

No método de SRS o melhor valor encontrado para o parâmetro ω, após alguns testes, foi $\omega = 1.2$. Assim, os tempos registrados na coluna são oriundos de sistemas resolvidos nesse parâmetro.

Analisando os dados na tabela é perceptível a maior velocidade de convergência do método de Jacobi em relação ao de Gauss Seidel e muito superior ao de SRS na questão do tempo, o que contraria a teoria, já que os dois últimos deveriam alcançar mais rapidamente a solução. Pois a quantidade de iterações que Gauss Seidel e SRS levam para o cálculo, é menor que as de Jacobi. Nesse sentido, o método de Jacobi é mais rápido do que os outros dois métodos pois utilizamos o comando de vetorização do MATLAB (que elimina a necessidade de usar o comando `for`), enquanto que nas outras duas rotinas foi necessário usar o comando `for` o que acarretou em um tempo computacional maior.

Para verificar o quão próximos da solução estão os valores encontrados pelos métodos faremos o cálculo do erro absoluto, definido por

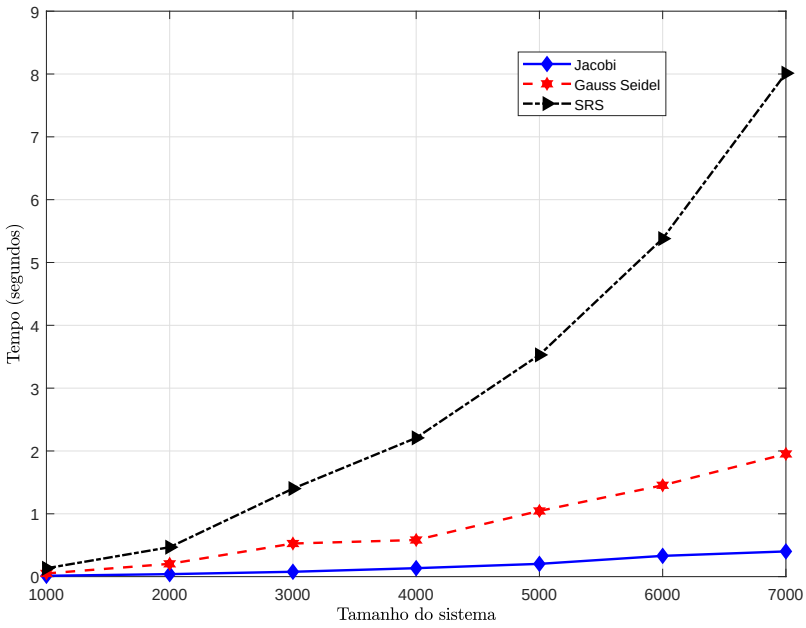
$$E = \|\overline{x} - x^{(k)}\|$$

sendo $\bar{x}$ o valor exato definido por $\bar{x}$ = `A\b`, e $x^{(k)}$ o valor aproximado alcançado pelos métodos iterativos.

Assim sendo, os métodos apresentam: Jacobi = 5×10^{-2}, Gauss Seidel = 3×10^{-6} e SRS = 2×10^{-5}.

Na figura abaixo estão representados os gráficos referentes aos tempos de cada método.

Figura 7 – Gráfico referente aos dados da tabela 9



Fonte: elaborada pelo autor

O *software* MATLAB apresenta algumas funções próprias para o cálculo da solução dos sistemas de equações lineares, dentre elas temos a própria divisão de b por A, definida por `x=A\b`. A função `linsolve` que utiliza a fatoração LU como método de solução. A função `lu`, que fatora a matriz A em LU nos dando a possibilidade de resolvermos os sistemas por `y=L\b` e `x=U\y`. Por fim, a função que resolve os sistemas pelo método do Gradiente Conjugado, a `cgs`.

Na tabela abaixo estão representadas funções próprias do MATLAB.

Tabela 10 – Tempo gasto: Funções MATLAB

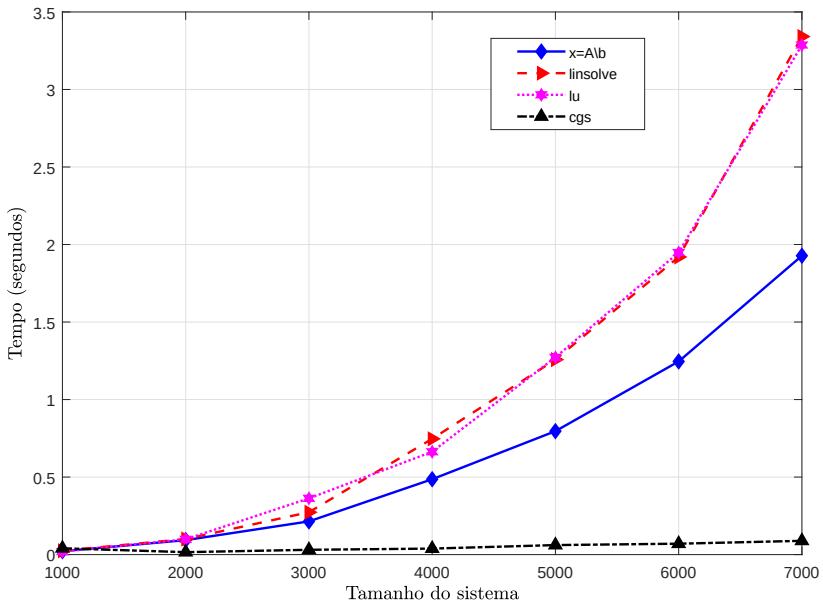
n	x=A\b	linsolve	lu	cgs
1000	0,02206	0,02472	0,01945	0,04073
2000	0,09333	0,100531	0,09859	0,01568
3000	0,21424	0,27333	0,36279	0,03099
4000	0,48607	0,74676	0,66314	0,03874
5000	0,79617	1,25753	1,27226	0,06134
6000	1,24581	1,91956	1,94860	0,07028
7000	1,92758	3,34161	3,28680	0,08881

Fonte: elaborada pelo autor

Analisando os valores apresentados nas tabelas 9 e 10 podemos observar que as funções próprias do MATLAB são muito mais eficientes do que a implementação feita para a eliminação Gaussiana e a Fatoração LU.

Na figura abaixo estão representados os gráficos correspondentes a cada uma das funções.

Figura 8 – Gráfico referente aos dados da tabela 10



Fonte: elaborada pelo autor

Ao analisarmos o gráfico é perceptível o quão eficiente é o método do Gradiente Conjugado em relação aos outros, lembrando que `x=A\b`, `linsolve` e `lu` são métodos diretos, e o método `cgs` é iterativo.

Na próxima tabela, estão registrados os dados das mesmas funções do MATLAB, mas com dimensões maiores que as registradas anteriormente.

Tabela 11 – Tempo gasto: Funções MATLAB para sistemas maiores

n	`x=A\b`	`linsolve`	`lu`	`cgs`	Jacobi	Gauss Seidel
10.000	4,9639226	8,5932481	8,3257953	0,2657513	0,81616	4,03363
20.000	33,056928	62,754067	64,654026	0,6596990	3,33159	13,56972
30.000	106,38788	210,85990	206,60921	1,409891	7,94299	28,71620
40.000	252,34501	492,28709	483,06424	2,6246781	110,06667	117,42262

Fonte: elaborada pelo autor

Para os sistemas com as dimensões expressas na tabela 11, os métodos de Jacobi e Gauss Seidel levam menos tempo para solução que os métodos diretos.

O método Gradiente Conjugado continua superior aos demais métodos comparados nas mesmas dimensões dos sistemas.

4 CONSIDERAÇÕES FINAIS

Os sistemas de equações lineares aparecem em muitos problemas das mais diversas áreas do conhecimento, o que nos dá uma dimensão do quão importante é encontrarmos soluções eficientes para estes sistemas. As pesquisas bibliográficas aqui analisadas possibilitou-nos uma melhor compreensão de alguns métodos de solução mais conhecidos.

A aquisição do *software* matemático MATLAB pela UFT campus Arraias, amplia os horizontes para estudos deste tipo, além de auxiliar nas disciplinas em sala de aula possibilitando aos professores trabalhar determinados conteúdos com a linguagem de programação.

O trabalho com o *software* matemático MATLAB nos auxiliou bastante na implementação dos métodos numéricos, ainda que não tenhamos conseguido uma eficiência tão satisfatória quanto, como é perceptível ao analisarmos os dados obtidos.

Com um nível de programação considerado básico, tentamos ao máximo implementar os métodos numéricos para que uma melhor eficiência fosse alcançada. Com os conhecimentos aqui adquiridos/inseridos temos bagagem para possíveis versões futuras de trabalhos como este, na tentativa de chegarmos a um tempo computacional mais próximo em relação às funções próprias do MATLAB.

REFERÊNCIAS

ARENALES, S.; DAREZZO, A. **Cálculo Numérico**: aprendizagem com apoio de software. 1. ed. São Paulo: Thomson Learning, 2008. Citado na página 37.

BECKER, A. J.; SILVA, D. M. I.; DIAS, F. H. S.; PINHEIRO, L. K. **Noções Básicas de Programação em MATLAB**. Santa Maria, 2010. 68 p. Disponível em: <http://w3.ufsm.br/petmatematica/images/minicursos/Apostilas/Apostila_Matlab.pdf>. Acesso em: 03 nov. 2019. Citado na página 16.

BURDEN, R. L.; FAIRES, J. D. **Análise Numérica**. São Paulo: Thomson Learning, 2003. 736 p. Citado na página 21.

CAMPOS, F. F. **Algoritmos Numéricos**. 2. ed. Rio de Janeiro: LTC, 2007. 420 p. Citado na página 21.

CHAIA, A. V.; DAIBERT, M. R. **Mini Curso Introdução ao MATLAB**. Juiz de Fora. 27 p. Disponível em: <http://www.ufjf.br/getproducao/files/2013/05/Apostila-Mini-Curso-MATLAB-GET-EP1.pdf>. Acesso em: 20 set. 2019. Citado na página 16.

CUNHA, M. C. C. **Métodos Numéricos**. 2. ed. Campinas: Editora da Unicamp, 2000. 280 p. Citado 7 vezes nas páginas 15, 21, 23, 34, 36, 37 e 42.

FRANCO, N. B. **Cálculo Numérico**. 1. ed. São Paulo: Pearson, 2006. 505 p. Citado na página 34.

LOPES, P. C.; RANGEL, R. L.; MARTHA, L. F. **Introdução ao MATLAB**. Rio de Janeiro, 2019. 41 p. Disponível em: <http://webserver2.tecgraf.puc-rio.br/ftp_pub/lfm/CIV2801-IntroducaoMATLAB.pdf>. Acesso em: 04 nov. 2019. Citado na página 16.

RUGGIERO, M. A. G.; LOPES, V. L. d. R. **Cálculo Numérico**: Aspectos teóricos e computacionais. 2. ed. São Paulo: Pearson, 1996. 406 p. Citado 4 vezes nas páginas 15, 21, 22 e 37.

SANTOS, R. N. d. **Uma Breve história do desenvolvimento das teorias dos determinantes e matrizes**. 271 p. Dissertação (Mestrado) — Universidade do Estado de São Paulo, São Paulo, 2007. Disponível em: <http://milanesa.ime.usp.br/imath/files/1/43.pdf>. Acesso em: 03 dez. 2019. Citado na página 47.

SCHWARZ, H. R.; RUTISHAUSER, H.; STIEFEL, E. **Numerical analysis of symmetric matrices**. Englewood Cliffs: Prentice-Hall, 1973. 271 p. Citado na página 36.

SPERANDIO, D.; MENDES, J. T.; SILVA, L. H. M. **Cálculo Numérico**: Características matemáticas e computacionais dos métodos numéricos. São Paulo: Pearson Prentice Hall, 2003. 354 p. Citado na página 14.

Apêndices

Código 1 – Eliminação Gaussiana

```
% Autor: Paulo Cezar Rodrigues Oliveira
% Orientador: Dirlei Ruscheinsky
% Titulo: Eliminacao Gaussiana
% Data: 05/04/19

%======== Metodo de Eliminacao de Gauss ========

function [tempo_um,x]=a_Elim_Gauss(A,b,n);
tic
%========= Triangularizacao do Sistema ========
for k = 1 : n - 1
    for i = k + 1 : n
        m(i,k) = A(k,k)\A(i,k);          % Multiplicador.
        b(i) = b(i) - m(i,k)*b(k);       % Os elementos de b tem seus ...
            valores alterados.
        A(i,:) = A(i,:) - m(i,k)*A(k,:);% Triangularizacao de A
    end
end
%======== Solucao do Sistema Triangular ========
x(n) = A(n,n)\b(n);                  % Primeira equacao a ser resolvida
for k = n-1:-1:1
    sum = b(k);                      % Comando soma recebe o vetor b na ...
        etapa k
    for j = k+1:n
        sum = sum-A(k,j)*x(j);  % Somatorio dos coeficientes com as ...
            respectivas variaveis
    end
    x(k) = A(k,k)\sum;           % Isolamento das variaveis. ...
        Construcao do vetor solucao
end

tempo_um=toc
```

Código 2 – Fatoração LU

```
% Autor: Paulo Cezar Rodrigues Oliveira
% Orientador: Dirlei Ruscheinsky
% Titulo: Fatoracao LU
% Data: 15/04/19

%======== Metodo de Fatoracao LU ========

function [tempo_quatro,x]=b_Fatoracao_LU(A,b,n);
tic
U = A;
L = zeros(n,n);
L = eye(n);
for k = 1 : n - 1
    for i = k + 1 : n
        L(i,k) = U(k,k)\U(i,k);            % Montagem da matriz L
        U(i,:) = U(i,:) - L(i,k)*U(k,:);   % Os elemento de U tem ...
            seus valores alterados
    end
end
%======== Solucao dos Sistemas ========

% Substituicao progressiva LY=B
y=zeros(n,1);
y(1)=b(1);
for i=2:n
    somaLY=0;
    for j=1:i-1
        somaLY = somaLY + L(i,j)*y(j);
    end
    y(i)=b(i)-somaLY;
end
% Substituicao retroativa UX=Y
x = zeros(n,1);
x(n) = U(n,n)\y(n);
for i = n-1:-1:1
    somaUX = 0;
    for j = i+1:n
        somaUX = somaUX + U(i,j)*x(j);
    end
    x(i) = U(i,i)\(y(i) - somaUX);
end

tempo_quatro=toc
```

Código 3 – Jacobi

```
% Autor: Paulo Cezar Rodrigues Oliveira
% Orientador: Dirlei Ruscheinsky
% Titulo: Jacobi
% Data: 23/05/19

%======== Metodo Iterativo de Jacobi ========

function [tempo_oito,xx,k]=c_Jacobi(A,b,n,tol_e)
tic
x(:,1)=zeros(n,1);
L=-tril(A,-1);          % Matriz estritamente triangular inferior
D=diag(A);              % Matriz diagonal
U=-triu(A,1);           % Matriz estritamente triangular superior
R=(L+U)./D;             % Matriz de iteracao
s=b./D;
at=eig(R);              % Autovalor de R
raio=norm(at)           % Raio espectral da matriz R
erro(1)=1;
k=1;
while erro(k)>tol_e
    k=k+1;
    x(:,k)=R*x(:,k-1)+s;    % Resultado da iteracao
    erro(k) = norm(x(:,k) - x(:,k-1));  % Erro absoluto
end
xx=x(k-1,:);

tempo_oito=toc
```

Código 4 – Gauss Seidel

```
% Autor: Paulo Cezar Rodrigues Oliveira
% Orientador: Dirlei Ruscheinsky
% Titulo: Gauss Seidel
% Data: 10/06/19

%======== Metodo de Gauss Seidel ========

function [tempo_nove,w,erro1,k]=d_Gauss_Seidel_(A,b,n,tol_e);
tic
x(:,1)=zeros(n,1);
y=x;
L=-tril(A,-1);        % Matriz estritamente diagonal inferior
U=-triu(A,1);         % Matriz estritamente triangular superior
D=diag(A);            % Matriz diagonal
R=(L+U)./D;           % Matriz de iteracao
s=b./D;
at=eig(R);            % Autovalor de R
raio=norm(at)         % Raio espectral da matriz R
erro(1)=1;
k=2;
while (erro>tol_e)
    for j=1:n
        x(j,k)=R(j,:)*y(:,k-1)+s(j);
        y(j,k-1)=x(j,k);
    end
    y=x;
    erro(k)=norm(x(:,k)-x(:,k-1));
    k=k+1;
end
k=k-1;
w=x(:,k);
erro1=erro(k);

tempo_nove=toc
```

Código 5 – Sobre Relaxação Sucessiva

```
% Autor: Paulo Cezar Rodrigues Oliveira
% Orientador: Dirlei Ruscheinsky
% Titulo: Sobre Relaxacao Sucessiva
% Data: 15/06/19

%============= Metodo SRS =============

function [tempo_dez,w,erro1,k]=e_SRS(A,b,n,W,tol_e);
tic
x(:,1)=zeros(n,1);        % Vetor solucao inicial composto por zeros
xx=x;
L=tril(A,-1);
D=diag(diag(A));
U=triu(A,1);
R = (D + W*L)\((1.0 - W)*D - W*U);      % Matriz de iteracao
s = (D + W*L)\(W*b);
at=eig(R);                % Autovalor de R
rai=norm(at)              % Raio espectral da matriz R
erro(1)=1;
k=2;
while (erro>tol_e)
    for j=1:n
        xx(j,k)=R(j,:)*x(:,k-1)+s(j);
        x(j,k-1)=xx(j,k);
    end
    x=xx;
    erro(k) = norm(xx(:,k) - xx(:,k-1));
    k=k+1;
end
k=k-1;
w=x(:,k);
erro1=erro(k);

tempo_dez=toc
```