PROFESSOR DE MATEMÁTICA ONLINE
Revista eletrônica da Sociedade Brasileira de Matemática

PMO v.12, n.1, 2024
ISSN: 2319-023X
https://doi.org/10.21711/2319023x2024/pmo1202

Ensino e aprendizagem de métodos numéricos para resolução de sistemas lineares utilizando a ferramenta Google Colaboratory

Valdex de J. Santos Danton R. da Costa Pitombo Marinilton J. O. Lima júnior
Lucas R. de Souza Meira Victor Bastos G. Moura

Resumo

Este artigo trata do ensino de métodos numéricos para resolução de sistemas lineares com o uso da ferramenta Google Colaboratory. A grande vantagem de utilizar tal ferramenta é a possibilidade de incorporar e executar códigos, assim como gerar gráficos em um ambiente de texto, inclusive utilizando linguagem Latex. Neste sentido, foi possível abordar os métodos numéricos e, ao mesmo tempo, implementar tais métodos para resolver sistemas lineares, executando-os e gerando gráficos interpretativos das soluções obtidas. São apresentados exemplos de como o Colaboratory pode ser utilizado para ensinar métodos numéricos para resolução de sistemas lineares, tais como: Método de Gauss, Gauss-Jordan, decomposição LU e métodos iterativos. Os resultados mostram que o uso do Colaboratory pode auxiliar a identificar erros e a entender melhor o comportamento dos algoritmos, possibilitando uma análise mais detalhada das soluções obtidas. Neste sentido, o Colaboratory é uma ferramenta útil para o ensino de métodos numéricos para resolução de sistemas lineares, possibilitando uma aprendizagem mais integradora entre desenvolvimento, implementação e interpretação de resultados.

Palavras-chave: Sistemas lineares; métodos numéricos; Google colaboratory; ensino.

Abstract

This article deals with the teaching of numerical methods for solving linear systems using the Google Colaboratory tool. The great advantage of using this tool is the possibility of incorporating and executing codes, as well as generating graphics in a text environment, including using Latex language. In this sense, it was possible to address the numerical methods and, at the same time, implement such methods to solve linear systems, running them and generating interpretive graphics of the obtained solutions. Examples are presented of how Colaboratory can be used to teach numerical methods for solving linear systems, such as: Gauss Method, Gauss-Jordan, LU decomposition, and iterative methods. The results show that the use of Colaboratory can help identify errors and better understand the behavior of algorithms, enabling a more detailed analysis of the solutions obtained. In this sense, Colaboratory is a useful tool for teaching numerical methods for solving linear systems, enabling a more integrated learning between development, implementation, and interpretation of results.

Keywords: Linear systems; numerical methods; Google Colaboratory; teaching.

1. Introdução

Neste artigo, apresentamos de forma concisa o estudo desenvolvido pelos quatro últimos autores, sob orientação do primeiro, na disciplina de Cálculo Numérico. Foram apresentados cinco métodos para resolução de sistemas lineares de qualquer ordem, a saber: Gauss, Gauss-Jordan, Fatoração LU, Jacobi e Gauss-Seidel. Para isso, implementamos algoritmos utilizando o Google Colaboratory, que se mostrou uma ferramenta útil para o desenvolvimento dos métodos numéricos, por dois motivos principais.

O primeiro motivo foi a laboriosidade de aferição em algoritmos puramente transcritos em ambientes não preparados para executá-los e/ou modificá-los, como a linguagem LaTeX. O segundo motivo foi a possibilidade de interação conjunta entre o algoritmo e a parte escrita em tempo real, o que permitiu a avaliação do comportamento do algoritmo quando submetido a diferentes parâmetros de entrada fundamentais, definidos pelo próprio usuário.

A utilização do ambiente Colabory foi feita durante as aulas de Cálculo Numérico, de maneira colaborativa entre estudantes e professor, visando melhorar a aprendizagem da disciplina. Neste sentido, fez-se um comparativo entre os métodos numéricos, de forma que os estudantes pudessem perceber as vantagens e desvantagens da utilização de cada um deles, conforme a necessidade do problema a ser resolvido.

Iniciamos com uma breve introdução aos conceitos fundamentais de equações e sistemas lineares. Em seguida, foram abordados cada um dos métodos numérico diretos: Gauss, Guass-Jordan e Fatoração LU, apresentando exemplos resolvidos e algoritmos correspondentes. Também foram explorados os métodos iterativos, demonstrando sua aplicação com exemplos práticos e algoritmos editáveis.

Para avaliar a eficiência dos métodos de forma imparcial, geramos matrizes aleatórias. É relevante destacar que tanto os exemplos resolvidos quanto os códigos são apresentados de forma didática e editável, permitindo que qualquer usuário os personalize conforme sua necessidade.

O material produzido (link: https://abre.ai/calcnumsistlinear) é uma valiosa ferramenta para professores da Educação Básica que desejam experimentar esses métodos em suas aulas, proporcionando uma abordagem diferenciada. Eles podem facilmente copiar, reproduzir e adaptar o conteúdo de acordo com suas preferências e objetivos pedagógicos.

2. Sistemas lineares

Uma equação linear é uma expressão matemática na forma $a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n = b$, onde $a_1, a_2, a_3, \ldots, a_n$ são números reais, coeficientes das variáveis $x_1, x_2, \ldots, x_n$, respectivamente, e b é um número real que corresponde ao termo independente. Quando $b = 0$, a equação é chamada de homogênea.

Um sistema linear é constituído por um conjunto de equações lineares com a forma:

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

Em um sistema linear, o conjunto solução é solução de todas as equações lineares do sistema. Em

outras palavras, o valor de $x_1, x_2, x_3, \ldots x_n$ deve satisfazer simultaneamente todo o conjunto de equações do sistema.

2.1. Classificação de sistemas lineares

Os sistemas lineares são classificados de acordo com o número de soluções que possuem, sendo que existem três possibilidades:

SPD — Sistema Possível e Determinado — possui uma única solução.

SPI — Sistema Possível e Indeterminado — possui infinitas soluções.

SI — Sistema Impossível — não possui solução.

Na Figura 1 apresentamos uma ilustração da classificação dos sistemas lineares, conforme destacamos acima.

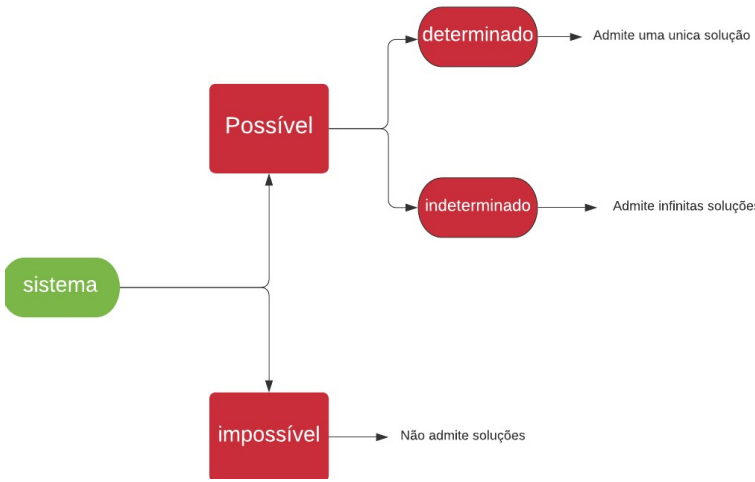

Figura 1: Ilustração da classificação de sistemas lineares
Fonte: Criado pelos autores na ferramenta https://www.lucidchart.com.

2.2. Matriz associada a um sistema linear

Podemos associar um sistema linear a uma matriz, de forma que seus coeficientes ocupem as linhas e colunas na matriz. Por exemplo, considerando o sistema linear abaixo:

$$\begin{cases} x + y + z = 1 \\ 4x + 4y + 2z = 2 \\ 2x + y - z = 0 \end{cases}$$

Podemos escrevê-lo na forma matricial como:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 4 & 4 & 2 \\ 2 & 1 & -1 \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}}_{b}$$

Onde A é a matriz de coeficientes, X é o vetor coluna das variáveis do sistema, e b é o vetor coluna dos termos independentes do sistema. Logo, podemos escrever o sistema como AX = b. Na matriz A, cada linha representa uma equação do sistema e cada coluna representa uma variável do sistema.

3. Solução numérica de sistemas lineares

Quando o número de equações e incógnitas é pequeno, é possível resolver sistemas lineares por técnicas simples. Entretanto, quando o número de equações é muito grande, a resolução manual do sistema torna-se um processo laborioso e, muitas vezes, impraticável. Nesse sentido, a computação e a utilização de métodos numéricos computacionais tornam-se de grande valia.

Este artigo apresenta cinco métodos para resolução de sistemas lineares de qualquer ordem: o método de Gauss, o método de Gauss-Jordan, o método de Fatoração LU, o método de Jacobi e o método de Gauss-Seidel.

3.1. Método de Gauss

O Método de Gauss, também conhecido como escalonamento ou eliminação gaussiana, é uma técnica empregada para solucionar sistemas lineares, por meio da realização de operações elementares na matriz estendida do sistema. Esse processo visa transformá-la em uma matriz triangular superior, conhecida como *matriz escalonada do sistema*.

Uma vez alcançada a matriz escalonada, a solução pode ser facilmente encontrada por meio da substituição regressiva. É importante ressaltar que as operações elementares realizadas devem preservar a solução do sistema, e elas consistem em: multiplicar uma linha por uma constante não nula, substituir uma linha por ela mesma acrescida de um múltiplo de outra linha, ou permutar as linhas.

A seguir, será apresentado um exemplo de como aplicar o Método de Gauss para resolver um sistema linear:

Exemplo 1:
Resolva o sistema, por eliminação gaussiana:

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 0 \\ 2x + 2y + z &= 1 \end{aligned} \tag{2}$$

Solução:

Passo 1: Consideramos a matriz estendida do sistema, dada por:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 2: Para zerar o elemento imediatamente inferior ao primeiro elemento da matriz, substitui-se a linha 2 pela diferença entre ela mesma e a linha 1 multiplicada por 2:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_2 \rightleftharpoons l_2 - 2l_1 \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 3: Substituímos a linha 3 pela subtração dela mesma com a linha 1 multiplicada por 2:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_3 \rightleftharpoons l_3 - 2l_1 \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

Passo 4: multiplicamos as linhas 2 e 3 por –1 de modo a deixar os valores da digonal principal positivos, obtendo assim:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

A matriz obtida é a matriz escalonada do sistema linear (2), escalonado pelo método de eliminação gaussiana. Para escrever esse sistema na forma de equações lineares, basta usar as linhas da matriz. E assim, temos:

$$\begin{aligned} x + y + z &= 1 \\ y + 3z &= 2 \\ z &= 1 \end{aligned} \tag{3}$$

Observe que a terceira equação já nos fornece o valor de $z = 1$. Podemos substituir esse valor nas equações anteriores e obter os valores de x e y. Fazendo isso, encontramos $\{x = 1, y = -1, z = 1\}$.

3.2. Método de Gauss com pivoteamento

Muitas vezes, o método de Gauss, embora seja tradicional e eficiente, apresenta um problema sistemático: ele não funciona bem quando os valores dos pivôs são próximos a zero, podendo até mesmo falhar completamente se o pivô for igual a zero ou muito próximo. Para solucionar esse problema, utilizamos o **Método de Gauss com Pivoteamento**.

Essa técnica é uma variação do método de Gauss que visa evitar erros numéricos que podem ocorrer durante a eliminação gaussiana padrão. O método consiste em escolher o elemento pivotante (ou pivô) - o valor absoluto máximo na coluna atual - antes de executar as operações elementares de linha.

O pivoteamento parcial ajuda a garantir que os elementos diagonais principais sejam sempre os maiores de suas respectivas colunas, reduzindo a possibilidade de erros numéricos.

A seguir, será ilustrado como resolver um sistema utilizando esse método, por meio de um exemplo.

Exemplo 2: Vamos resolver o sistema

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 0 \\ 2x + 2y + z &= 1 \end{aligned} \tag{4}$$

por eliminação gaussiana com pivoteamento.

Passo 1: Consideramos a matriz estendida do sistema, dada por:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 2: Observamos que os maiores elementos em valor absoluto encontram-se nas linhas 2 e 3 (são iguais). Portanto, fazemos a permutação das linhas 1 e 2:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_1 \rightleftharpoons l_2 \rightarrow \begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 3: Agora podemos realizar subtrações de linhas para zerar elementos da matriz. Primeiro, subtraímos da segunda linha a primeira linha multiplicada por 1/2, e depois subtraímos a terceira linha pela primeira multiplicada pelo número 1:

$$\begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_2 \rightleftharpoons l_2 - \frac{1}{2} l_1 \text{ e } l_3 \rightleftharpoons l_3 - l_1 \rightarrow \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & \frac{1}{2} & \frac{3}{2} & 1 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$

Passo 4: Como o maior elemento da segunda coluna está na terceira linha, fazemos a permutação das linhas 2 e 3:

$$\begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & \frac{1}{2} & \frac{3}{2} & 1 \\ 0 & 1 & 2 & 1 \end{bmatrix} l_3 \rightleftharpoons l_2 \rightarrow \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & \frac{1}{2} & \frac{3}{2} & 1 \end{bmatrix}$$

Passo 5: Subtraímos da terceira linha, a segunda multiplicada por 1/2 ($l_3 = l_3 - \frac{1}{2} l_2$) para zerar o elemento da terceira linha e segunda coluna.

$$\begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & \frac{1}{2} & \frac{3}{2} & 1 \end{bmatrix} l_3 \rightleftharpoons \frac{1}{2} l_2 - l_3 \rightarrow \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Após obtermos a matriz escalonada na forma escalonada reduzida, podemos usar o método de substituição regressiva para encontrar a solução do sistema. A última equação fornece-nos o valor de $z = 1$. Substituindo esse valor na segunda equação, encontramos que $y + 2 \cdot 1 = 1$, o que implica que $y = -1$. Finalmente, substituindo os valores de y e z na primeira equação, temos $2x - 1 - 1 = 0$, o que implica que $x = 1$. Portanto, a solução do sistema é $\{x = 1, y = -1, z = 1\}$.

3.2.1 Algoritmo de Gauss

Abaixo segue o algoritmo do Método de Gauss com pivoteamento parcial, implementado no Google Colabory.

```
%%writefile Gauss.m
function x = Gauss(A, b)
% Implementação do método de eliminação de Gauss para resolver sistemas
% lineares Ax = b.
    % Verificação de pré-condição
    [m, n] = size(A);
    if m ~= n
        error('A matriz deve ser quadrada');
    end
    % Adicionando a coluna b à matriz A
    Aug = [A, b];
    % Inicializando vetor de solução x
    x = zeros(n, 1);
    % Eliminação gaussiana
    for k = 1:n-1
        [~, i] = max(abs(Aug(k:n, k)));
        ipr = i + k - 1;
        if ipr ~= k
            Aug([k, ipr], :) = Aug([ipr, k], :);
        end
        for i = k+1:n
            pivor = Aug(i, k) / Aug(k, k);
            Aug(i, k:n+1) = Aug(i, k:n+1) - pivor*Aug(k, k:n+1);
        end
    end
    % Retrosubstituição
    x(n) = Aug(n, n+1) / Aug(n, n);
    for i = n-1:-1:1
        x(i) = (Aug(i, n+1) - Aug(i, i+1:n)*x(i+1:n)) / Aug(i, i);
    end
end
```

3.3. Método de Gauss-Jordan

O Método de Gauss-Jordan é um procedimento para resolver sistemas de equações lineares por meio da aplicação de operações elementares nas linhas da matriz aumentada do sistema. Essas operações são executadas até que se obtenha uma matriz na forma diagonal, com elementos normalizados, que permita a fácil resolução do sistema. O vetor da última coluna da matriz diagonalizada corresponde à solução do sistema.

O método de Gauss-Jordan tem algumas vantagens em relação ao método de eliminação de Gauss, como a resolução direta de todas as variáveis, e pode ser utilizado para resolver sistemas lineares

com múltiplas soluções de maneira eficiente. Isso é possível porque, ao final do processo de eliminação, a matriz aumentada é transformada em uma matriz na forma escalonada reduzida por linhas, que permite identificar de maneira clara se o sistema possui soluções únicas, múltiplas ou nenhuma solução.

No entanto, o método de Gauss-Jordan pode ser computacionalmente mais custoso do que outros métodos, especialmente para matrizes grandes, devido ao seu maior número de operações aritméticas. Além disso, o pivoteamento parcial é geralmente necessário para evitar divisões por zero e garantir a estabilidade numérica do método, o que pode aumentar o custo computacional. Por esses motivos, o método de Gauss-Jordan é mais frequentemente utilizado em sistemas de tamanho moderado, em que as suas vantagens superam as suas desvantagens.

Para exemplificar o método, vamos considerar o seguinte sistema de equações:

$$\begin{aligned} x + y + z &= 1 \\ 2x + y + z &= 0 \\ 2x + 2y + z &= 1 \end{aligned}$$

pelo método de Gauss-Jordan.

Passo 1: O primeiro passo é construir a matriz estendida do sistema, como segue:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 2: O segundo passo é isolar o pivô da primeira linha e zerar os demais elementos da mesma coluna. Para isso, substitui-se a linha 2 pela diferença da linha 2 pela linha 1 multiplicada por 2:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_2 \rightleftharpoons l_2 - 2l_1 \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -2 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Passo 3: O terceiro passo consiste em subtrair da linha 3 a linha 1 multiplicada por 2, obtendo-se o valor 0 para o último elemento da coluna 1. Zera-se também o último elemento da segunda coluna:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -2 \\ 2 & 2 & 1 & 1 \end{bmatrix} l_3 \rightleftharpoons l_3 - 2l_1 \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -2 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

Passo 4: No quarto passo, multiplica-se a linha 2 e a linha 3 por –1, para deixar os valores positivos:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Passo 5: No quinto passo, substitui-se a linha 1 pela diferença da própria linha 1 pela linha 2:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix} l_1 \rightleftharpoons l_1 - l_2 \rightarrow \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Passo 6: Por fim, substitui-se a linha 2 pela diferença da própria linha 2 pela linha 3:

$$\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix} l_2 \rightleftharpoons l_2 - l_3 \rightarrow \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

A matriz resultante é a matriz escalonada do sistema, com os elementos da diagonal principal da matriz dos coeficientes todos unitários. Por consequência, a solução do sistema corresponde ao vetor da última coluna da matriz. Ou seja, a solução do sistema é:

$$\begin{Bmatrix} x = -1 \\ y = 1 \\ z = 1 \end{Bmatrix}$$

3.3.1 Algoritmo de Gauss Jordan

```
%%writefile GaussJordan.m
 function GaussJordan(A,b)
% Concatenando a matriz A e o vetor b em uma matriz aumentada
Aug = [A b];
% Obtendo o tamanho da matriz aumentada
[m,n] = size(Aug);
for i = 1:n-1
    % Encontrando o pivô
    [~,k] = max(abs(Aug(i:n-1,i)));
    ipr = i+k-1;
    if ipr ~= i
        % Trocando as linhas
        Aug([i,ipr],:) = Aug([ipr,i],:);
    end
    % Verificando se o pivô é zero
    if Aug(i,i) == 0
        error("O pivô é zero, não é possível continuar a eliminação.");
    end
    % Normalizando a linha do pivô
    N = Aug(i,:);
    N = N/Aug(i,i);
    Aug(i,:) = N;
    % Subtraindo as outras linhas
    for j = 1:n-1
        if i ~= j
            Aug(j,:) = Aug(j,:) - N*Aug(j,i);
        end
    end
end
% Obtendo a solução
x = Aug(:,n);
% Imprimindo a solução
```

```
 x= Aug(:,n)
end
```

3.4. Método de Fatoração LU

O método de Fatoração LU (Lower-Upper) é uma técnica utilizada para resolver sistemas de equações lineares que consiste em fatorar a matriz dos coeficientes A em duas matrizes: uma matriz triangular inferior L e uma matriz triangular superior U, para escrever A = LU. A fatoração da Matriz A nas matrizes L e U é explicada em livros de Cálculo Numérico, como feito por Lopes e Ruggiero[1].

Feita a fatoração, obtêm-se dois outros sistemas: Ly = b, triangular inferior, e Ux = y, triangular superior. Assim, o sistema de equações lineares Ax = b é resolvido em duas etapas: primeiro, o sistema Ly = b é resolvido para y, onde L é uma matriz triangular inferior, e depois o sistema Ux = y é resolvido para x, onde U é uma matriz triangular superior.

Uma das vantagens da fatoração LU é que, uma vez que as matrizes L e U foram calculadas, podemos resolver o mesmo sistema para diferentes vetores b com um custo computacional reduzido, pois os sistemas triangulares resultantes são mais simples de resolver do que o sistema original.

O método LU é amplamente utilizado em aplicações em engenharia, física, matemática, economia e outras áreas onde sistemas de equações lineares precisam ser resolvidos. Além disso, muitos algoritmos de métodos numéricos usam o método LU como uma etapa fundamental em seus cálculos.

Segue um exemplo passo a passo como transformar uma matriz A em uma matriz triangular inferior L e uma matriz triangular superior U. Para tanto, considere a matriz A, dada por:

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

Passo 1: A matriz L começa como matriz identidade, e a matriz U é obtida através da matriz A através da fatoração gaussiana, conforme explicado na Seção 3.1. Dessa forma, temos:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{e} \quad U = \begin{bmatrix} 2 & 1 & -1 \\ -4 & 2 & 3 \\ -2 & 1 & 2 \end{bmatrix}$$

Passo 2: Colocamos as duas matrizes como matriz estendida, dada por:

$$[L|U] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & 1 & -1 \\ 0 & 1 & 0 & -4 & 2 & 3 \\ 0 & 0 & 1 & -2 & 1 & 2 \end{array}\right]$$

Passo 3: Dividimos a primeira linha por 2:

$$[L|U] = \left[\begin{array}{ccc|ccc} 1/2 & 0 & 0 & 1 & 1/2 & -1/2 \\ 0 & 1 & 0 & -4 & 2 & 3 \\ 0 & 0 & 1 & -2 & 1 & 2 \end{array}\right]$$

Passo 4: Multiplicamos a primeira linha por 4 e somamos a segunda linha, e a multiplicamos por 2 e somamos a terceira linha, obtendo:

$$\left[\begin{array}{ccc|ccc} 1/2 & 0 & 0 & 1 & 1/2 & -1/2 \\ 2 & 1 & 0 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 2 & 1 \end{array}\right]$$

Passo 5: Dividimos a segunda linha por 4:

$$\left[\begin{array}{ccc|ccc} 1/2 & 0 & 0 & 1 & 1/2 & -1/2 \\ 1/2 & 1/4 & 0 & 0 & 1 & 1/4 \\ 1 & 0 & 1 & 0 & 2 & 1 \end{array}\right]$$

Passo 6: Multiplicamos a segunda linha –2 e somamos a terceira.

$$\left[\begin{array}{ccc|ccc} 1/2 & 0 & 0 & 1 & 1/2 & -1/2 \\ 1/2 & 1/4 & 0 & 0 & 1 & 1/4 \\ 0 & -1/2 & 1 & 0 & 0 & 1/2 \end{array}\right]$$

Assim, obtemos as matrizes

$$\mathrm{L} = \begin{bmatrix} 1/2 & 0 & 0 \\ 1/2 & 1/4 & 0 \\ 0 & -1/2 & 1 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 1/2 & -1/2 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1/2 \end{bmatrix}$$

Assim, dado qualquer vetor coluna de termos indpendentes b, resolvemos o sistema Ly = b, obtendo os valores do vetor y e, em seguida, resolvemos o sistema Ux = y, obtendo a solução do sistema apresentado. Por exemplo, se tivermos o sistema

$$\begin{cases} 2\mathrm{x} + \mathrm{y} - \mathrm{z} = 5 \\ -4\mathrm{x} + 2\mathrm{y} + 3\mathrm{z} = 3 \\ -2\mathrm{x} + \mathrm{y} + 2\mathrm{z} = 1 \end{cases}$$

obteremos a solução: $(\mathrm{x} = 1/4, \mathrm{y} = 7/2, \mathrm{z} = -1)$.

3.4.1 Algoritmo da Fatoração LU

O código abaixo mostra uma implementação feita no Google Colaboratory do método de fatoração LU:

```
function x = LU(A, b)
    [m, n] = size(A);
    if m ~= n
        error('A matriz deve ser quadrada');
    end
    % Pivoteamento:
    for j = 1:n
        [pivot_val, pivot_row] = max(abs(A(j:n, j)));
        pivot_row = pivot_row + j - 1;
```

```
        if pivot_val == 0
            error('A matriz é singular');
        end
        if pivot_row ~= j
            A([j, pivot_row], :) = A([pivot_row, j], :);
            b([j, pivot_row]) = b([pivot_row, j]);
        end
    end
    % Fatoração LU:
    L = eye(n);
    for k = 1:n-1
        for i = k+1:n
            L(i, k) = A(i, k) / A(k, k);
            A(i, k+1:n) = A(i, k+1:n) - L(i, k) * A(k, k+1:n);
        end
    end
    U = triu(A);
    % Solução do sistema:
    y = zeros(n, 1);
    for i = 1:n
        y(i) = (b(i) - L(i, 1:i-1) * y(1:i-1)) / L(i, i);
    end
    x = zeros(n, 1);
    for i = n:-1:1
        x(i) = (y(i) - U(i, i+1:n) * x(i+1:n)) / U(i, i);
    end
end
```

3.5. Métodos iterativos

Na área das ciências exatas computacionais, os métodos iterativos são amplamente aplicados para resolver problemas complexos sem soluções analíticas ou de difícil obtenção. Esses métodos permitem gerar sequências de resultados aproximados que são refinados e aprimorados a cada iteração executada. Dentre esses, destacam-se os métodos de Jacobi e de Gauss-Seidel.

Os métodos iterativos são particularmente valiosos na resolução de sistemas esparsos, onde os métodos diretos podem ser ineficazes, podendo estar sujeitos a erros de arredondamento. Como o erro nos métodos iterativos é controlado pelo número de iterações, os erros de arredondamento não são um motivo de preocupação.

Por exemplo, considere o sistema linear:

$$\begin{aligned} 2x + 0y + 0z &= 4 \\ 0x + 3y + 0z &= 9 \\ 0x + 0y + 5z &= 10 \end{aligned} \tag{5}$$

que tem solução $(x = 2, y = 3, z = 2)$. Os métodos de Jacobi e Gauss-Seidel resolvem esse sistema de forma eficaz, embora seja trivial nesse caso. No entanto, sua importância cresce em sistemas maiores com estrutura esparsa.

Neste exemplo, a matriz de coeficientes é diagonal, o que significa que a maioria dos elementos é zero, exceto os elementos na diagonal principal. Esse é um caso clássico em que os métodos diretos podem ser ineficientes, pois eles não se beneficiam da estrutura esparsa da matriz.

Por outro lado, os métodos iterativos de Jacobi e Gauss-Seidel são altamente eficazes para sistemas esparsos. Eles exploram a estrutura da matriz, atualizando iterativamente as variáveis com base nas entradas não nulas, resultando em convergência mais rápida e economia de memória em comparação com métodos diretos. Portanto, esses métodos são adequados para resolver sistemas lineares esparsos, como o exemplo dado.

É importante salientar que, como não sabemos *a priori* a solução dos sistemas a serem resolvidos, precisamos calcular o erro relativo que, no caso dos métodos iterativos, pode ser expresso pela equação[2]:

$$\left|\varepsilon_{a,i}\right| = \left|\frac{x_i^j - x_i^{j-1}}{x_i^j}\right| 100\% < \varepsilon_s \tag{6}$$

para todo i, onde j e j – 1 representam a iteração atual e a anterior.

3.5.1 Método de Jacobi

O método iterativo de Jacobi é utilizado para aproximar a solução de sistemas lineares representados por uma matriz A e um vetor b na forma AX = b, onde o objetivo é encontrar uma solução aproximada a partir de um chute inicial $x^{(1)}$.

Assim, dado o sistema linear,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= y_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= y_2 \\ &\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= y_n \end{aligned} \tag{7}$$

o processo iterativo consiste em isolar cada elemento x_n da equação correspondente e utilizar os elementos da iteração atual $x_1^{(k)}, x_2^{(k)}, \ldots, x_{n-1}^{(k)}, x_{n+1}^{(k)}, \ldots, x_n^{(k)}$ para estimar a próxima iteração x_{n+1}. A fórmula para essa estimativa é dada por:

$$x_n^{(k+1)} = \frac{y_n - (a_{n1}x_1^{(k)} + \cdots + a_{n,n-1}x_{n-1}^{(k)})}{a_{nn}}$$

O processo iterativo é executado até se alcançar uma precisão desejada ou um número máximo de iterações. É importante ressaltar que a convergência do método de Jacobi depende da matriz A ser estritamente diagonal dominante ou simétrica definida positiva. Este método, conforme ressaltado anteriormente, é extremamente útil na resolução de sistemas esparsos.

3.5.2 Método de Gauss-Seidel

O método de Gauss-Seidel é semelhante ao método de Jacobi, pois ambos utilizam manipulações algébricas para isolar os elementos do sistema em cada iteração. No entanto, o Método de Gauss-Seidel é mais rápido, pois utiliza o valor atualizado de x na mesma iteração para obter o próximo valor, proporcionando uma convergência mais rápida para a solução do sistema.

O algoritmo do Método de Gauss-Seidel envolve a resolução de um sistema triangular inferior para cada iteração e a atualização imediata dos valores de x. A partir de um chute inicial $x^{(1)}$, os elementos $x_n^{(k)}$ são isolados para estimar a próxima iteração x_{n+1}. Dessa forma, o método é iterativo e refinado a cada iteração até que a solução desejada seja atingida.

Exemplo: Use o método de Gauss-Seidel para obter a solução do sistema:

$$3x_1 - 0,1x_2 - 0,2x_3 = 7,85 \tag{8}$$
$$0,1x_1 + 7x_2 - 0,3x_3 = -19,3 \tag{9}$$
$$0,3x_1 - 0,2x_2 + 10x_3 = 71,4 \tag{10}$$

Passo 1: Em cada uma das equações, isolamos a variável na diagonal:

$$x_1 = \frac{7,85 + 0,1x_2 + 0,2x_3}{3}$$
$$x_2 = \frac{-19,3 - 0,1x_1 + 0,3x_3}{7}$$
$$x_3 = \frac{71,4 - 0,3x_1 + 0,2x_2}{10}$$

Passo 2: Supondo-se que x_2 e x_3 sejam iguais a zero, a Equação (8) pode ser usada para calcular

$$x_1 = \frac{7,85 + 0 + 0}{3} = 2,616667$$

Passo 3: Esse valor, junto com o valor suposto de $x_3 = 0$, pode ser substituído na Equação (9) para calcular

$$x_2 = \frac{-19,3 - 0,1(2,616667) + 0}{7} = -2,794524$$

Passo 4: A primeira iteração é completada substituindo-se os valores calculados para x_1 e x_2 na Equação (10) para obter

$$x_3 = \frac{71,4 - 0,3(2,616667) + 0,2(-2,794524)}{10} = 7,005610$$

Passo 5: Para a segunda iteração, o mesmo processo é repetido para calcular

$$x_1 = \frac{7,85+0,1(-2,794524)+0,2(7,005610)}{3} = 2,990557$$
$$x_2 = \frac{-19,3-0,1(2,990557)+0,3(7,005610)}{7} = -2,499625$$
$$x_3 = \frac{71,4-0,3(2,990557)+0,2(-2,499625)}{10} = 7,000291$$

O método está, portanto, convergindo para a verdadeira solução, uma vez que o erro relativo é pequeno. Iterações adicionais podem ser aplicadas para melhorar as respostas. Porém, em um problema real, a resposta verdadeira não seria conhecida *a priori*. Consequentemente, a Equação (6) fornece um meio de estimar o erro. Por exemplo, para x_1,

$$|\varepsilon_{a,1}| = \left|\frac{2,990557 - 2,616667}{2,990557}\right| 100\% = 12,5\%$$

Para x_2 e x_3, as estimativas de erro são $|\varepsilon_{a,2}| = 11,8\%$ e $|\varepsilon_{a,3}| = 0,076\%$. Assim, quando elas são satisfeitas, garantem que o resultado é conhecido pelo menos dentro da tolerância especificada por ε_s. Tal processo iterativo pode continuar para diminuir mais o erro relativo e obter aproximações melhores.

No caso do método de Jacobi, utilizaríamos o valor original de x_1 no passo 3 e não o valor atualizado no passo anterior. O mesmo procede-se para as demais variáveis. Nesse sentido o Método de Gauss-Seidel permite-nos encontrar soluções mais rápidas, dentro da aproximação aceitável.

Os algoritmos dos métodos de Jacobi e Gauss-Seidel estão disponíveis no arquivo projeto em: https://abre.ai/calcnumsistlinear

4. Ambiente Colaboratory (COLAB) nas Aulas de Cálculo Numérico

O Colaboratory (COLAB), que se encontra no *link* https://colab.research.google.com/, é uma plataforma em nuvem desenvolvida pelo Google que possibilita aos usuários escrever, executar e compartilhar códigos em linguagens como Python, R, Octave e MatLab. É amplamente adotado por estudantes, pesquisadores, cientistas de dados e desenvolvedores em todo o mundo, devido à sua eficiência e à capacidade de aprimorar a experiência de aprendizado e trabalho.

Entre as suas vantagens, destaca-se o acesso gratuito aos recursos de computação de alta qualidade oferecidos pelo Google, bem como a possibilidade de compartilhar *notebooks* e arquivos com outros usuários. Além disso, o COLAB integra-se perfeitamente com outras ferramentas do Google[3]. Sua utilização é facilitada por diversos recursos úteis, tais como autocompletar código, depuração interativa e visualizações interativas, o que a torna uma ferramenta poderosa e versátil.

Utilizamos a plataforma Colab no curso de Cálculo Numérico para a implementação colaborativa dos Métodos Numéricos destinados à resolução de sistemas lineares, envolvendo tanto alunos quanto o professor. Com isso, a ferramenta passou a ser uma parte rotineira das aulas, auxiliando na resolução de exemplos e na compreensão dos conceitos abordados na disciplina. Com o desenvolvimento colaborativo dos métodos, qualquer pessoa pode acessar e simular soluções para sistemas lineares diversos, simplesmente acessando o *link* do projeto no Google Colaboratory, disponível neste link.

Na Figura 2, apresentamos o método de Gauss implementado no ambiente Colaboratory para este artigo, permitindo modificar os parâmetros de entrada do algoritmo e observar o seu comportamento em relação a diferentes sistemas. Na Figura 3 ilustramos a execução do código para uma matriz de coeficientes 10×10 gerada com números aleatórios e termos independentes gerados de forma similar. Dessa forma, é possível modificar os parâmetros de entrada do algoritmo e observar como ele se comporta em relação a diferentes sistemas.

5. Comparações entre os métodos numéricos

Existem diversas abordagens em métodos numéricos para solução de sistemas lineares, cada uma com suas particularidades. Para escolher a melhor ferramenta para cada situação, é crucial realizar uma análise cuidadosa. É possível categorizar os métodos em duas principais vertentes: diretos e indiretos. Para fazer essa avaliação, é importante considerar aspectos como custo computacional, facilidade de implementação, garantia de convergência e complexidade. Assim, é possível identificar as vantagens e desvantagens de cada método e suas possíveis limitações, a fim de escolher o mais efetivo em cada situação.

```
%%writefile Gauss.m
function x = Gauss(A,b)
[m,n] = size(A);
if m~=n
    fprintf("Matriz deve ser quadrada\n")
endif
nb = n+1;
Aug = [A b];
for k = 1:n-1
    [~,i] = max(abs(Aug(k:n,k)));
    ipr = i +k-1;
    if ipr ~= k
        Aug([k,ipr],:)=Aug([ipr,k],:);
    endif
    for i = k+1:n
        pivor = Aug(i,k)/Aug(k,k);
        Aug(i,k:nb) = Aug(i,k:nb)-pivor*Aug(k,k:nb);
    endfor
endfor
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
    x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
endfor
endfunction
```

```
Writing Gauss.m
```

Figura 2: Algoritmo de Gauss efetivado no COLAB

```
[6] %%writefile testem100_1_Gauss.m
    A=randi(100,10,10);
    b=randi(110,10,1);
    Gauss(A,b)
```

```
Overwriting testem100_1_Gauss.m
```

```
# Solução de matriz 100x100 com o método de Gauss
!octave -W testem100_1_Gauss.m
```

```
ans =

    0.4021754
    0.8604504
   -0.3506931
    0.0116896
    0.9963863
    0.0041950
    0.6265070
   -1.4769818
   -0.4926100
    0.3207672
```

Figura 3: Formulação dos parâmetros de entrada

5.1. Métodos diretos

Os métodos numéricos de eliminação de Gauss, Gauss-Jordan e fatoração LU possuem semelhanças significativas, já que são métodos diretos que transformam o sistema em uma forma equivalente, mais simples de ser resolvida. No método de Gauss, o sistema é manipulado por operações básicas até se obter uma matriz triangular superior ou inferior, seguida de substituições regressivas para encontrar a solução.

No método de Gauss-Jordan, a matriz é manipulada até se obter um sistema equivalente na forma diagonal, com elementos normalizados, dispensando a necessidade de substituições regressivas para encontrar a solução do sistema.

Já na fatoração LU, a matriz A é fatorada em L (matriz triangular inferior) e U (matriz triangular superior), ou seja, A = LU. Para encontrar a solução de Ax = b, resolve-se primeiro o sistema

triangular inferior Ly = b e, em seguida, o sistema triangular superior Ux = y.

Todos os métodos requerem rotinas de pivoteamento para evitar problemas quando um dos elementos da diagonal principal for nulo. O pivoteamento consiste em fazer uma permutação de linhas para escolher o maior pivô em módulo em cada etapa.

Cláudio G. Canuto e Carlos E. Kenig, [4], afirmam que "os métodos diretos apresentam-se como a melhor escolha quando o problema é de dimensões reduzidas e dispõe-se de suficiente capacidade de armazenamento e processamento".

5.2. Métodos indiretos

Existem alternativas aos métodos diretos para solucionar sistemas lineares, como os métodos iterativos. Ao contrário dos métodos diretos, que fornecem soluções exatas, a solução obtida por meio dos métodos iterativos é uma aproximação que pode se aproximar da solução real conforme aumentamos o número de iterações. Os métodos iterativos são iniciados com uma estimativa preliminar para a solução e aprimoram a solução a cada iteração.

Dois exemplos de métodos iterativos são os métodos de Jacobi e Gauss-Seidel. A principal diferença entre eles está no modo como os valores calculados são atualizados: no método de Jacobi, os valores das incógnitas são atualizados apenas após o fim de cada iteração, enquanto no método de Gauss-Seidel, o valor de cada incógnita é atualizado assim que uma nova estimativa é calculada.

Os métodos indiretos para solução de sistemas lineares apresentam características distintas e comparados aos métodos diretos. Por exemplo, enquanto os métodos diretos possuem convergência garantida por serem processos finitos, os métodos iterativos têm convergência assegurada apenas sob certas condições, portanto é importante avaliar essas condições antes de utilizar os métodos de Jacobi e Gauss-Seidel.

"Os métodos iterativos, como o método de Gauss-Seidel, são mais adequados para sistemas grandes, pois seu tempo de cálculo é proporcional ao tamanho da matriz. No entanto, sua convergência pode ser lenta para sistemas mal condicionados" [5]. No entanto, podem ser usados mesmo quando a matriz é singular ou mal condicionada, enquanto os métodos diretos podem falhar nesses casos [1].

"Os métodos diretos são eficientes para sistemas relativamente pequenos, em que o custo computacional é justificável pela precisão. Para sistemas maiores e mais esparsos, os métodos iterativos geralmente são mais eficientes" [2].

Além disso, os métodos iterativos são menos suscetíveis ao acúmulo de erros de arredondamento que podem ocorrer em métodos diretos. Isso ocorre porque os erros de arredondamento são acumulados durante a aplicação das transformações elementares sobre as equações do sistema, o que não ocorre nos métodos iterativos, onde apenas a solução corrente é sujeita a erros.

5.3. Tempo de performance

Aqui, vamos avaliar o desempenho dos métodos apresentados e determinar qual se destaca em termos de tempo de execução.

Usando como exemplo o seguinte sistema:

$$\begin{aligned} 4x + 3y + 2z &= 1 \\ x + 5y + 7z &= 1 \\ x + y + 2z &= 1 \end{aligned} \tag{11}$$

Faremos um teste de desempenho usando esta matriz para comparar os métodos e analisar o tempo de execução de cada um. Para isso, usamos a ferramenta **MATLAB** com os comandos **Profile on** e **profile viewer** para monitorar a *performance*. O método de Jacobi não apresentou convergência para este exemplo, portanto, não faremos comparações de desempenho para tal método. A Figura 4 mostra a tabela de desempenho obtida:

Nome da função	Chamadas	Tempo Total	Tempo Próprio	Gráfico de tempo total
Gauss	1	0.004	0.004	
Gauss-Seidel	1	0.005	0.005	
Gauss-Jordan	1	0.006	0.006	
LU	1	0.055	0.055	

Figura 4: Tempo de *performance* de cada método

Esta tabela mostra o tempo necessário para executar cada método no exemplo acima, o que é um bom indicador de comparação entre eles. Podemos notar que não houve diferenças significativas entre os métodos de Gauss e Gauss-Jordan, ambos apresentaram tempo de execução relativamente próximo. O método indireto de Gauss-Seidel apresentou tempo de processamento muito semelhante aos métodos diretos neste exemplo. No entanto, o método de decomposição LU exibiu o maior tempo de execução entre todos os métodos, sendo maior até do que a soma dos tempos de execução dos outros métodos.

5.4. Tempo de Performance em relação ao tamanho da matriz

Será discutido agora o desempenho de diferentes métodos numéricos para a resolução de sistemas lineares de diferentes tamanhos. Embora o exemplo anteriormente apresentado seja de um sistema de três equações e três incógnitas, é comum trabalhar com sistemas maiores e, para comparar o tempo de execução entre os métodos, é necessário considerar o tamanho do sistema e como cada método comporta-se ao trabalhar com sistemas de ordens mais elevadas.

Utilizamos a mesma ferramenta anterior para encontrar o tempo de execução para monitoramento de desempenho e a função *randi(iMAX,m,n)* para gerar matrizes de tamanho m, n, onde m são as linhas e n as colunas da matriz, onde **iMAX** é o valor máximo que o número randômico gerado pode assumir.

A Figura 5 apresenta uma comparação entre cada método em relação ao tempo médio gasto para resolver sistemas de n equações. O gráfico permite verificar que, ao aumentar o tamanho da matriz, o tempo de execução para o método de Gauss-Jordan varia pouco em comparação a outros métodos, sendo este que resolve o sistema no menor tempo possível até a matriz de tamanho n = 200

analisada. Isso se deve à diagonalização da matriz, que já fornece a solução sem necessidade de processos retroativos ou iterações indeterminadas.

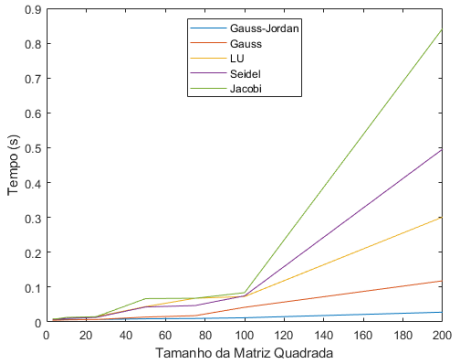

Figura 5: Tempo de *performance* de cada método em relação ao tamanho da matriz

Ao analisarmos o gráfico, é possível verificar quanto tempo, em média, cada método leva para solucionar um sistema de tamanho n, considerando n o tamanho da matriz quadrada. O método de Gauss-Jordan apresentou um desempenho superior em relação aos métodos indiretos, com o menor tempo de execução possível até a matriz de tamanho n = 200 analisada, devido à diagonalização da matriz que já fornece a solução sem necessidade de processos retroativos ou iterações indeterminadas.

Para avaliar com precisão o tempo de desempenho em relação ao tamanho da matriz, utilizamos ferramentas computacionais capazes de cronometrar o tempo de execução de um algoritmo. No caso deste estudo, empregamos o *software* Matlab e seus comandos "tic" para iniciar a contagem do tempo, e "toc" para finalizá-la e obter o tempo decorrido.

No entanto, é importante lembrar que o tempo de *performance* pode ser influenciado por diversos fatores, tais como o *hardware* do computador utilizado, a implementação do algoritmo e a precisão da solução obtida. Portanto, é necessário interpretar os resultados obtidos com cautela, já que eles podem variar em diferentes situações.

6. Conclusão

Os métodos numéricos vistos até aqui são fundamentais para resolver problemas complexos em diversas áreas do conhecimento, como engenharia, física, finanças, entre outras. Ao analisar o desempenho desses métodos, é possível observar que cada um apresenta vantagens e desvantagens, dependendo do problema em questão.

No caso da solução de sistemas lineares, vimos que os métodos diretos, como a eliminação de Gauss e a decomposição LU, são mais eficientes para sistemas pequenos ou com pouca esparsidade. Já os métodos indiretos, como o método de Jacobi e Gauss-Seidel, são mais indicados para sistemas grandes e esparsos, além de serem mais simples de implementar. Por fim, o método de Gauss-Jordan destaca-se por ter o menor tempo de execução até para sistemas grandes.

É importante lembrar que a escolha do método mais adequado depende do contexto em que o problema insere-se, bem como dos recursos computacionais disponíveis e do nível de precisão re-

querido. Além disso, há outros fatores a considerar, como a estabilidade numérica e a possibilidade de utilização de técnicas de paralelização para acelerar a solução do problema.

Em resumo, a análise do desempenho dos métodos numéricos estudados permite-nos entender suas características e aplicabilidades, contribuindo para a escolha adequada do método em diferentes situações e para o desenvolvimento de soluções eficientes e precisas em problemas reais.

Referências

[1] Ruggiero, M. & Lopes, V. *Cálculo numérico: aspectos teóricos e computacionais*. (Makron Books do Brasil,1997)

[2] Chapra, S. & Canale, R. *Métodos Numéricos para Engenharia*-7ª Edição. (McGraw Hill Brasil,2016)

[3] Bisong, Ekaba . *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*. (Springer,2019)

[4] Canuto, C. G., & Kenig, C. E. (2011). *Análise Numérica: uma introdução*. Rio de Janeiro: LTC.

[5] Kincaid, D., & Cheney, W. (2002). *Análise Numérica*. LTC.

Valdex de J. Santos
Instituto Federal da Bahia
<valdexsantos@ifba.edu.br>

Danton R. da Costa Pitombo
Instituto Federal da Bahia
<dantonpitombo2015.1@gmail.com>

Marinilton J. O. Lima júnior
Instituto Federal da Bahia
<marinilton77@gmail.com>

Lucas R. de Souza Meira
Instituto Federal da Bahia
<lukas.meiira@gmail.com>

Victor Bastos G. Moura
Instituto Federal da Bahia
<victor.moura@ifba.edu.br>

Recebido: 18/04/2023
Publicado: 20/02/2024