UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS

INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

UTILIZAÇÃO DA LINGUAGEM DE PROGRAMAÇÃO C/C++ NO ESTUDO E ENSINO DE ALGUNS TÓPICOS DA ÁLGEBRA LINEAR

MOSSORÓ
2021

HIAN ALVES DANTAS

UTILIZAÇÃO DA LINGUAGEM DE PROGRAMAÇÃO C/C++ NO ESTUDO E ENSINO DE ALGUNS TÓPICOS DA ÁLGEBRA LINEAR

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Ciência e Tecnologia.

Orientador: Prof. Dr. Paulo César Linhares Da Silva

Co-orientadora: Profª Dra. Maria Joseane Felipe Guedes Mâcedo.

MOSSORÓ

2021

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central Orlando Teixeira (BCOT)
Setor de Informação e Referência (SIR)

D192u Dantas, Hian Alves.
Utilização da linguagem de programação C/C++ no estudo e ensino de alguns tópicos da Álgebra Linear / Hian Alves Dantas. - 2021.
83 f. : il.

Orientador: Paulo Da Silva.
Coorientadora: Maria Mâcedo.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2021.

1. Álgebra Linear. 2. Programação em C/C++. 3. Algoritmos. I. Da Silva, Paulo , orient. II. Mâcedo, Maria , co-orient. III. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

HIAN ALVES DANTAS

UTILIZAÇÃO DA LINGUAGEM DE PROGRAMAÇÃO C/C++ NO ESTUDO E ENSINO DE ALGUNS TÓPICOS DA ÁLGEBRA LINEAR

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Ciência e Tecnologia.

Defendida em: 18 /11 / 2021.

BANCA EXAMINADORA

_____
Prof. Dr Paulo César Linhares Da Silva (UFERSA)
Presidente

_____
Profª. Dra. Maria Joseane Felipe Guedes Mâcedo (UFERSA)
Coorientadora

_____
Profª. Dra. Antônia Jocivania Pinheiro (UFERSA)
Membro Examinador

_____

AGRADECIMENTOS

Agradeço aos meus pais que sempre me incentivaram em meus estudos, e me apoiaram nas horas difíceis. Agradeço aos meus amigos e colegas de curso, que sempre estiveram presentes compartilhando as dificuldades passadas ao decorrer do curso. Agradeço a todos os meus professores, que sempre tiveram paciência, didática, e empenho para me ensinar os conteúdos necessários. Agradeço também aos meus professores orientadores, pelas correções, pelo tempo disponível. E agradeço a todos que diretamente ou indiretamente fizeram parte de minha formação acadêmica.

Não há estrada real para a ciência e só têm possibilidade de chegar aos seus cumes luminosos aqueles que não temem fatigar-se a escalar as suas veredas escarpadas.

Karl Marx

RESUMO

Este trabalho trata de uma metodologia para os que estudam, ensinam e pesquisam tópicos relacionados com álgebra linear. Tal metodologia, eficaz na abordagem do estudo de assuntos de Álgebra Linear, faz uso de algoritmos numéricos que abordam problemas sobre vetores, operações com vetores, matrizes e suas operações, cálculo de determinantes, inversão de matrizes e solução de sistemas lineares. Estes algoritmos são implementados e tratados fazendo o uso da linguagem de programação C\C++. Isto cria um mecanismo capaz de tornar o aprendizado deste assunto mais prático, visando encontrar mais aplicabilidades à tópicos de álgebra linear direcionado para problemas em ciências exatas e tecnológicas. Ao fim do trabalho, é mostrado os algoritmos trabalhados e implementados na linguagem de programação C/C++.

Palavras-chave: Álgebra Linear. Programação em C/C++. Algoritmos.

ABSTRACT

This work deals with a methodology for those who study, teach and research topics related to linear algebra. Such methodology Such methodology is effective in approaching the study of linear algebra subjects, makes use of numerical algorithms that address problems on vectors, operations with vectors, matrices and their operations, calculation of determinants, inversion of matrices and solution of linear systems. These algorithms are implemented and treated using the C\C++ programming language. This creates a mechanism capable of making the learning of this subject more practical, aiming to find more applicability to linear algebra topics directed to problems in exact and technological sciences. At the end of the work, the algorithms worked and implemented in the C/C++ programming language are shown.

Keywords: Linear algebra. Programming in C/C++. Algorithms.

SUMÁRIO

1. INTRODUÇÃO

A tecnologia é um fator fundamental para o avanço da humanidade, e na educação não é diferente, a principal função da tecnologia na educação é facilitar o aprendizado do aluno diante do conteúdo ministrado (DE MELO e ZANONI, 2018). A disciplina de Álgebra Linear em geral é uma disciplina inicial nos cursos de engenharia, computação, matemática, e outros cursos da área de exatas e tecnológicas, e corriqueiramente é pouco compreendida pelos estudantes por requerer um alto nível de abstração, e pouco ser mostrado sua aplicação. Entretanto a álgebra linear está presente em vários problemas de engenharia e tecnologia, como a resolução de circuitos elétricos, modelagem e otimização, criptografia, balanceamento das reações químicas, cálculo das equações de força no dimensionamento de estruturas, além de ser fundamental na computação gráfica, onde cada pixels de uma imagem gerada na tela de um computador são representados por matrizes formadas por zero e um (PEREIRA e COSTA, 2017).

A programação torna-se uma ferramenta importante para a álgebra linear, visto que os computadores se tornaram indispensáveis em diversos trabalhos, principalmente para as engenharias, pois problemas reais geralmente envolvem várias variáveis, tornando o seu cálculo extremamente extenso e monótono. Dessa forma saber programar algoritmos que resolvam esses problemas é fundamental para o exercício da profissão dos futuros engenheiros, além de fazer a ponte entre o conteúdo ministrado e sua aplicação.

As linguagens C/C++ estão intrinsicamente ligadas, pois C é a fundação em que o C++ foi construído. A linguagem C está contida dentro do C++ de modo que é possível usar quase toda a sua lógica de programação na linguagem C++. O C é tido como uma linguagem de médio nível, possuindo então as vantagens das linguagens de baixo nível como a manipulação de bits e bytes e endereçamentos, e também possui as vantagens das linguagens de alto nível como o suporte dos *data types* (SCHILDT, 2003).

Na década de 1970, Dennis Richard desenvolveu a linguagem C, essa seria a sucessora da linguagem utilizada na época, a linguagem B. Dennis a criou pois necessitava-se de uma linguagem que fosse mais veloz (otimizando o uso da memória) e portável (poderia facilmente ser adaptada de um sistema operacional para outro). A linguagem C++, foi criada em 1979 por Bjarne Stroustrup com o objetivo de ser uma expansão da linguagem C, permitindo assim que fosse feita uma programação orientada a objetos, dessa forma a programação tanto seria

organizada em função do código, como seria também organizada em função dos dados (SCHILDT, 2003).

As linguagens C/C++ possuem uma alta performance de execução, e isso as tornam a principal escolha quando se necessita realizar cálculos complexos e extensos em tempos menores, de acordo com o *ranking* Tiobe (https://www.tiobe.com/tiobe-index/, acessado em 20 de outubro de 2021), *ranking* que mensalmente faz uma estimativa das linguagens de programação mais utilizadas, baseado em seu uso por engenheiros, quantidade de buscas na internet, cursos ofertando o ensino delas, e em seu uso por terceiros. Na data de onze de outubro de 2021, a linguagem C é a segunda linguagem de programação mais utilizada no mundo, e a linguagem C++ ocupa o quarto lugar no ranking, além disso ela é a linguagem ensinada na maioria das disciplinas de lógica de programação nas faculdades de engenharia, além de ser a linguagem com a qual mais possuo familiaridade, justificando assim a minha escolha nesse trabalho.

A Álgebra Linear é entendida como uma teoria que unifica várias áreas diferentes do estudo da matemática, como análise matricial, equações diferenciais lineares, geometria, e por isso possui alto nível de abstração, observa-se também um alto grau de reprovação em nível global, e muitos alunos terminam a disciplina questionando-se se ela possui alguma serventia em sua vida profissional e acadêmica (COIMBRA, 2008). Como disciplina acadêmica, a Álgebra Linear possui uma história relativamente recente, nas universidades estadunidenses ela aparece pela primeira vez como uma disciplina oferecida apenas em 1965, e outros países como França, Itália, Chile, Irã, também possuem datas parecidas para a formalização dela como disciplina acadêmica (SANTOS, 2018), dessa forma, pode-se entender que seu ensino formal é recente na história da humanidade, tendo assim muito a ser inovado na sua forma de ensino e aprendizagem.

O objetivo desse trabalho é construir algoritmos como ferramenta de estudo, ensino e aprendizado para alunos e professores na disciplina de Álgebra Linear, buscando fornecer ferramentas metodológicas para o aluno, mostrando assim que é possível fazer a integração da matemática com a programação. Além de servir como ferramenta de ensino para o professor, podendo assim mostrar computacionalmente o uso da programação na álgebra linear. Os objetivos específicos visam desenvolver algoritmos para o desenvolvimento de Laplace, eliminação gaussiana, operações básicas entre matrizes, determinar inversa de matrizes e operações de produto escalar, produto vetorial, e implementar esses algoritmos na linguagem C/C++.

2. REVISÃO BIBLIOGRÁFICA

O estudo da Álgebra Linear é um processo conturbado para a maior parte dos alunos, e identificando esse problema, diversos professores empenharam-se em buscar e aplicar métodos inovativos que facilitem o aprendizado, 2018 os professores Milagres, Lima e Cardoso (MILAGES, LIMA e CARDOSO, 2018) que desenvolveram um software que tem como objetivo mediar esse processo. Nesse software foi desenvolvida uma calculadora usando a linguagem JavaScript, e as tecnologias HTML5 e CSS, no qual ela executa operações matriciais e mostra o passo a passo para o usuário. Em algumas etapas do processo já se percebeu um avanço em relação ao problema proposto.

> Questões sobre o grau de abstração da Álgebra Linear em relação a outros conteúdos matemáticos, cálculos repetitivos e falta de sentido nas exemplificações por meio de tabelas bidimensionais foram parcialmente sanadas nessa etapa que envolveu, além de todo aporte teórico e prático dado pelos orientadores, atividades como brainstorming e reuniões para autorreflexão. (MILAGRES; LIMA e CARDOSO, 2018, P.271)

O uso de softwares já existentes também foi uma alternativa pensada por Santana et al.(SANTANA et al., 2019), em que o GeoGebra foi utilizado na disciplina de álgebra linear, para a resolução de sistemas lineares, aplicação do método dos mínimos quadrados, e a plotagem de gráficos.

> Adotar e incentivar o uso de softwares educativos, como o GeoGebra, no ensino e estudo de conteúdos matemáticos do ensino superior contribui com a aprendizagem dos estudantes. A possibilidade de ilustrar graficamente as soluções, algumas vezes analisadas apenas de forma abstrata, e realizar cálculos complexos de forma simples e segura torna o ensino mais atrativo e interessante. (SANTANA et al., 2019, P.15104)

Em 2015, entre os meses de outubro e novembro na Universidade do Centro-Oeste, foi realizado uma metodologia de aprendizado com os alunos do segundo ano do curso de matemática a partir do uso do software WxMáxima, em que a base dessa metodologia foi uma aplicação semelhante ocorrida em 2014 com o uso do software *Pascalzim*. Nessa metodologia os alunos construíram algoritmos para a resolução de matrizes e sistemas a partir dos métodos de Gauss e Gauss-Jordan.

> Diante do estudo dos trabalhos relacionados ao *software* WxMáxima, nota-se que o uso de recursos computacionais auxiliou no processo de ensinoaprendizagem, assim como favoreceu o desenvolvimento do raciocínio lógico melhorando o desempenho nos conteúdos estudados. Com *software* é possível fazer uma análise numérica,

> algébrica e gráfica, mostrando-se viável a sua utilização na abordagem dos conteúdos e aplicável em diversas áreas de conhecimento. (MOLINARI, 2015, P.3)

3. ÁLGEBRA LINEAR: ASPECTOS TEÓRICOS

3.1. Matrizes

Matrizes são estruturas bidimensionais compostas por linhas e colunas, podendo ser iguais ou diferentes, em que cada elemento é denotado pelos índices que indicam respectivamente a linha e coluna, ou seja, "a_{ij}" em que "i" representa a linha e "j" representa a coluna, onde "i" e "j" variam de 1 até n. Denota-se matrizes por letras maiúsculas, e seus elementos por letras minúsculas. Toda matriz possui uma ordem, a ordem de uma matriz define-se pelo número de linhas e colunas que formam a matriz, a notação usada nesse trabalho para a ordem será a representação do número de linhas e o número de colunas subscrito após a letra maiúscula que denota a matriz: $\boldsymbol{A_{m\times n}}$. As matrizes são vastamente utilizadas nas ciências exatas e tecnológicas, na resolução de problemas nestas áreas. As matrizes podem ser escritas nas formas:

$$\boldsymbol{M_{m\times n}} = \begin{bmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a_{m1}} & \cdots & \boldsymbol{a_{mn}} \end{bmatrix}.$$

3.2. Operações com matrizes

As operações mais elementares são a soma e a multiplicação por escalar de matriz, a partir delas pode-se definir todas as outras operações. As operações de soma de duas matrizes podem ser efetuadas quando ambas possuem a mesma ordem e a soma dos elementos ocorre com os elementos que possuem a mesma posição. Dada as matrizes: $\boldsymbol{A_{m\times n} + B_{m\times n}}$, a soma delas resultará em uma matriz $\boldsymbol{C_{m\times n}}$.

Dada as matrizes $\boldsymbol{A_{m\times n}} = \begin{bmatrix} \boldsymbol{x_{11}} & \cdots & \boldsymbol{x_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1}} & \cdots & \boldsymbol{x_{mn}} \end{bmatrix}$, e $\boldsymbol{B_{m\times n}} = \begin{bmatrix} \boldsymbol{y_{11}} & \cdots & \boldsymbol{y_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y_{m1}} & \cdots & \boldsymbol{y_{mn}} \end{bmatrix}$, a soma destas resultará em uma nova matriz $\boldsymbol{C_{m\times n}} = \begin{bmatrix} \boldsymbol{z_{11}} & \cdots & \boldsymbol{z_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{z_{m1}} & \cdots & \boldsymbol{z_{mn}} \end{bmatrix}$, tem-se a expressão: $\boldsymbol{C = A + B}$

$$\begin{bmatrix} \boldsymbol{z_{11}} & \cdots & \boldsymbol{z_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{z_{m1}} & \cdots & \boldsymbol{z_{mn}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_{11}} & \cdots & \boldsymbol{x_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1}} & \cdots & \boldsymbol{x_{mn}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{y_{11}} & \cdots & \boldsymbol{y_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y_{m1}} & \cdots & \boldsymbol{y_{mn}} \end{bmatrix}$$

onde $z_{ij} = x_{ij} + y_{ij}$, para todos $1 \leq i \leq m$ e $1 \leq j \leq n$.

Exemplo 1: Dada duas matrizes quadradas de ordem 2×2, sendo a matriz $\boldsymbol{A_{2\times2}} = \begin{bmatrix} \boldsymbol{-9} & \boldsymbol{2} \\ \boldsymbol{5} & \boldsymbol{0} \end{bmatrix}$ e a matriz $\boldsymbol{B_{2\times2}} = \begin{bmatrix} \boldsymbol{5} & \boldsymbol{-1} \\ \boldsymbol{-2} & \boldsymbol{2} \end{bmatrix}$ e a soma entre essas duas matrizes dar-se-á:

$$\begin{bmatrix} \boldsymbol{-9} & \boldsymbol{2} \\ \boldsymbol{5} & \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{5} & \boldsymbol{-1} \\ \boldsymbol{-2} & \boldsymbol{2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{-4} & \boldsymbol{1} \\ \boldsymbol{3} & \boldsymbol{2} \end{bmatrix}.$$

Dada as matrizes A e B de mesma ordem, a adição de matrizes possui as seguintes propriedades:

- Comutatividade: Independentemente da ordem dos operandos o resultado será o mesmo, $\boldsymbol{A + B = B + A}$.
- Associatividade: $\boldsymbol{A + (B + C) = (A + B) + C}$.
- Existência do elemento neutro: $\boldsymbol{A + 0 = A}$, onde $\boldsymbol{0}$ é a matriz nula.

Transposição de matriz é uma operação realizada onde transforma-se as linhas de uma matriz em colunas e as colunas em linhas, obtendo uma nova matriz chamada de matriz transposta. Para obter a transposta de uma matriz qualquer $\boldsymbol{A[a_{ij}]_{m\times n}}$, deve-se obter uma matriz cujo as linhas da nova matriz será as colunas da matriz a ser transposta, assim, dada a matriz $\boldsymbol{A[a_{ij}]_{m\times n}}$, quando transposta tornar-se-á a matriz $\boldsymbol{A'[b_{ij}]_{n\times m}}$, em que $\boldsymbol{b_{ij} = a_{ji}}$.

$$\boldsymbol{A_{m\times n}} = \begin{bmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a_{m1}} & \cdots & \boldsymbol{a_{mn}} \end{bmatrix} \rightarrow \boldsymbol{A'_{m\times n}} = \begin{bmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{m1}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a_{1n}} & \cdots & \boldsymbol{a_{mn}} \end{bmatrix}$$

Exemplo 2: Dada a matriz $\boldsymbol{A_{3\times2}} = \begin{bmatrix} \boldsymbol{5} & \boldsymbol{6} \\ \boldsymbol{4} & \boldsymbol{8} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix}$, sua transposta dar-se-á:

$$\boldsymbol{A_{3\times2}} = \begin{bmatrix} \boldsymbol{5} & \boldsymbol{6} \\ \boldsymbol{4} & \boldsymbol{8} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix} \rightarrow \boldsymbol{A'_{2\times3}} = \begin{bmatrix} \boldsymbol{5} & \boldsymbol{4} & \boldsymbol{1} \\ \boldsymbol{6} & \boldsymbol{8} & \boldsymbol{0} \end{bmatrix}.$$

Seja “h” um escalar e $A = [a_{ij}]$ uma matriz. A multiplicação do escalar “**h**” pela matriz $\boldsymbol{A = [a_{ij}]}$, resultará em uma nova matriz: $\boldsymbol{h \times A_{m\times n} = [ha_{ij}]}$

$$\boldsymbol{h \times \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} h \times a_{11} & \cdots & h \times a_{1n} \\ \vdots & \ddots & \vdots \\ h \times a_{m1} & \cdots & h \times a_{mn} \end{bmatrix}}.$$

Exemplo 3: Dada a matriz $\boldsymbol{A_{2\times 2} = \begin{bmatrix} -2 & 8 \\ 5 & 1 \end{bmatrix}}$, a multiplicação da matriz **A** pelo escalar 3 dar-se-á:

$$\boldsymbol{A_{2\times 2} = \begin{bmatrix} -2 & 8 \\ 5 & 1 \end{bmatrix} \rightarrow A_{2\times 2} \times 3 = \begin{bmatrix} -2 & 8 \\ 5 & 1 \end{bmatrix} \times 3 = \begin{bmatrix} -6 & 24 \\ 15 & 3 \end{bmatrix}}.$$

A partir das definições das operações de soma e multiplicação por escalar, pode-se definir a operação de subtração entre duas matrizes, matriz A e matriz B como: $\boldsymbol{A - B = A + (-1) \times B}$.

Para a multiplicação entre duas matrizes é necessário obedecer a condição: o número de colunas da primeira matriz deve ser igual ao número de linhas da segunda (BOLDRINI, 1980). Multiplicar-se-á os elementos da linha da primeira matriz pelos elementos da primeira coluna da segunda matriz, em seguida, somam-se os resultados da multiplicação encontrando o valor do primeiro elemento, assim sucessivamente até formar a matriz resultante. A matriz resultante terá o número de linhas da primeira matriz, e o número de colunas da segunda matriz.

Propriedades da multiplicação dada as matrizes A, B e C:

- Distributiva: $\boldsymbol{A(B + C) = AB + AC}$, o produto da matriz é distributivo em relação a soma, dado que a ordem da matriz B e C são iguais, e o número de colunas de A é igual ao número de linhas de B e de C.
- Associatividade: $\boldsymbol{(AB)C = A(BC)}$, dado que o número de colunas da matriz A é igual ao número de linhas da matriz B, e o número de colunas da matriz B é igual ao número de linhas da matriz C.
- Não comutativa: $\boldsymbol{A \times B \neq B \times A}$, a ordem da multiplicação entre as duas matrizes altera o resultado. Dado $A_{m\times n}$ e $B_{n\times p}$, a multiplicação $\boldsymbol{A \times B}$ pode ser efetuada, porém $\boldsymbol{B \times A}$ não pode ser efetuada. Caso **A** e **B** possuam a mesma ordem, o resultado de $\boldsymbol{A \times B}$ e $\boldsymbol{B \times A}$, serão diferentes.

- Transposição: $(AB)^T = A^T \times B^T$, a transposição da multiplicação entre a matriz A e a matriz B, é igual a multiplicação da transposta da matriz A pela transposta da matriz B
- Elemento neutro:$\boldsymbol{A \times 1 = A}$
- Elemento nulo: $\boldsymbol{A \times 0 = 0 \times A = 0}$.

Dada as matrizes $\boldsymbol{B_{m\times n}} = \begin{bmatrix} \boldsymbol{x_{11}} & \cdots & \boldsymbol{x_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1}} & \cdots & \boldsymbol{x_{mn}} \end{bmatrix}$, $\boldsymbol{C_{m\times n}} = \begin{bmatrix} \boldsymbol{y_{11}} & \cdots & \boldsymbol{y_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y_{m1}} & \cdots & \boldsymbol{y_{mn}} \end{bmatrix}$ e a matriz D como o resultado da multiplicação da matriz B pela matriz C, tem-se: $\boldsymbol{B \times C = D}$

$$\begin{bmatrix} \boldsymbol{x_{11}} & \cdots & \boldsymbol{x_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1}} & \cdots & \boldsymbol{x_{mn}} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{y_{11}} & \cdots & \boldsymbol{y_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y_{m1}} & \cdots & \boldsymbol{y_{mn}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_{11} \times y_{11} + \cdots + x_{1n} \times y_{m1}} & \cdots & \boldsymbol{x_{11} \times y_{1n} + \cdots + x_{1n} \times y_{mn}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1} \times y_{11} + \cdots + x_{mn} \times y_{m1}} & \cdots & \boldsymbol{x_{m1} \times y_{1n} + \cdots + x_{mn} \times y_{mn}} \end{bmatrix}.$$

Exemplo 4: Dadas duas matrizes $\boldsymbol{B_{3\times 2}} = \begin{bmatrix} \boldsymbol{2} & \boldsymbol{0} \\ \boldsymbol{1} & \boldsymbol{-5} \\ \boldsymbol{4} & \boldsymbol{1} \end{bmatrix}$ e $\boldsymbol{C_{2\times 2}} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{2} \\ \boldsymbol{2} & \boldsymbol{3} \end{bmatrix}$, a multiplicação de $B \times C$, dar-se-á:

$$\boldsymbol{B \times C} = \begin{bmatrix} \boldsymbol{2 \times 1 + 0 \times 2} & \boldsymbol{2 \times 2 + 0 \times 3} \\ \boldsymbol{1 \times 1 + (-5) \times 2} & \boldsymbol{1 \times 2 + (-5) \times 3} \\ \boldsymbol{4 \times 1 + 1 \times 2} & \boldsymbol{4 \times 2 + 1 \times 3} \end{bmatrix} \rightarrow \boldsymbol{B \times C} = \begin{bmatrix} \boldsymbol{2} & \boldsymbol{4} \\ \boldsymbol{-9} & \boldsymbol{-13} \\ \boldsymbol{6} & \boldsymbol{11} \end{bmatrix}.$$

3.3. Determinante

O determinante é um número associado a uma matriz quadrada. Para entender o conceito do cálculo do determinante é necessário entender o que é uma permutação; uma permutação é a mudança de ordem de uma determinada sequência, como por exemplo a sequência: $\boldsymbol{x\ y\ z}$, pode ser permutada para formar a sequencia $\boldsymbol{x\ z\ y}$, de modo que existem 6 permutações possíveis para essa sequencia de 3 elementos. Quando dois elementos estão em ordem diferente da sequencia original, diz-se que eles formam uma inversão. Na permutação realizada anteriormente, existe 1 inversão, pois o "z" está localizando antes do "y", uma permutação pode ser classificada como classe par ou classe ímpar, de acordo com o número de inversões. No exemplo anterior foi encontrada uma inversão, logo essa permutação é de classe ímpar, a sequência original possui zero inversões, assim como o número de inversões é par, a permutação é de classe par.

O produto dos termos da diagonal principal dá-se o nome de termo principal, e o produto dos termos da diagonal secundária dá-se o nome de termo secundário, de forma que o produto dos termos da diagonal principal se encontra da seguinte forma: $\boldsymbol{a_{11} \times a_{22} \times a_{33} \times \ldots \times a_{nn}}$;

e os produto dos termos da diagonal secundaria encontra se da seguinte forma: $\boldsymbol{a_{1\,n} \times a_{2\,n-1} \times a_{3\,n-2} \times \ldots \times a_{n\,1}}$.

> Chama-se determinante de uma matriz quadrada à soma algébrica dos produtos que se obtém efetuando todas as permutações dos segundos índices do termo principal, fixados os primeiros índices, e fazendo-se preceder os produtos do sinal + ou -, conforme a permutação dos segundos índices seja de classe par ou de classe ímpar. (STEINBRUCH e WINTERLE, 1987, P.421).

O determinante de uma matriz pode ser dado pela fórmula de Leibniz:

$$\boldsymbol{detA = \sum_{\sigma \epsilon S_n} \left(sgn(\sigma) \prod_{i=1}^{n} a_i\,, \sigma_i \right)},$$

em que $\boldsymbol{\sigma}$ é a função da permutação que reordena a sequência de 1 até n, o valor da posição $\boldsymbol{i}$ após a reordenação por $\boldsymbol{\sigma}$ é dado por $\boldsymbol{\sigma_i}$, $\boldsymbol{S_n}$ é a sequência de todas as permutações, $\boldsymbol{sgn(\sigma)}$ é a função sinal, que conferirá sinal positivo para um número de permutações pares, e sinal negativo para um número de permutações ímpares.

Para o caso de uma matriz quadrada de qualquer ordem, o determinante pode ser calculado pelo desenvolvimento de Laplace, onde o determinante da matriz será expresso em função do determinante das submatrizes. O desenvolvimento de Laplace foi o método escolhido para o cálculo do determinante nesse trabalho.

Exemplo 5: Dada uma matriz $\boldsymbol{A_{2\times2} = \begin{bmatrix} 5 & 6 \\ 4 & 8 \end{bmatrix}}$, o cálculo de seu determinante dar-se-á:

$$\boldsymbol{A_{2\times2} = \begin{bmatrix} 5 & 6 \\ 4 & 8 \end{bmatrix} \rightarrow A_{2\times2} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}$$

$$\rightarrow (termo\ principal)\ a_{11} \times\ a_{22} \quad (termo\ secundario)\ \ a_{12} \times a_{21} \ \rightarrow$$

$$\boldsymbol{\rightarrow\ +(a_{11} \times\ a_{22}) - (a_{12} \times a_{21}) \rightarrow\ +(5 \times\ 8) - (6 \times 4) = 16}$$

$$\boldsymbol{detA = 16}.$$

Observando que o termo principal não possui nenhuma inversão, e o termo secundário possui uma inversão, colocar-se-á seus respectivos sinais de + e -.

Método de Laplace:

Dada uma matriz quadrada **A** de ordem 3:

$$\boldsymbol{A_{3\times3}} = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \boldsymbol{a_{13}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} & \boldsymbol{a_{23}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{32}} & \boldsymbol{a_{33}} \end{bmatrix},$$

seu determinante pode ser calculado da seguinte forma:

$$\boldsymbol{det}\begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \boldsymbol{a_{13}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} & \boldsymbol{a_{23}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{32}} & \boldsymbol{a_{33}} \end{bmatrix} = \boldsymbol{a_{11}} \times \boldsymbol{det}\begin{bmatrix} \boldsymbol{a_{22}} & \boldsymbol{a_{23}} \\ \boldsymbol{a_{32}} & \boldsymbol{a_{33}} \end{bmatrix} - \boldsymbol{a_{12}} \times \boldsymbol{det}\begin{bmatrix} \boldsymbol{a_{21}} & \boldsymbol{a_{23}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{33}} \end{bmatrix} +$$

$$\boldsymbol{a_{13}} \times \boldsymbol{det}\begin{bmatrix} \boldsymbol{a_{21}} & \boldsymbol{a_{22}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{32}} \end{bmatrix}.$$

O determinante da matriz **A** pode ser expresso em função das matrizes dos cofatores. O cofator é um complemento algébrico relativo a um elemento de uma matriz quadrada, em que a partir desse elemento elimina-se a linha e a coluna da matriz de acordo com a posição dele, assim a matriz do cofator A_{ij}, é a matriz eliminando a i-enésima linha e a j-enésima coluna, calcula-se o cofator a partir da fórmula: $\Delta_{ij} = (-1)^{i+j} \times detA_{ij}$. A matriz dos cofatores será uma matriz gerada pelos cofatores da matriz original.

$$\boldsymbol{detA} = \boldsymbol{a_{11}}\Delta_{\boldsymbol{11}} + \boldsymbol{a_{12}}\Delta_{\boldsymbol{12}} + \boldsymbol{a_{13}}\Delta_{\boldsymbol{13}}.$$

3.4. Matriz inversa

O cálculo da inversa de uma matriz quadrada depende do cálculo do determinante, que só é possível se ele for diferente de zero. Dada uma matriz A quadrada de ordem "n", a sua inversa é dada por:

$$\boldsymbol{A^{-1}} = \frac{\boldsymbol{1}}{\boldsymbol{detA}}\boldsymbol{C^T},$$

em que $\boldsymbol{A^{-1}}$ é a matriz invertida, $\boldsymbol{detA}$ é o determinante da matriz, $\boldsymbol{C^T}$ é a transposta da matriz dos cofatores, também chamada de matriz adjunta.

A matriz inversa pode ser definida como: Sendo a matriz A dada como $\boldsymbol{A_{m\times n}} =$

$$\begin{bmatrix} \boldsymbol{x_{11}} & \cdots & \boldsymbol{x_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{m1}} & \cdots & \boldsymbol{x_{mn}} \end{bmatrix} \text{ e } \boldsymbol{A^{-1}_{m\times n}} = \begin{bmatrix} \boldsymbol{y_{11}} & \cdots & \boldsymbol{y_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y_{m1}} & \cdots & \boldsymbol{y_{mn}} \end{bmatrix},$$

tem-se a expressão $\boldsymbol{A \times A^{-1} = I}$

$$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \times \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mn} \end{bmatrix} = \boldsymbol{I},$$

em que **I** é uma matriz identidade.

Propriedades:

- Unicidade: a matriz inversa é única.
- Se uma matriz possui inversa, a sua transposta também possui inversa.
- Se duas matrizes de mesma ordem são invertíveis, o resultado de sua multiplicação será uma matriz invertível.

Pode-se calcula a inversa de uma matriz com a seguinte formula: $\boldsymbol{A^{-1} = \frac{1}{detA} C^T}$, onde $\boldsymbol{C^T}$ é a matriz inversa dos cofatores.

Exemplo 6: Dada a matriz $\boldsymbol{A_{2\times2}} = \begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix}$, o cálculo da sua inversa, dar-se-á:

$$\boldsymbol{A_{2\times2}} = \begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix} \rightarrow \boldsymbol{A^{-1} = \frac{1}{detA} C^T} \quad \rightarrow \boldsymbol{detA = (7 \times 1) - (3 \times 2)} \rightarrow \boldsymbol{detA = 1}$$

$$\boldsymbol{\Delta_{11} = (-1)^{1+1} \times 1}$$

$$\boldsymbol{\Delta_{11} = 1}$$

$$\boldsymbol{\Delta_{12} = (-1)^{1+2} \times 2}$$

$$\boldsymbol{\Delta_{12} = -2}$$

$$\boldsymbol{\Delta_{21} = (-1)^{2+1} \times 3}$$

$$\boldsymbol{\Delta_{12} = -3}$$

$$\boldsymbol{\Delta_{22} = (-1)^{2+2} \times 7}$$

$$\boldsymbol{\Delta_{22} = 7}$$

$$\boldsymbol{C_{2\times2}} = \begin{bmatrix} \Delta_{11} & \Delta_{12} \\ \Delta_{21} & \Delta_{22} \end{bmatrix}$$

$$\boldsymbol{C_{2\times2}} = \begin{bmatrix} \mathbf{1} & \mathbf{-2} \\ \mathbf{-3} & \mathbf{7} \end{bmatrix} \rightarrow \boldsymbol{C^{T}_{2\times2}} = \begin{bmatrix} \mathbf{1} & \mathbf{-3} \\ \mathbf{-2} & \mathbf{7} \end{bmatrix}$$

$$\boldsymbol{A^{-1}_{2\times2}} = \frac{\mathbf{1}}{\mathbf{1}}\begin{bmatrix} \mathbf{1} & \mathbf{-3} \\ \mathbf{-2} & \mathbf{7} \end{bmatrix} \rightarrow \boldsymbol{A^{-1}_{2\times2}} = \begin{bmatrix} \mathbf{1} & \mathbf{-3} \\ \mathbf{-2} & \mathbf{7} \end{bmatrix}.$$

3.5. Produto escalar

O produto escalar entre dois vetores é uma operação que retoma um número real. Dados dois vetores de mesma dimensão o produto escalar é definido como:

$$(\boldsymbol{x_1, x_2, \dots, x_n}) \cdot (\boldsymbol{y_1, y_2, \dots, y_n}) = (\boldsymbol{x_1 \times y_1 + \ x_2 \times y_2 + \cdots + \ x_n \times y_n}) \ .$$

As propriedades do produto escalar são:

- Comutatividade: $\boldsymbol{A \cdot B = B \cdot A}$, o produto escalar é comutativo, assim, a ordem dos vetores na operação não influência no resultado do produto escalar.
- Distributiva: $\boldsymbol{A \cdot (B + C) = A \cdot B + A \cdot C}$, o produto escalar é distributivo em relação a soma.
- Associativa: $\boldsymbol{(A \cdot B) \cdot C = A \cdot (B \cdot C)}$.

Exemplo 7: Dado dois vetores $\boldsymbol{\vec{u} = (5\ , 8)}$ e $\boldsymbol{\vec{v} = (3\ , 1)}$, o produto escalar entre eles dar-se-á:

$$\boldsymbol{\vec{u} \cdot \vec{v} = (5 \times 3) + (8 \times 1) \ \rightarrow \ \vec{u} \cdot \vec{v} = \ 23}$$

3.6. Produto vetorial

O produto vetorial é uma operação entre dois vetores, neste trabalho restrito a um espaço vetorial tridimensional $\mathbb{R}^3$, que retorna outro vetor perpendicular aos vetores iniciais. Caso os vetores sejam linearmente dependentes, ou seja, se eles forem combinações lineares um do outro, o produto vetorial resultará em um vetor nulo. Para calcular de maneira algébrica o produto vetorial, é necessário realizar o cálculo de um determinante, onde a matriz será composta pelos dois vetores e por um versor.

$$\boldsymbol{\vec{u} = (x_1, x_2, x_3)} \ ; \ \boldsymbol{\vec{v} = (y_1, y_2, y_3)}$$

$$\vec{u} \times \vec{v} = \boldsymbol{det}\begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \boldsymbol{x_3} \\ \boldsymbol{y_1} & \boldsymbol{y_2} & \boldsymbol{y_3} \end{bmatrix}$$

$$\vec{u} \times \vec{v} = [\boldsymbol{i}(\boldsymbol{x_2} \times \boldsymbol{y_3}) + \boldsymbol{j}(\boldsymbol{x_3} \times \boldsymbol{y_1}) + \boldsymbol{k}(\boldsymbol{x_1} \times \boldsymbol{y_2})] - [\boldsymbol{i}(x_3 \times \boldsymbol{y_2}) + \boldsymbol{j}(\boldsymbol{x_1} \times \boldsymbol{y_3}) + \boldsymbol{k}(\boldsymbol{y_1} \times \boldsymbol{x_2})]$$

As propriedades do produto vetorial são:

- O produto vetorial entre dois vetores iguais é 0: $\vec{a} \times \vec{a} = 0$.
- Anticomutativa: A ordem dos vetores na operação muda o resultado. $\vec{a} \times \vec{b} = -(\vec{b} \times \vec{a})$.
- Distributiva em relação a adição: $\vec{a} \times (\vec{b} + \vec{c}) = (\vec{a} \times \vec{b}) + (\vec{a} \times \vec{c})$.
- Não associativa: $\vec{a} \times (\vec{b} \times \vec{c}) \neq (\vec{a} \times \vec{b}) \times \vec{c}$.

Exemplo 8: Dado dois vetores $\vec{u} = (\mathbf{2}, -\mathbf{1}, \mathbf{3})$ e $\vec{v} = (\mathbf{5}, \mathbf{1}, \mathbf{3})$, o produto vetorial entre eles dar-se-á:

$$\vec{u} \times \vec{v} = \boldsymbol{det}\begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \mathbf{2} & -\mathbf{1} & \mathbf{3} \\ \mathbf{5} & \mathbf{1} & \mathbf{3} \end{bmatrix}$$

$$\vec{u} \times \vec{v} = -\mathbf{6}\boldsymbol{i} + \mathbf{9}\boldsymbol{j} + \mathbf{7}\boldsymbol{k} \rightarrow \vec{u} \times \vec{v} = (-\mathbf{6}, \mathbf{9}, \mathbf{7}) .$$

3.7. Sistemas lineares

Um sistema de equações é um conjunto de equações que podem possuir ou não relação umas com as outras. Sistema linear é um conjunto de equações lineares que possuem a forma: $\boldsymbol{a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n = b}$. Com a aplicação de sistemas lineares é possível encontrar valores para as variáveis a partir da relação que as equações daquele sistema possuem. Os principais métodos de resolução de sistema linear utilizados nesse trabalho serão, o método de eliminação de Gauss e o método iterativo de Jacobi.

A solução de um sistema linear pode ser encontrada utilizando-se o método da matriz inversa, no sistema de equação na forma $\boldsymbol{Ax = b}$, em que $\boldsymbol{A}$ é uma matriz quadrada invertível, $\boldsymbol{x}$ é a matriz dos coeficientes e $\boldsymbol{b}$ é a matriz coluna das incógnitas, a solução desse sistema pode ser dada por: $\boldsymbol{x = A^{-1} \times b}$. Essa forma de solução não é computacionalmente eficiente, pois requer muito poder computacional.

O método de eliminação de Gauss consiste na manipulação das linhas da matriz, a partir de multiplicação por escalar, e soma entre as linhas, de forma que após a aplicação dessas operações a última linha da matriz tenha apenas dois termos diferentes de 0, em que um desses

termos é o último termo da matriz, encontrando o valor dessa variável e substituindo seu valor sequencialmente na linha acima, até encontrar o valor de todas as variáveis das equações que compõem a matriz.

Para o caso geral de um sistema linear de n incógnitas tem-se a seguinte forma:

$$\boldsymbol{a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1}$$

$$\boldsymbol{a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2}$$

$$\vdots$$

$$\boldsymbol{a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_n}$$

$$\boldsymbol{a_{11}x_1 + a_{12}x_2 = b_1}$$

$$\boldsymbol{a_{21}x_1 + a_{22}x_2 = b_2}$$

$$[\boldsymbol{A|b}] = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \boldsymbol{b_1} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} & \boldsymbol{b_2} \end{bmatrix}$$

Assumindo que $\boldsymbol{a_{11} \neq 0}$, pode-se fazer as seguintes operações:

$$L2 - \frac{\boldsymbol{a_{21}}}{\boldsymbol{a_{11}}} \times L1 \rightarrow L2,$$

em L2 representa a segunda linha da matriz, e L1 representa a primeira linha da matriz, assim a expressão:

$$L2 - \frac{\boldsymbol{a_{21}}}{\boldsymbol{a_{11}}} \times L1 \rightarrow L2 \ ,$$

significa que os elementos da segunda linha serão subtraídos pelo valor que será encontrado da divisão do elemento $\boldsymbol{a_{21}}$ pelo elemento $\boldsymbol{a_{11}}$ **,** multiplicados pelos elementos da primeira linha.

$$[\boldsymbol{A|b}] = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \boldsymbol{b_1} \\ 0 & \boldsymbol{a_{22}} & \boldsymbol{b_2} \end{bmatrix}$$

Tem-se então que $\boldsymbol{x_2 = \frac{b_2}{a_{22}}}$ e $\boldsymbol{x_1 = [b_1 - (a_{12} \times \frac{b_2}{a_{22}})] \div a_{11}.}$

O método interativo de Jacobi é um método aberto de resolução de sistema. Escolhe-se um valor para as variáveis a serem calculadas e resolve-se o sistema, em seguida escolhe-se

outro valor e resolve-se novamente o sistema, calcula-se o erro relativo e determina-se a o ponto de parada do método quando esse erro relativo for menor ao erro escolhido para o processo. O erro relativo de parada deve ser um valor em que os resultados das iterações indiquem uma convergência nos valores.

$$A_{m\times n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$A \times x = b.$$

Pode-se isolar as variáveis “x” e criar uma nova matriz, que terá sua diagonal composta por 0

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{(k+1)} = \begin{vmatrix} 0 & a_{12}/-a_{11} & \cdots & a_{1n}/-a_{11} \\ a_{21}/-a_{22} & 0 & \cdots & a_{2n}/-a_{22} \\ \vdots & \vdots & & 0 \quad \vdots \\ a_{m1}/-a_{mm} & a_{m2}/-a_{mm} & & \cdots \quad 0 \end{vmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{k} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

O erro relativo é encontrado com a seguinte fórmula em que “x” é a matriz dos valores encontrados, e “n” é o passo do método:

$$\in_r = \frac{|x_{n+1} - x_n|}{|x_{n+1}|}.$$

4. ÁLGEBRA LINEAR: ASPECTOS COMPUTACIONAIS

Na linguagem C/C++ pode-se declarar facilmente uma matriz de “n” linhas e “m” colunas apenas definindo sua variável seguida de dois colchetes que definirão respectivamente o número de linhas e de colunas. Um vetor pode ser entendido com uma matriz unidimensional, dessa forma, abre-se um colchete apenas, e escolhe-se o seu tamanho.

Para atribuir valores a cada elemento de uma matriz requirindo ao usuário que esses valores sejam digitados, utiliza-se uma estrutura de repetição. Faz-se uso de dois “for” definindo duas variáveis com valor inicial zero, e programando para que a cada repetição seja incrementado o valor de cada variável criada, e o número digitado seja armazenado naquela

posição, repetindo esse laço até que as variáveis tenham o mesmo valor que o número de linhas e de colunas respectivamente.

Em vários algoritmos produzidos nesse trabalho foi utilizada a forma dinâmica de criar matrizes com a utilização de ponteiros, criando matrizes dinâmicas que não precisam ter um número de linhas e colunas predeterminado, fazendo uso do operador “size of” determina-se o número de linhas e colunas para a matriz a partir de variáveis.

A criação de funções foi utilizada em vários algoritmos nesse trabalho, reduzindo e otimizando-os, de maneira que ações realizadas diversas vezes dentro de um mesmo algoritmo só precisam ser programadas uma única vez, a exemplo disso temos as funções de ler uma matriz, escrever a matriz, calcular o determinante e calcular os cofatores.

Neste trabalho foram utilizadas várias estruturas de programação, a estrutura de leitura permite a leitura de valores que serão alocados em variáveis ou em posições no caso de matrizes e vetores, na linguagem C a estrutura de leitura é o “scanf”, sua sintaxe é: scanf (“expressão de controle”, “lista de argumentos”). Na linguagem C++ a estrutura de leitura é o “cin”, e sua sintaxe é: cin >> variável.

As estruturas de escrita são similares as estruturas de leitura, elas são usadas para escrever na tela texto e/ou o valor das variáveis que serão chamadas por elas. Na linguagem C a estrutura de escrita é o “printf”, sua sintaxe é: printf(“texto a ser escrito” , “lista de argumentos”). Na linguagem C++ a estrutura de escrita é o “cout” e sua sintaxe é: cout << “texto a ser escrito” << “lista de variáveis”.

Os laços de repetição, ou estruturas de repetição, são estruturas que vão executar um determinado comando até que uma condição inicial seja satisfeita. Em vários algoritmos os laços de repetição foram utilizados para ler e escrever elementos de matrizes e vetores. As estruturas de repetição também são uteis quando é necessário que determinada condição seja satisfeita para o prosseguimento do algoritmo, fazendo-o entrar em loop e não avançar em sua execução até que essa condição seja satisfeita. Neste trabalho foram utilizadas duas estruturas de repetição, o “for” e o “while”. Na estrutura “for” toma-se uma variável como parâmetro e compara-se com outra utilizando os operadores de maior que (>), menor que (<) ou igual a (=) e até que essa condição seja satisfeita incrementa-se ou decrementa-se o valor da variável de parâmetro, sua sintaxe é: for (“variável de parâmetro”, “condição final”, “valor de incremento”).

Na estrutura de repetição “while” define-se uma condição, e enquanto essa condição for verdadeira, o algoritmo executará aquilo que estiver dentro do laço “while”, sua sintaxe é: while(condição) {algoritmo a ser seguido}.

As estruturas de decisão são aquelas que o algoritmo decide quais instruções seguir a partir de uma ou mais condições satisfeitas. Sua sintaxe é: if (condição) {instruções a serem executadas} else{instruções a serem executadas}.

Os ponteiros são um tipo de variável que armazena um endereço de memória dados (SCHILDT, 2003). A declaração de um ponteiro consiste em adicionar um asterisco na frente da variável (*). Neste trabalho eles foram utilizados para a criação dinâmica de matrizes e vetores junto com a função “malloc” e o operador “sizeof”. A função malloc permite a alocação dinâmica na memória, sua sintaxe é: malloc (número que será alocado). A função “sizeof” retorna o tamanho do dado a qual ela recebe como parâmetro. Sua sintaxe é: sizeof(variável). A criação de uma matriz dinâmica utilizando os ponteiros, a função “malloc” e a função “sizeof” terão a seguinte sintaxe: matriz= (tipo da variável*) malloc (sizeof (tipo davariável) * número de linhas * número de colunas).

4.1. ALGORITMO 1: Determinante

Para calcular o determinante foi utilizado a macro “define” para que um tamanho máximo para a matriz fosse estipulado e então a matriz fosse criada de forma dinâmica, a criação de funções para receber e escrever as matrizes utilizando laços de repetição, e a criação das funções do cálculo do determinante e dos cofatores. Como esse algoritmo utiliza o desenvolvimento de Laplace para encontrar o determinante, a função determinante precisa utilizar a função cofator, e a função cofator precisar utilizar a função determinante até que a matriz dos cofatores esteja em sua forma mais reduzida. Além disso foram utilizadas as condicionais “while” para só permitir valores de ordem positivos, e “if” para critério de parada do laço de repetição e para a chamada ou não de funções.

Inicio

Declaração de variáveis

Inteiros: i,j,m,a,b,n,h,w, l=0 , c=0

Racionais:matriz[MAX][MAX],matrizb[MAX][MAX],cofatores[MAX][MAX],det

***Leia** o a ordem da matriz (m)*

Leia *a matriz (M[i][j])*

Enquanto

m<0

Escreva *"números negativos são inválidos"*

Escreva *"Digite a ordem da matriz quadrada*

Leia *m*

Para *i=0, e i<m, incrementar 1 ao valor de "i"*

Para *j=0 e y<m, incrementar 1 ao valor de "j"*

Leia *(M[i][j])*

Escreva *a matriz (M[i][j])*

Para *i=0, e i<m, incrementar 1 ao valor de "i"*

Para *j=0 e y<m, incrementar 1 ao valor de "j"*

Escreva *(M[i][j])*

Função determinante *(Racional: matriz[MAX][MAX]; inteiro: m)*

Declaração de variáveis

Racional: det=0,0

Se *"m" for igual a 1, então, atribuir o valor da variável "matriz[0][0]" a variável "det"(det=matriz[0][0])*

Caso contrario

Para *j=0, e j<m, incrementar 1 ao valor de "j"*

Atribuir o a soma das variáveis "det" e "matriz[0][j]" multiplicada pelo valor do retorno da chamada ***da função cofator*** *passando os valores das variáveis "matriz" , "m" , "0" e "j" como parâmetros (det = det + matriz[0][j] * cofator(matriz,m, 0, j)*

Retornar *"det"*

Fim da função

Função cofator *(Racional: matriz[MAX][MAX]; Inteiro: m, linha, coluna)*

Declaração de variáveis

Racional: submatriz[MAX][MAX]

Inteiros: n=m-1 , a=0, b=0

Para *i=0, e i<m, incrementar 1 ao valor de "i"*

Para *j=0 e y<m, incrementar 1 ao valor de "j"*

Se *"i" for diferente de "linha" e "j" for diferente de "coluna"*

Então*: atribuir o valor da variável "matriz[i][j]" a variável "submatriz[a][b]" (submatriz[a][b]=matriz[i][j]), e incrementar 1 ao valor de "b"*

Se *"b" for maior ou igual a "n"*

Então*: incrementar 1 ao valor de "a" e atribuir 0 ao valor de "b" (b=0)*

Retornar $(\boldsymbol{linha + coluna})^{-1} \times \boldsymbol{determinante\ submatriz}$

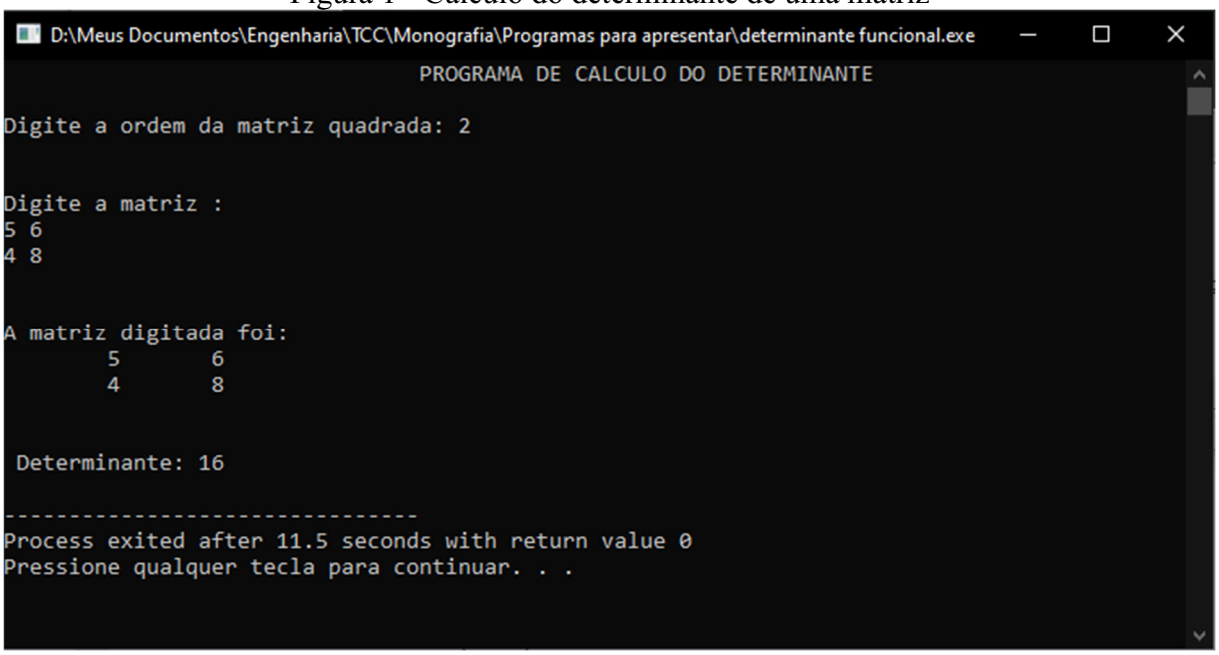
Fim da função

*Calcular determinante (Chamar função "****Função determinante****" passando os valores das variáveis "matriz" e "m" como parâmetros)*

Escreva *o determinante (det)*

Fim

Figura 1 - Cálculo do determinante de uma matriz



4.2. ALGORÍTMO 2: Somar matrizes

Nesse algoritmo foi utilizado os ponteiros para declarar as matrizes de forma dinâmica. Criou-se funções para ler, escrever e somar as matrizes declaradas utilizando laços de repetição.

Início

Declaração das variáveis

*Racional: *m1, *m2, *m3*

Inteiro: l,c,i,j

***Leia** o número de linhas(l)*

***Leia** o número de colunas(c)*

Criar matriz 1(m1) com "l" linhas e "c" colunas

Criar matriz 2(m2) com "l" linhas e "c" colunas

Criar matriz 1(m3) com "l" linhas e "c" colunas

***Função Ler elementos da matriz** (Racional: *matriz; inteiros: l, c)*

Declaração de variável

*Racional: *amatriz*

Inteiro: i, j

Atribuir o valor da variável "matriz" a variável "amatrzi" (amatriz=matriz)

***Para** i=0, e i<l, incrementar 1 ao valor de "i"*

***Para** j=0 e j<c, incrementar 1 ao valor de "j"*

***Ler** (amatriz)*

Fim da função

***Função escrever matriz** (racional: *matriz; inteiros: l,c)*

Declaração de variável

*Racional: *amatriz*

Inteiro: i, j

Atribuir o valor da variável" matriz" a variável "amatriz" (amatriz=matriz)

***Para** i=0, e i<l, incrementar 1 ao valor de "i"*

***Para** j=0 e j<c, incrementar 1 ao valor de "j"*

***Escreva** *(amatriz+i*c+j)*

Fim da função

Função somar matrizes *(Racional: *x, *y, *z; inteiros: l, c)*

Declaração de variável

*Racional: *pX=x, *pY=y, *pZ=z;*

Inteiro: i, j

Para *i=0, e i<l, incrementar 1 ao valor de "i"*

Para *j=0 e j<c, incrementar 1 ao valor de "j"*

*Atribuir o valor da soma das variáveis "*pX" incrementando 1 ao seu valor com a variável "*pY" incrementando 1 ao seu valor" a variável "*pZ" incrementando 1 ao seu valor (*pZ++=*pX++ + *pY++)*

Fim da função

*Ler a primeira matriz (chamar função "**Função Ler elementos da matriz**" passando as variáveis "m1", "l", e "c" como parâmetros)*

*Escrever a primeira matriz (chamar função "**Função escrever matriz**" passando as variáveis "m1", "l" e "c" como parâmetros)*

*Ler a segunda matriz (chamar função "**Função Ler elementos da matriz**" passando as variáveis "m2", "l"e "c" como parâmetros)*

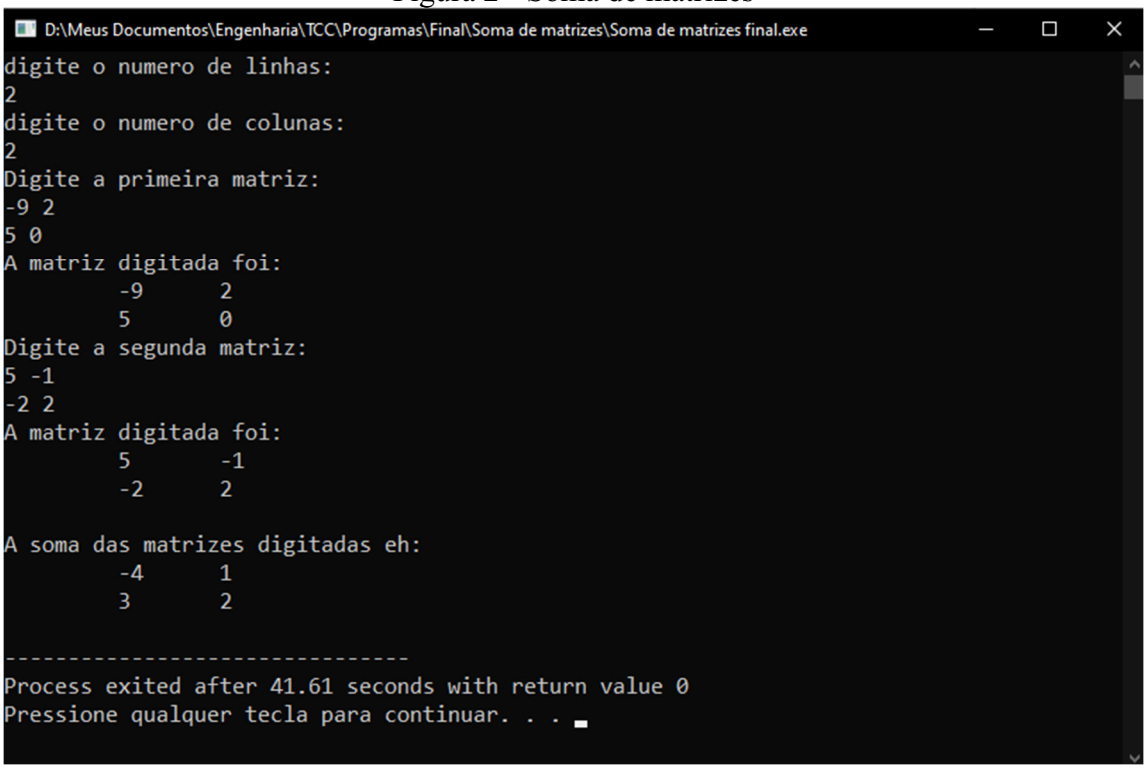
*Escrever a segunda matriz (chamar função "**Função escrever matriz**" passando as variáveis "m2", "l" e "c" como parâmetros)*

*Somar a primeira e a segunda matriz (chamar função "**somar matrizes**" passando as variáveis "m1", "m2", "m3", "l" e "c" como parâmetros)*

*Escrever o resultado da soma de matrizes (chamar função "**Função escrever matriz**" passando as variáveis "m3", "l", e "c" como parâmetros)*

Fim

Figura 2 - Soma de matrizes



4.3. ALGORÍTMO 3: Subtrair matrizes

Nesse algoritmo foi utilizado os ponteiros para declarar as matrizes de forma dinâmica. Criou-se funções para ler, escrever e subtrair as matrizes declaradas utilizando laços de repetição.

Ínicio

Declaração das variáveis

*Racional:*m1, *m2, *m3*

Inteiros: l,c,i,j

Leia*(l)*

Leia*(c)*

Criar matriz 1(m1) com "l" linhas e "c" colunas

Criar matriz 2(m2) com "l" linhas e "c" colunas

Criar matriz 1(m3) com "l" linhas e "c" colunas

Função Ler elementos da matriz *(Racional: *matriz; inteiros: l, c)*

Declaração das variáveis

*Racional: *amatriz*

Inteiro: i, j

Atribuir o valor da variável "matriz" a variável "amatriz" (amatriz=matriz)

***Para** i=0, e i<l, incrementar 1 ao valor de "i"*

***Para** j=0 e j<c, incrementar 1 ao valor de "j"*

***Ler** (amatriz)*

Fim da função

***Função escrever matriz** (racional: *matriz; inteiros: l,c)*

Declaração de variável

*Racional: *amatriz*

Inteiro: i, j

Atribuir o valor da variávei "matriz" a variável "amatriz" (amatriz=matriz)

***Para** i=0, e i<l, incrementar 1 ao valor de "i"*

***Para** j=0 e j<c, incrementar 1 ao valor de "j"*

***Escreva** *(amatriz+i*c+j)*

Fim da função

***Função subtrair matrizes** (Racional: *x, *y, *z; inteiros: l, c)*

Declaração de variável

*Racional: *pX=x, *pY=y, *pZ=z;*

Inteiro: i, j

***Para** i=0, e i<l, incrementar 1 ao valor de "i"*

***Para** j=0 e j<c, incrementar 1 ao valor de j*

*Atribuir o valor da subtração das variáveis "*pX" incrementando 1 ao seu valor com a variável "*pY" incrementando 1 ao seu valor a variável "*pZ" incrementando 1 ao seu valor (*pZ++=*pX++ - *pY++)*

Fim da função

*Ler a primeira matriz (chamar função "**Função Ler elementos da matriz**" passando as variáveis "m1", "l", e "c" como parâmetros)*

*Escrever a primeira matriz (chamar função "**Função escrever matriz**" passando as variáveis "m1", "l" e "c" como parâmetros)*

*Ler a segunda matriz (chamar função "**Função Ler elementos da matriz**" passando as variáveis "m2", "l"e "c" como parâmetros)*

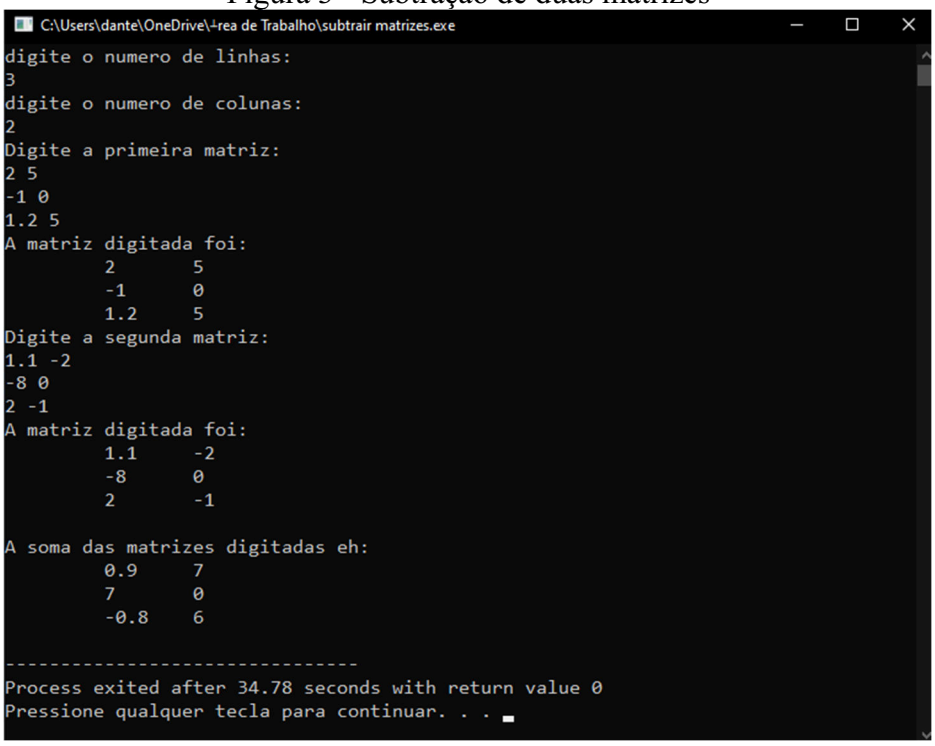
*Escrever a segunda matriz (chamar função "**Função escrever matriz**" passando as variáveis "m2", "l" e "c" como parâmetros)*

*Subtrair a primeira e a segunda matriz (chamar função "**subtrair matrizes**" passando as variáveis "m1", "m2", "m3", "l" e "c" como parâmetros)*

*Escrever o resultado da subtração de matrizes (chamar função "**Função escrever matriz**" passando as variáveis "m3", "l"e "c" como parâmetros)*

Fim

Figura 3 - Subtração de duas matrizes



4.4. ALGORÍTMO 4: Multiplicar matrizes

No referido algoritmo as matrizes foram definidas a partir de variáveis para seu número de linhas e colunas a partir da função "scanf". Os laços de repetição foram utilizados para ler e escrever, e multiplicar as matrizes. A condicional "if" foi utilizada para determinar se o algoritmo continuaria ou não de acordo com a igualdade entre o valor do número de colunas da primeira matriz, com o número de linhas da segunda matriz.

Início

Declaração das variáveis

Inteiros: i, j, lA, cA, lB, cB, X

Ler quantidade de linhas da matriz A

Leia *lA*

Ler quantidade de colunas da matriz A

Leia *CA*

Ler quantidade de linhas da matriz B

Leia *lB*

Ler quantidade de colunas da matriz B

Leia *cB*

Declaração de variáveis

Flutuantes: matrizA[lA][cA], matrizB[lB][cB], matrizC[lA][cB], Aux=0;

Se *(o número de colunas da matriz A for igual ao número de linhas da matriz B)*

Então*:*

Ler*(matrizA)*

Para *i=0, e i<lA, incrementar 1 ao valor de "i"*

Para *j=0 e j<cA, incrementar 1 ao valor de "j"*

Leia *Valor da linha "l" e da coluna "c" da matrizA*

Ler*(matrizB)*

Para *i=0, e i<lB, incrementar 1 ao valor de "i"*

Para *j=0 e j<cB, incrementar 1 ao valor de "j"*

Leia *Valor da linha "l" e da coluna "c" da matrizB*

Escrever(matrizA)

Para *i=0, e i<lA, incrementar 1 ao valor de "i"*

Para *j=0 e j<cA, incrementar 1 ao valor de "j"*

Escreva *matrizA[i][j]*

Escrever(matrizB)

Para *i=0, e i<lB, incrementar 1 ao valor de "i"*

Para *j=0 e j<cB, incrementar 1 ao valor de "j"*

Escreva *matrizB[i][j]*

Multiplicando as matrizes

Para *i=0, e i<lA, incrementar 1 ao valor de "i"*

Para *j=0 e j<cB, incrementar 1 ao valor de "j"*

Atribuir o valor 0 a variável "matrizC[i][j]" (matrizC[i][j] = 0)

Para *X=0, e X<lB, incrementar 1 ao valor de "X"*

*Atribuir o valor da multiplicação das variáveis" matrizA[i][X]" e "matrizB[X][j]" a variável "Aux" somando ao seu valor atual (Aux += matrizA[i][X] * matrizB[X][j])*

Atribuir o valor da variável "aux" a variável "matrizC[i][j]" (matrizC[i][j] =Aux)

Atribuir o valor de 0 a variável "Aux" (Aux=0)

Escrever *resultado da multiplicação*

Para *i=0, e i<lA, incrementar 1 ao valor de "i"*

Para *j=0 e j<cB, incrementar 1 ao valor de "j"*

Escreva *matrizC[i][j]*

Caso contrário

Escreva *"Não é possível fazer o produto das duas matrizes, pois, o número de linhas da matriz A é diferente do número de linhas da matriz B"*

Fim

Figura 4 - Multiplicação entre duas matrizes parte 1

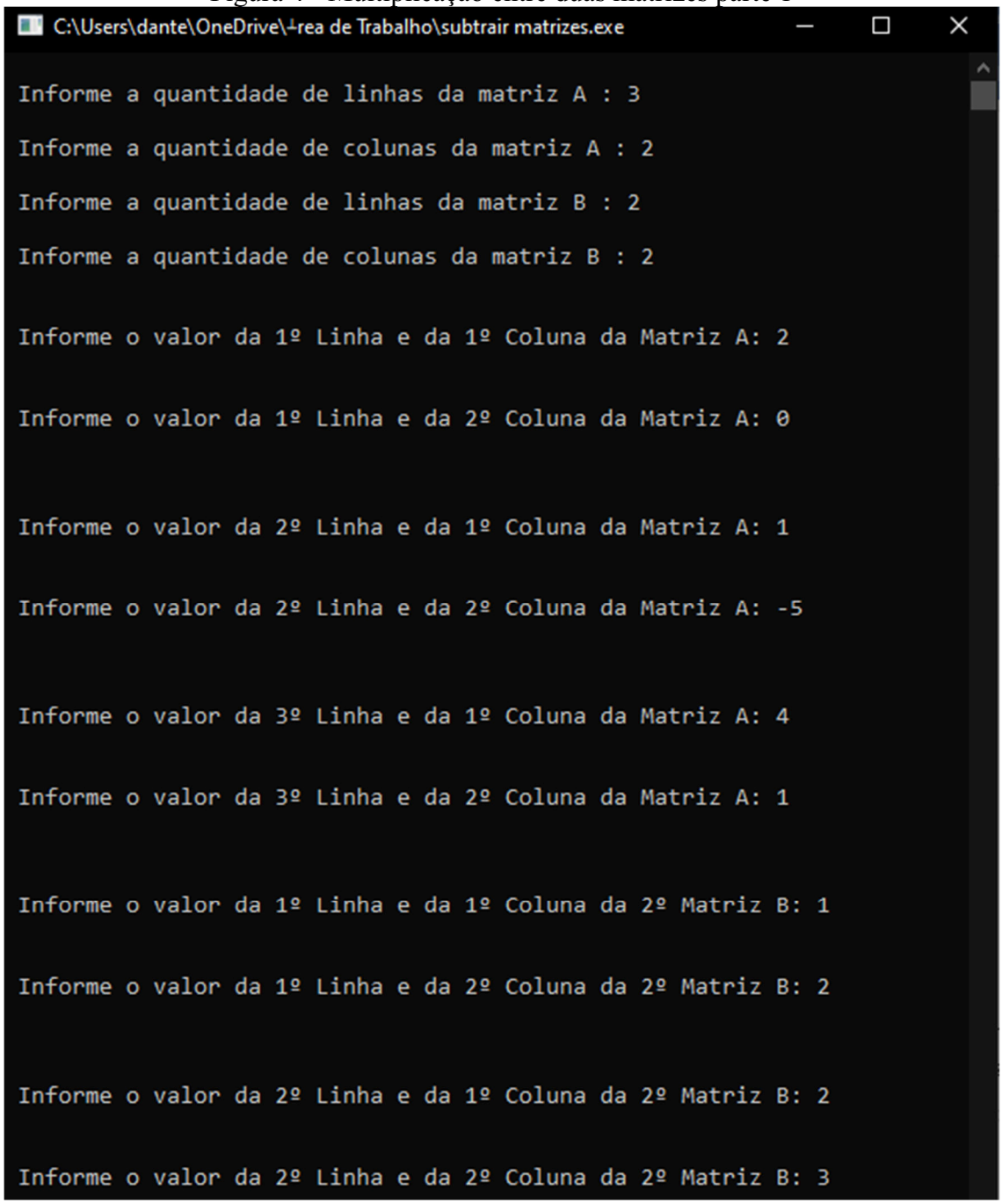
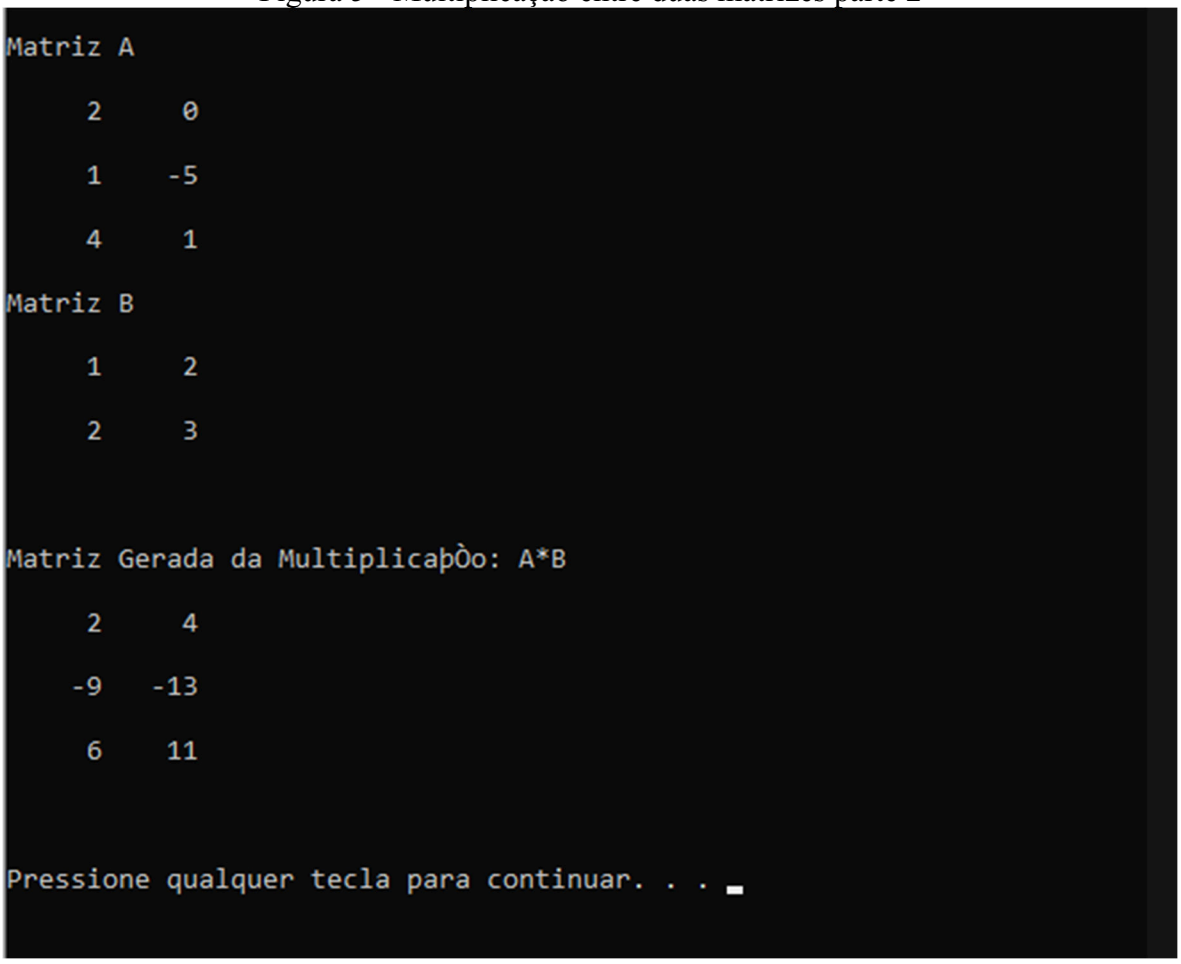


Figura 5 - Multiplicação entre duas matrizes parte 2



4.5. ALGORÍTMO 5: Transpor matriz

As matrizes foram definidas dinamicamente através dos ponteiros e criadas funções para ler, escrever e transpor a matriz utilizando laços de repetição.

Início

Declaração das variáveis

Inteiros: a,b,i

*Flutuantes: **matrizA, **matrizB*

Ler a quantidade de linhas da matrizA

***Leia** a*

Ler quantidade de colunas da matrizA

***Leia** b*

Criar matriz A com número de linhas "a", e número de colunas "b"

Criar matriz B com número de linhas "b", e número de colunas "a"

Função Ler elementos da matriz *(Flutuante: **matriz; inteiros: m, n)*

Declaração de variável

Inteiro: i, j

Para *i=0, e i<m, incrementar 1 ao valor de "i"*

Para *j=0 e j<n, incrementar 1 ao valor de "j"*

Ler *(matriz[i][j])*

Fim da função

Função escrever matriz *(racional: *matriz; inteiros: m,n)*

Declaração de variável

Inteiro: i, j

Para *i=0, e i<n, incrementar 1 ao valor de "i"*

Para *j=0 e j<m, incrementar 1 ao valor de "j"*

Escreva *(matriz[i][j])*

Fim da função

Função transpor matriz *(Inteiro: a,b ; Flutuante: **matrizA, **matrizB)*

Declaração de variáveis

Inteiro: i, j

Para *i=0, e i<a, incrementar 1 ao valor de "i"*

Para *j=0 e j<b, incrementar 1 ao valor de "j"*

Atribuir o valor da variável "matrizA[i][j]" a variável "matrizB[j][i]" (matrizB[j][i] = matrizA[i][j])

Fim da função

*Ler a matriz (chamar função "**Função Ler elementos da matriz**" passando as variáveis "a", "b" e "matriza" como parâmetros)*

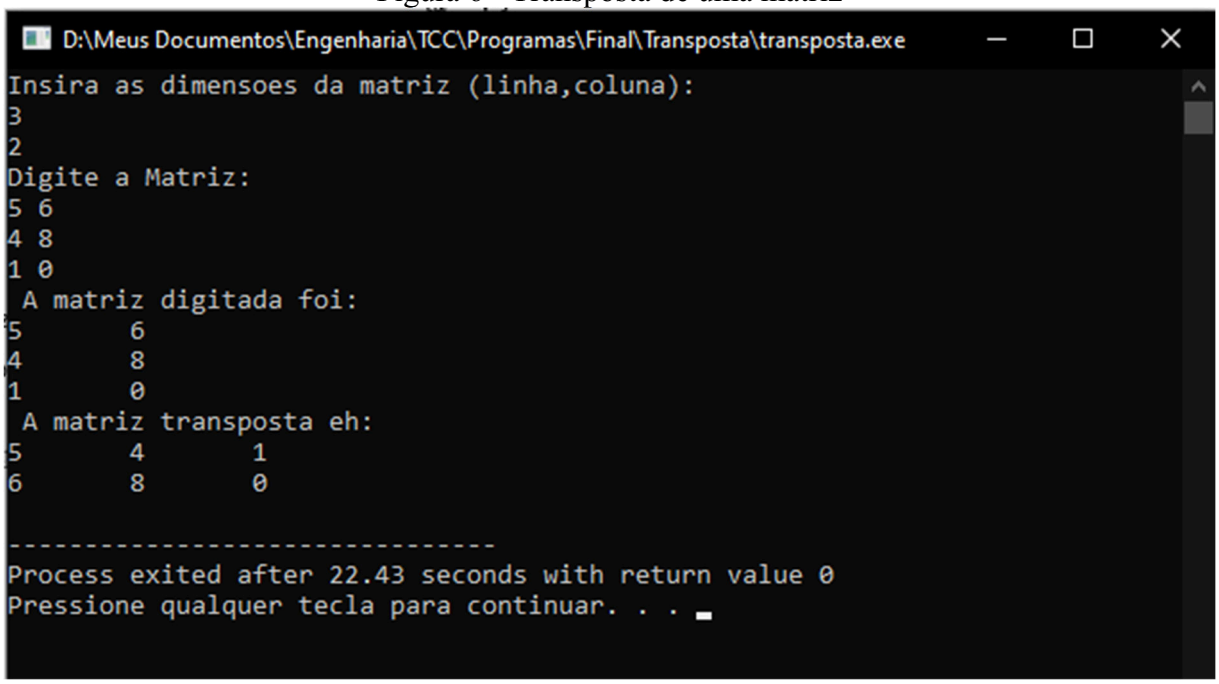
*Escrever a matriz (chamar função "**Função escrever matriz**" passando as variáveis "a", "b" e "matriza" como parâmetros)*

*Transpor a matriz (chamar a função "**Função transpor matriz**" passando as variáveis "a", "b", "matriza" e "matrizB" como parâmetros)*

*Escrever a matriz transposta (chamar função "**Função escrever matriz**" passando as variáveis "b", "a" e "matrizB" como parâmetros)*

Fim

Figura 6 - Transposta de uma matriz



4.6. ALGORÍTMO 6: Produto escalar

O cálculo do produto escalar acontece entre dois vetores, assim, foram definidos os vetores, com um laço de repetição simples (apenas um "for"), em seguida utilizado mais um laço de repetição para multiplicar os elementos do primeiro vetor com o segundo de acordo com sua posição, e em seguida mais um laço de repetição para somar o resultado das multiplicações.

Início

Declaração das variáveis

Racional: e=0

Inteiro: tam, i

Leia *dimensão dos vetores (tam)*

Declaração de variáveis

Racional:

a[tam], b[tam], vet[tam]

Leia *o primeiro vetor (a[tam])*

Para *i=0, e i<tam, incrementar 1 ao valor de "i"*

Leia *a[i]*

Escreva *o primeiro vetor*

Para *i=0, e i<tam, incrementar 1 ao valor de "i"*

Escreva *a[i]*

Leia *o segundo vetor (b[tam])*

Para *i=0, e i<tam, incrementar 1 ao valor de i""*

Leia *b[i]*

Escreva *o primeiro vetor*

Para *i=0, e i<tam, incrementar 1 ao valor de "i"*

Escreva *b[i]*

Para *i=0, e i<tam, incrementar 1 ao valor de "i"*

Atribuir o valor da multiplicação das variáveis "a[i]" e "b[i]" a variável "vet[i]" ($\boldsymbol{vet[i] = a[i] \times b[i]}$*)*

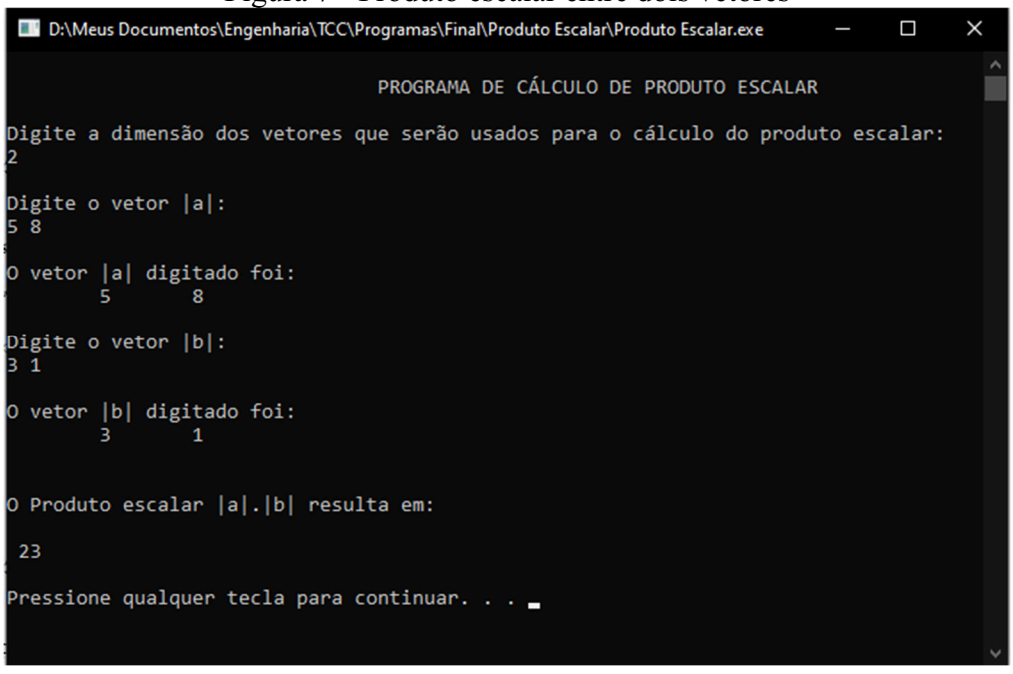
Para *i=0, e i<tam, incrementar 1 ao valor de "i"*

Atribuir o valor da soma das variáveis "e" e "vet[i]" ao valor da variável "e" ($\boldsymbol{e = e + vet[i]}$*)*

Escreva *o produto escalar (e)*

Fim

Figura 7 - Produto escalar entre dois vetores



4.7. ALGORÍTMO 7: Produto vetorial

O produto vetorial realizado neste trabalho é no espaço tridimensional, logo seu tamanho é predeterminado, então foi declarado dois vetores de dimensão 3, em seguida utilizado laço de repetição para determinar os valores, e foi escrito uma operação para cada posição do vetor resultante.

Início

Declaração das variáveis

Racional: vet [3], a [3], b [3]

Inteiro: tam, i

Leia *o primeiro vetor (a[tam])*

Para *i=0, e i<3, incrementar 1 ao valor de "i"*

Leia *a[i]*

Leia *o segundo vetor (b[tam])*

Para *i=0, e i<3, incrementar 1 ao valor de "i"*

Leia *b[i]*

Leia

Atribuir o valor da multiplicação das variáveis "a[1]" e "b[2]" menos a multiplicação das variáveis"b[1]" e "a[2]" ao valor da variável "vet[0]" ($\boldsymbol{vet[0] = (a[1] \times b[2] - b[1] \times a[2])}$*)*

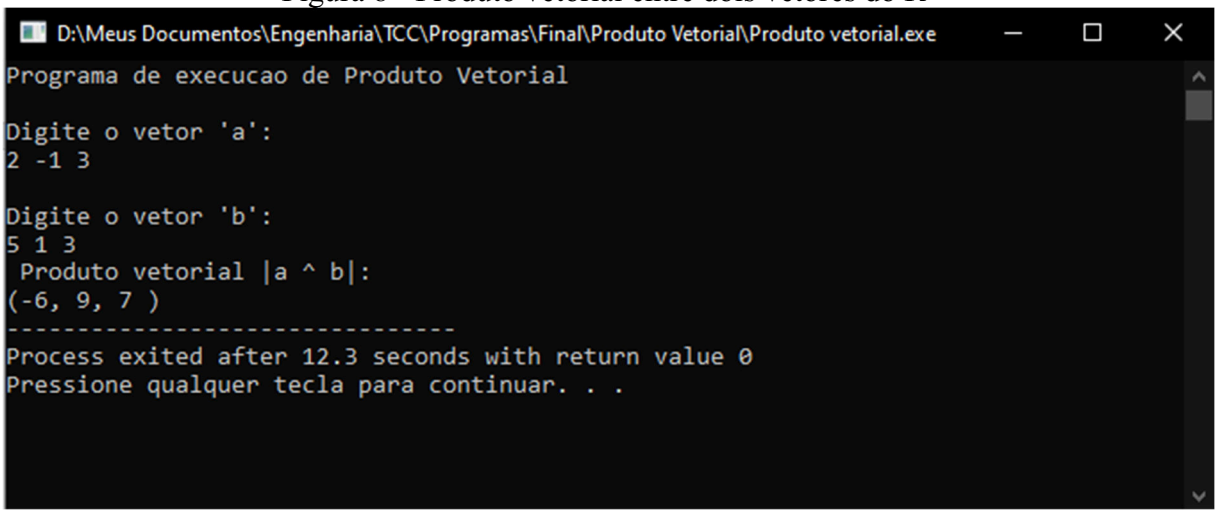
Atribuir o valor da multiplicação das variáveis "a[2]" e "b[0]" menos a multiplicação das variáveis"b[2]" e "a[0]" ao valor da variável "vet[1]" ($\boldsymbol{vet[1] = (a[2] \times b[0] - b[2] \times a[0])}$*)*

Atribuir o valor da multiplicação das variáveis "a[10]" e "b[1]" menos a multiplicação das variáveis"b[0]" e "a[1]" ao valor da variável "vet[2]" ($\boldsymbol{vet[2] = (a[0] \times b[1] - b[0] \times a[1])}$*)*

Escreva *o produto vetorial (vet[0] , vet[1] , vet[2])*

Fim

Figura 8 - Produto vetorial entre dois vetores do R^3



4.8. ALGORÍTMO 8: Resolução de sistema linear por eliminação de Gauss

Para a construção desse algoritmo foi criada uma matriz que armazena as equações, e um vetor para armazenar os resultados, a matriz foi preenchida usando laços de repetição, e também utilizando laços de repetição foi realizada a eliminação progressiva das linhas, e o cálculo das variáveis a serem encontradas.

Início

Declaração de variáveis

Inteiros: p, j, k,a,b,c,x,y,V

Flutuantes: fator, soma

Leia *o número de equações (V)*

Enquanto *V<0, escreva "o número de equações não pode ser negativo", leia (V)*

Declaração de variável

Inteiro: V2=V+1

Flutuante: L[V], M[V][V2]

Leia *a matriz do sistema de equação (M[V][V2])*

Para *x=0, e x<V, incrementar 1 ao valor de "x"*

Para *y=0 e y<V2, incrementar 1 ao valor de "y"*

Leia *(M[x][y])*

Escreva *a matriz do sistema de equação*

Para *p=0, e p<V, incrementar 1 ao valor de "p"*

Para *j=0 e j<V, incrementar 1 ao valor de "j"*

Escreva *(M[p][j])*

Eliminação Progressiva

Para *k=0, e k<V-1, incrementar 1 ao valor de "k"*

Para *p=k+1 e p<V, incrementar 1 ao valor de "p"*

Atribuir o valor da divisão das variáveis "M[p][k]" e "M[k][k]" a variável "fator "(fator= M[p][k] /M[k][k])

Para *j=0 e j<V ou j=V, incrementar 1 ao valor de "j"*

*Atribuir o valor da subtração da variável "M[p][j]" da multiplicação das variáveis "fator" e "M[k][j]" a variável "M[p][j]" (M[p][j] =M[p][j] – fator*M[k][j])*

Atribuir o valor da divisão das variáveis "M[V-1][V]" e "M[V-1][V-1]" a variável "L[V-1]" (L[V-1] =M[V-1][V]/ M[V-1][V-1])

Escreva *a matriz escalonada*

Para *p=0, e p<V, incrementar 1 ao valor de "p"*

Para *j=0 e j<V, incrementar 1 ao valor de "j"*

Escreva *(M[p][j])*

Substituição Progressiva

Para *p=V-2, e p>0 ou p=0, decrementar 1 ao valor de "p"*

Atribuir o valor 0 a variável "soma" (soma=0)

Para *j=p+1, e j<V, incrementar 1 ao valor de "j"*

*Somar o valor da vaiável "soma" ao valor variável M[p][j] multiplicado pela variável L[j] e atribuir a variável "soma" (soma = soma + M[p][j] *L[j])*

Subtrair a variável M[p][V] da variável soma e dividir pela variável M[p][p] (L[p] = (M[p][V] - soma) /M[p][p]

Escreva *o resultado da solução do sistema (L[p])*

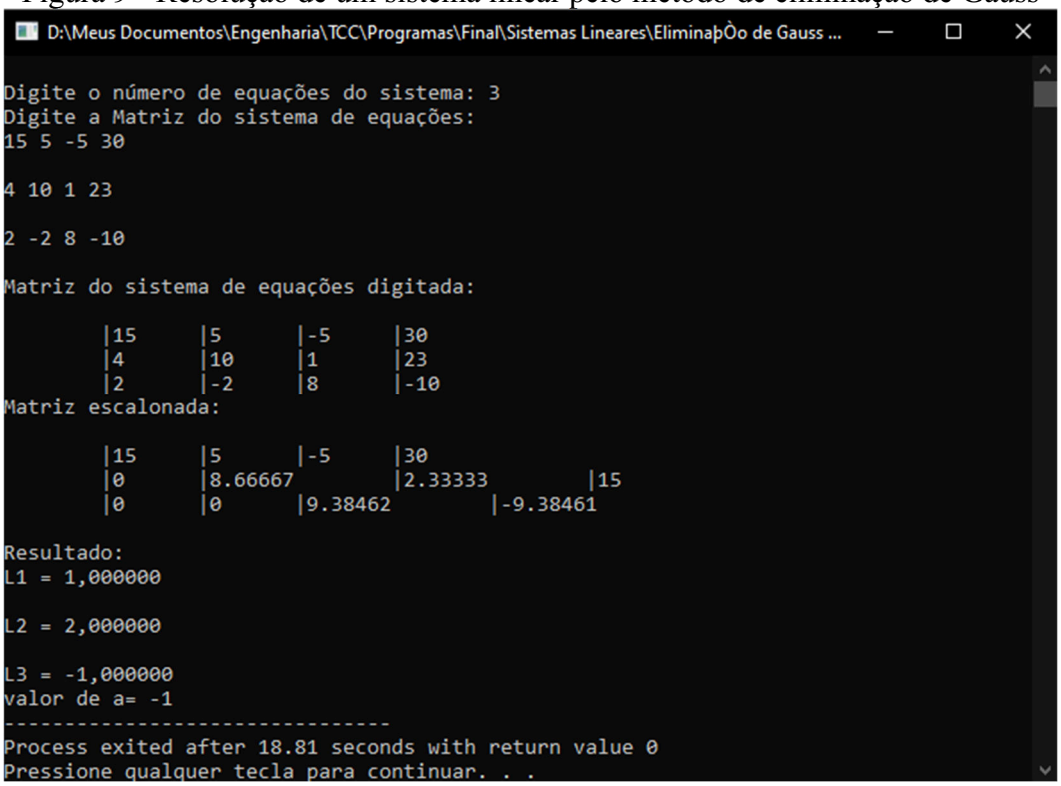
Para *p=0, e p<V, incrementar 1 ao valor de "p"*

Escrever *p+1, L[p]*

Escreva *o valor de "a" (L[2])*

Fim

Figura 9 - Resolução de um sistema linear pelo método de eliminação de Gauss



4.9. ALGORÍTMO 9: Resolução de sistema linear pelo método de Jacobi

Neste algoritmo foram declarados matrizes e vetores de maneira dinâmica através de ponteiros e utilizado o teste lógico booleano para aferir a convergência do método de Jacobi, foram criadas funções para a aplicação do teste de convergência, e verificação da existência da diagonal dominante.

Inicio

Declaração de variáveis

Inteiros: dimA

*Racional: matriz_a, matriz_b, *x0*

Leia *a dimensão da matriz (dimA)*

Para *i=0, e i<dimA, incrementar 1 ao valor de "i"*

Definir dimA como a ordem da matriz (matriz_a

Preencher valores na matriz_a

Para *i=0, e i<dimA, incrementar 1 ao valore de "i"*

Para *j=0, e j<dimA, incrementar 1 ao valor de "j"*

Leia*(matriz_a[i][j])*

Preencher valores da matriz_b

Para *i=0, e i<dimA, incrementar 1 ao valor de "i"*

Leia *(matriz_b[i])*

Função verificar linhas *(Racional: **matriz ; Inteiro: dimensao)*

Declaração de variáveis:

Inteiros: i,j

Para *i=0, e i<dimensão, incrementar 1 ao valor de "i"*

Declarar variável:

Inteiro: soma=0

Para *j=0, e j <dimensão, incrementar 1 ao valor de "j"*

Se *"i" for diferente de "j"*

Somar o valor da variável "matriz[i][j]" a variável "soma" (soma+=matriz[i][j])

Se "*soma" for maior que o valor da variável "matriz[i][i]"* ***Retornar "falso"***

Retornar verdadeiro

Fim da função

Função verificar colunas *(Racional:**matriz; Inteiro:dimensão)*

Declaração de variáveis:

Inteiros: i,j

Para *i=0, e i<dimensão, incrementar 1 ao valor de "i"*

Declarar variável:

Inteiro: soma=0

Para *j=0, e j <dimensão, incrementar 1 ao valor de "j"*

Se *"i" for diferente de "j"*

Somar o valor da variável "matriz[i][j]" a variável "soma" (soma+=matriz[i][j])

Se "*soma" for maior que o valor da variável "matriz[i][i]"* ***Retornar "falso"***

Retornar verdadeiro

Fim da função

Função testar convergência *(Racional: *x_anterior, *x_proximo; Inteiro: dimensão)*

Declaração de variáveis:

Racional: maior_num=0 , maior_den=0

Para *valor absoluto de (x_proximo[i] – x_anterior[i]) maior que "maior_num"*

Atribuir o valor absoluto da subtração de x_proximo[i] por x+anterior[i] ao valor de "maior_num" (maior_num=x_proximo[i]- x+anterior[i])

Se *valor absoluto de "x_proximo[i]" for maior que "maior_den"*

Atribuir o valor absoluto de "x_proximo[i]" a varável "maior_den"(maior_den = x_proximo[i])

Retornar *maior_num/maior_den*

Fim da função

Verificar se o método vai convergir a partir da verificação da existência da diagonal dominante

*Aplicar teste lógico Booleano chamando a função "****função verificar linhas****" passando as variáveis "matriz_a" e "dimA" como parâmetros*

Se *teste for igual a "falso"*

*Atribuir o retorno da função "****função verificar linhas****" passando as variáveis "matriz_a" e "dimA" como parâmetros a variável "teste" (teste =* ***função verificar linhas*** *(matriz_a, dimA))*

Declarar variável:

Inteiro: num_ite = 0

Se *a variável teste for igual a falso*

Escreva: *O método não converge pela verificação da diagonal dominante.*

Escreva: *Digite um número de iterações > 0 para aplicar o método mesmo assim:*

Leia *(num_ite)*

Se *num_ite for menor ou igual a 0 (num_ite ≤0)*

***Retornar** -1*

Declaração de variável:

Racional: precisao

***Escreva:** Digite a precisão do método:*

***Leia**(precisa)*

***Para** i=0, e i<dimA, incremente 1 ao valor de "i"*

Atribuir o valor da divisão entre "matriz_b[i]/matriz_a[i][i] a variável x0[i] (x0[i]=matriz_b[i]/matriz_a[i][i])

Criando vetor de iterações

Declaração de variáveis:

*Racional: *xk , *xt (passando para os doisvetores como dimensão a variável "dimA")*

***Se** num_ite > 0*

***Para** num_ite > 0 , decrementar 1 ao valor de num_ite*

***Para** i=0, i<dimA, incrementar 1 ao valor de "i"*

Declarar variável:

Racional: soma = matriz_b[i]

***Para** j=0, j<dimA, incrementar 1 ao valor de "j"*

***Se** "j" for diferente de "i"*

*Multiplicar as variávei "x0[j]" e "matriz_a[i][j]" e subtrair esse valor da variável "soma" (soma -= x0[j] * matriz_a[i][j])*

*Atribuir a divisão de 1 pela variável "matriz_a[i][i]" multiplicado pela variável "soma" ao valor da variável "xk[i]" (xk[i] = (1.0 / matriz_a[i][i]) * (soma))*

***Para** i=0, i< dimA, incrementar 1 ao valor de "i"*

Atribuir o valor da variável "x0[i]" ao valor da variável "xt[i]"(xt[i] = x0[i])

Atribuir o valor da variável "xk[i]" ao valor da variável "x0[i]"(x0[i] = xk[i])

***Caso contrário**:*

Fazer:

***Para** i=0, i<dimA, incrementar o 1 ao valor de "i"*

Declarar variável:

Racional: soma = matriz_b[i]

Para *j=0, j<dimA , incrementar 1 ao valor de "j"*

Se *"j" for diferente de "i"*

*Multiplicar a variável "x0[j]" a variável "*matriz_a[i][j]" e subtrair do valor da variável soma (soma -= x0[j] * matriz_a[i][j])*

*Atribuir a divisão de 1 pela variável "matriz_a[i][i]" multiplicado pela variável "soma" ao valor da variável "xk[i]" (xk[i] = (1.0 / matriz_a[i][i]) * (soma))*

Para *i=0, i<dimA, incrementar 1 ao valor de "i"*

Atribuir o valor da variável "x0[i]" ao valor da variável "xt[i]"(xt[i] = x0[i])

Atribuir o valor da variável "xk[i]" ao valor da variável "x0[i]"(x0[i] = xk[i])

Enquanto *o valor retornado pela função, "**função testar convergência**" passando os valores de "xt", "xk" e "dimA" como parâmetros, for maior que o valor da variável "precisao"*

Escreva*: Resultado:*

Para *i=0, i<dimA, incrementar 1 ao valor de "i"*

Escreva *(x = xk[i])*

Para *i=0, i<dimA, incrementar 1 ao valor de "i"*

Liberar*(matriz_a[i])*

Liberar*(matriz_a)*

Liberar*(matriz_b)*

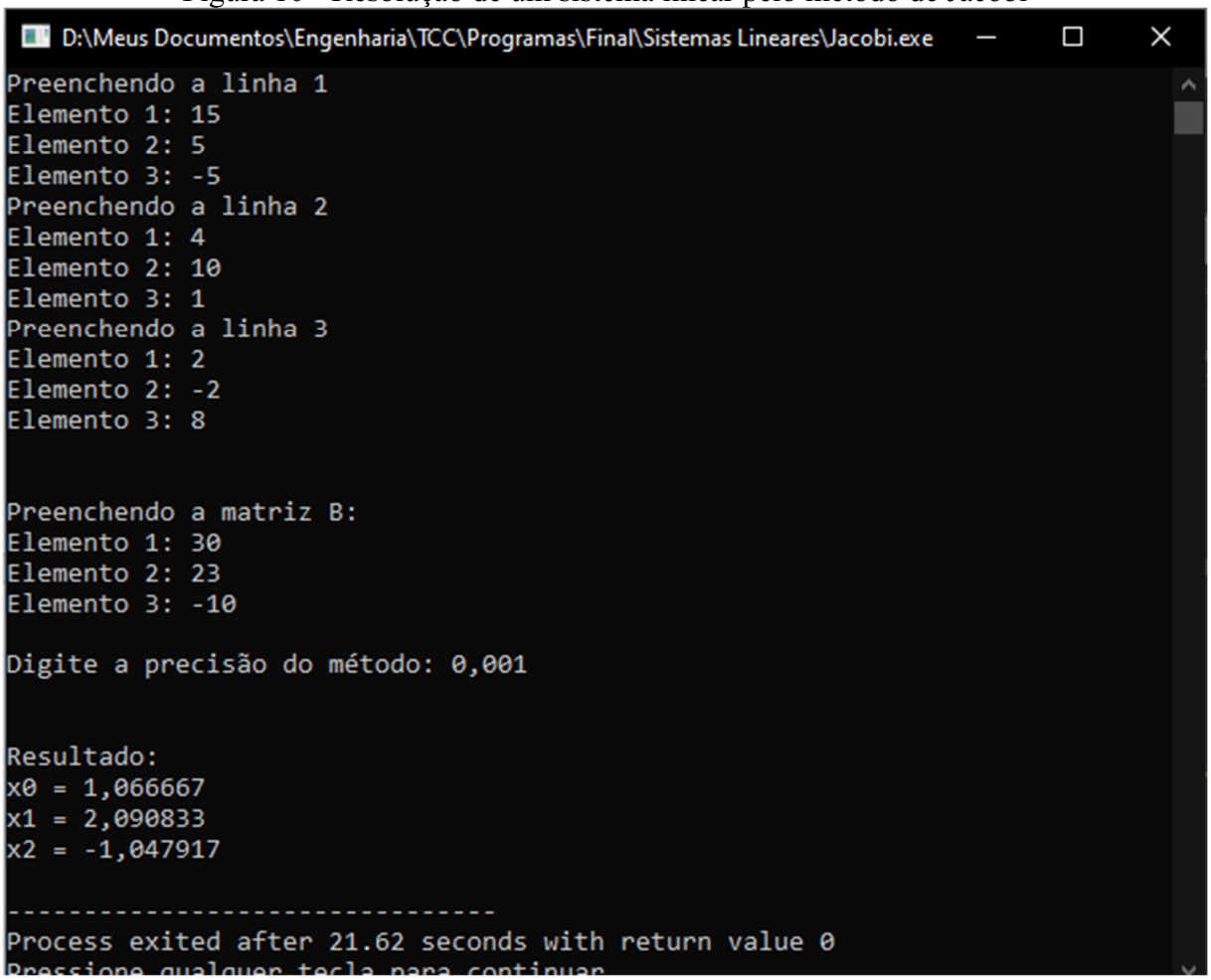
Liberar*(x0)*

Liberar*(xk)*

Liberar*(xt)*

Fim

Figura 10 - Resolução de um sistema linear pelo método de Jacobi



4.10. ALGORÍTMO 10: Matriz inversa

As matrizes declaradas nesse algoritmo fizeram uso da macro “define” para que fosse estipulado um tamanho máximo da matriz, dessa forma não precisando fazer uso de ponteiros para declara-la, utilizou-se laços de repetição para adicionar valores, e mostrar a matriz, e a condicional “if” para quando a matriz não pudesse ser invertida, quando sua ordem fosse 1, e quando sua ordem fosse maior que 1, foi utilizado também laços de repetição para o cálculo da transposta dos cofatores e a inversa, também foram criadas funções para calcular os cofatores e o determinante, já que o algoritmo calcula a inversa a partir da fórmula da inversão de matrizes.

Inicio

Declaração de variáveis

Inteiros: i,j,m,a,b,n,h,w, l=0, c=0

Racionais:matriz[MAX][MAX],matrizb[MAX][MAX],cofatores[MAX][MAX], inversa[MAX][MAX],transposta[MAX][MAX],det

***Leia** o a ordem da matriz (m)*

***Leia** a matriz (M[i][j])*

***Para** i=0, e i<m, incrementar 1 ao valor de "i"*

***Para** j=0 e y<m, incrementar 1 ao valor de "j"*

***Leia** (M[i][j])*

***Escreva** a matriz (M[i][j])*

***Para** i=0, e i<m, incrementar 1 ao valor de "i"*

***Para** j=0 e y<m, incrementar 1 ao valor de "j"*

***Escreva** (M[i][j])*

***Função determinante** (Racional: matriz[MAX][MAX]; inteiro: m)*

Declaração de variáveis

Racional: det=0,0

***Se** "m" for igual a 1, então, atribuir o valor da variável "matriz[0][0]" a variável "det"(det=matriz[0][0])*

Caso contrario

***Para** j=0, e j<m, incrementar 1 ao valor de "j"*

*Atribuir o a soma das variáveis "det" e "matriz[0][j]" multiplicada pelo valor do retorno da chamada **da função cofator** passando os valores das variáveis "matriz" , "m" , "0" e "j" como parâmetros (det = det + matriz[0][j] * cofator(matriz,m, 0, j)*

***Retornar** "det"*

Fim da função

***Função cofator** (Racional: matriz[MAX][MAX]; Inteiro: m, linha, coluna)*

Declaração de variáveis

Racional: submatriz[MAX][MAX]

Inteiros: n=m-1 , a=0, b=0

***Para** i=0, e i<m, incrementar 1 ao valor de "i"*

***Para** j=0 e y<m, incrementar 1 ao valor de "j"*

***Se** "i" for diferente de "linha" e "j" for diferente de "coluna"*

***Então**: atribuir o valor da variável "matriz[i][j]" a variável "submatriz[a][b]" (submatriz[a][b]=matriz[i][j]), e incrementar 1 ao valor de "b"*

***Se** "b" for maior ou igual a "n"*

***Então**: incrementar 1 ao valor de "a" e atribuir 0 ao valor de "b" (b=0)*

Retornar $(\boldsymbol{linha + coluna})^{-1} \times \boldsymbol{determinante\ submatriz}$

Fim da função

*Calcular determinante (Chamar função "**Função determinante**" passando os valores das variáveis "matriz" e "m" como parâmetros)*

***Escreva** o determinante (det)*

***Se** o determinante for igual a 0, **Escreva**: A MATRIZ NAO PODE SER INVERTIDA!!*

***Se** a ordem da matriz (m) for igual a 1, atribuir o valor da variável "matriz[0][0]" elevado a -1, a variável "matriz[0][0]" (*$\boldsymbol{matriz[0][0] = matriz[0][0]^{-1}}$*)*

***Escreva** a matriz inversa (matriz[0][0])*

***Se** o determinante for diferente de 0, e a ordem da matriz for maior ou igual a 2*

Então

Calcular cofatores

***Para** i=0, e i<m, incrementar 1 ao valor de "i"*

***Para** j=0 e y<m, incrementar 1 ao valor de "j"*

*(Chamar a função "**Função cofator**" passando as variáveis "matriz", "m", "i", "j" como parâmetros) e atribuir o retorno dessa função a variável "cofatores[i][j] (cofatores[i][j]=cofator(matriz,m,i,j))*

***Escreva** a matriz dos cofatores*

***Para** i=0, e i<m, incrementar 1 ao valor de "i"*

***Para** j=0 e y<m, incrementar 1 ao valor de "j"*

***Escreva cofatores**[i][j]*

Transpor a matriz

***Para** h=0, e h<m, incrementar 1 ao valor de "h"*

***Para** w=0 e w<m, incrementar 1 ao valor de "w"*

Atribuir o valor da variável "cofatores[h][w]" a variável "transposta[w][h]" (transposta[w][h]=cofatores[h][w])

Escreva *a matriz transposta dos cofatores*

Para *w=0, e w<m, incrementar 1 ao valor de "w"*

Para *h=0 e h<m, incrementar 1 ao valor de "h"*

Escreva *transposta[w][h]*

Calcular matriz inversa

Para *w=0, e w<m, incrementar 1 ao valor de "w"*

Para *h=0 e h<m, incrementar 1 ao valor de "h"*

*Atribuir o valor da divisão variável transposta[w][h] pelo valor da chamada da função "**Função determinante**" passando os valores das variáveis "matriz" e "m" como parâmetros, a variável "inversa[w][h]" (inversa[w][h]=transposta[w][h]/determinante (matriz,m))*

Escreva *a matriz inversa*

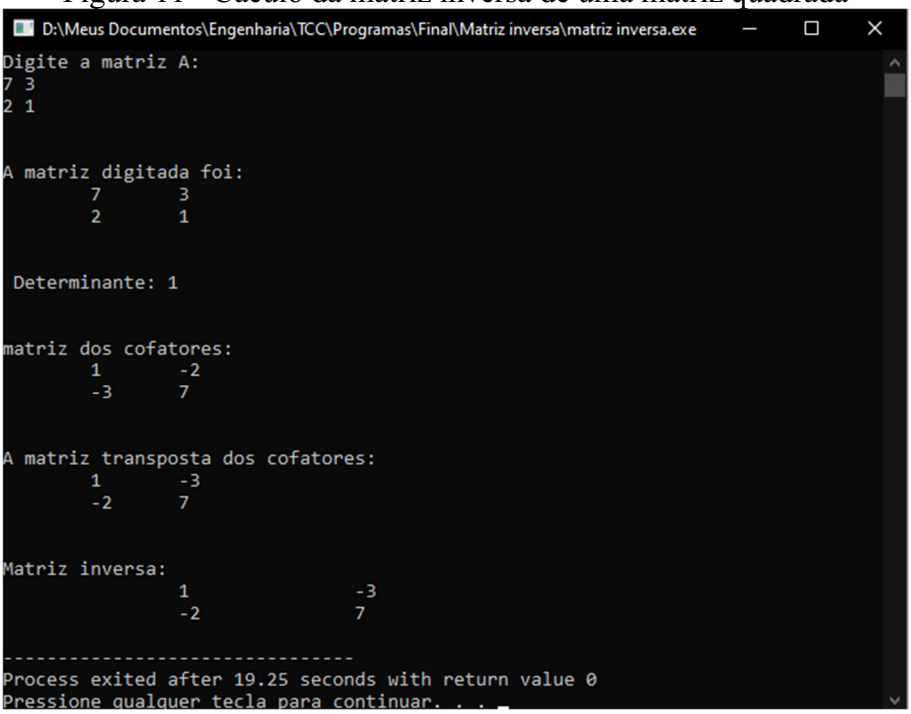
Para *w=0, e w<m, incrementar 1 ao valor de "w"*

Para *h=0 e h<m, incrementar 1 ao valor de "h"*

Escreva *inversa[w][h]*

Fim

Figura 11 - Cáculo da matriz inversa de uma matriz quadrada



5. CONCLUSÃO

A partir dos algoritmos construídos e da sua implementação e execução na linguagem de programação C/C++, acredito que estes são uma ferramenta metodológica bastante útil no estudo e ensino da Álgebra Linear. Deste feito, usa-se uma metodologia de estudo baseada no uso da programação, como recurso para o estudo da matemática. Mostrando assim, a importância da interdisciplinaridade entre estas áreas. Este trabalho abre novas oportunidades para o aprofundamento e criação de outros algoritmos para tópicos mais avançados da álgebra linear.

6. REFERÊNCIAS

BOLDRINI, J. et al. **Álgebra linear**. 3.ed. São Paulo: Harbra. 1980.

COIMBRA, J. Alguns aspectos problemáticos relacionados ao ensino-aprendizagem da álgebra linear. 2008. 70 p. Dissertação de mestrado, Universidade Federal do Pará, 2008. Disponível em: <http://repositorio.ufpa.br/jspui/bitstream/2011/1801/1/Dissertacao_AlgunsAspectosProblematicos.pdf>. Acesso em: 10 ago. 2021.

DE MELO, V.; ZANONI, E. O uso da tecnologia na educação. **Anápolis Digital**, Anápolis, v. 5, n.1, jun. 2018. Disponível em: <https://portaleducacao.anapolis.go.gov.br/revistaanapolisdigital/wp-content/uploads/vol5/11.pdf. Acesso em 10 ago. 2021.

MILAGRES, LIMA e CARDOSO. **Alglin: uma ferramenta para mediar o processo de ensino-aprendizagem em álgebra linear**. *In*: **SEMINÁRIO SUL-MATO-GROSSENSE DE PESQUISA EM EDUCAÇÃO MATEMÁTICA**, nº12, 2018, Aquidauana. Anais. Campo Grande: Divisão de Editora da UFMS, 2018, p. 268-273.

MOLINARI, J. **O software wxmáxima no ensino da matemática**. *In:* **SEMANA DE INTEGRAÇÃO ENSINO, PESQUISA E EXTENSÃO**, nº 6, 2015, Irati. Anais. Disponível em: < https://anais.unicentro.br/siepe/pdf/ivv4n1/1790.pdf>. Acesso em: 08 set. 2021.

PEREIRA, R.; COSTA, R. Álgebra Linear Numérica: Aplicações em Métodos Computacionais e sua Importância para a Engenharia. **Mecatrone**, *[S. l.]*, v. 2, n. 1, 2017. DOI: 10.11606/issn.2526-8260.mecatrone.2017.142042. Disponível em: <https://www.revistas.usp.br/mecatrone/article/view/142042>. Acesso em: 10 ago. 2021.

SANTANA, F. et al. Inovação no processo de ensino e aprendizagem de álgebra linear usando o software geogebra. **Brazilian Journal of Development**, [S.L.], v. 5, n. 9, p. 15095-15105, set. 2019. Disponível em: <https://www.brazilianjournals.com/index.php/BRJD/article/view/3209/3102>. Acesso em: 08 set. 2021.

SANTOS, E. História da disciplina álgebra linear: Primeiras aproximações. **Revista de História da Educação Matemática**, v. 4, n. 2, 20 out. 2018. Disponível em:

<http://www.histemat.com.br/index.php/HISTEMAT/article/view/221>. Acesso em: 10 ago. 2021.

SCHILDT, H. **C++**: The Complete Reference. 4 ed. New York: McGraw-Hill/Osborne, 2003.

STEINBRUCH, A.; WINTERLE, P. **Álgebra linear**. 2. ed. São Paulo: Pearson Makron Books, 1987.

TIOBE. **Índice TIOBE de outubro 2021**. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 14 out. 2021.

APÊNDICE A

Código 1: Código do cálculo do determinante de uma matriz

```
#include<iostream>
#define MAX 20
#include <cmath>
using namespace std;
double determinante(double matriz[MAX][MAX], int m);
double cofator(double matriz[MAX][MAX], int m, int linha, int coluna);
int main(){
   int i,j,m,a,b,n,h,w;
   int l=0,c=0;
   double matriz[MAX][MAX], matrizb[MAX][MAX],cofatores[MAX][MAX],det;
   cout << "\t\t\t\tPROGRAMA DE CALCULO DO DETERMINANTE\n\n";
   cout<< "Digite a ordem da matriz quadrada: ";
   cin >> m;
   while (m<0){
         cout<< "\n\nnumeros negativos sao invalodos!!";
         cout<<"\n\nDigite a ordem da matriz quadrada: ";
         cin >> m;
         }
   cout << "\n\nDigite a matriz : \n";
   for(i=0;i<m;i++){
      for(j=0;j<m;j++){
         cin >> matriz[i][j];
      }
   }
   cout << "\n\nA matriz digitada foi: \n";
      for(i=0;i<m;i++){
```

```
            for(j=0;j<m;j++){
                cout<< "\t" << matriz[i][j];
            }
            cout <<"\n";
        }
cout << "\n\n Determinante: " <<determinante(matriz,m)<<"\n";
  return 0;
}
double determinante(double matriz[MAX][MAX], int m)
{
   double det=0.000;
   if (m == 1){
      det = matriz[0][0];
   }else{
      for(int j=0; j<m; j++){
         det = det + matriz[0][j] * cofator(matriz,m, 0, j);
      }
   }
   return det;
}
double cofator(double matriz[MAX][MAX], int m, int linha, int coluna)
{
  double  submatriz[MAX][MAX];
  int n=m-1;
  int a=0;
  int b=0;
   //int j;
  for (int i=0; i<m; i++){
   for (int j=0; j<m; j++){
```

```
        if (i !=linha && j !=coluna){
            submatriz[a][b] = matriz[i][j];
            b++;
            if (b >= n){
                a++;
                b=0;
            }
        }
     }
   }
   return pow(-1,linha+coluna)*determinante(submatriz, n);
}
```

Código 2: Código do cálculo da soma de matrizes

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
void digitarmatriz(double *matriz, int l, int c);
void recebermatriz(double *matriz, int l, int c);
void somarmatrizes(double *x, double *y, double *z, int l, int c);
int main(){
    int l,c,i,j;
    double *m1, *m2, *m3;
    cout << "digite o numero de linhas: \n";
    cin >> l;
    cout << "digite o numero de colunas: \n";
    cin >> c;
```

```
    m1 = (double*)malloc(sizeof(double) * l * c);
    m2 = (double*)malloc(sizeof(double) * l * c);
    m3 = (double*)malloc(sizeof(double) * l * c);
    cout << "Digite a primeira matriz: \n";
    digitarmatriz(m1, l ,c);
    cout << "A matriz digitada foi: \n";
    recebermatriz(m1, l, c);
    cout << "Digite a segunda matriz: \n";
    digitarmatriz(m2, l ,c);
    cout << "A matriz digitada foi: \n";
    recebermatriz(m2, l, c);
    cout << "\n";
    somarmatrizes(m1,m2,m3,l,c);
    cout << "A soma das matrizes digitadas eh: \n";
    recebermatriz(m3, l, c);
    return 0;
}
void digitarmatriz(double *matriz, int l, int c)
{
    int i, j;
    double *amatriz;
    amatriz=matriz;
    for(i=0; i<l; i++)
        for(j=0; j<c; j++)
        scanf("%lf", amatriz++);
}
void recebermatriz(double *matriz, int l, int c)
{
    int i, j;
```

```
    double *amatriz;
    amatriz=matriz;
    //cout << "A matriz digitada foi: \n";
    for(i=0; i<l; i++){
        for(j=0; j<c; j++)
            printf("\t %g", *(amatriz+i*c+j)); //antes ("\t %3lf") agora ("\t %g)
        cout << "\n";
    }
}
void somarmatrizes(double *x, double *y, double *z, int l, int c)
{
    int i, j;
    double *pX=x , *pY=y , *pZ=z ;
    for(i=0; i<l; i++)
        for(j=0; j<c; j++)
        *pZ++ = *pX++ + *pY++;
}
```

Código 3: Código do cálculo da subtração de matrizes

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
void digitarmatriz(double *matriz, int l, int c);
void recebermatriz(double *matriz, int l, int c);
void somarmatrizes(double *x, double *y, double *z, int l, int c);
int main(){
    int l,c,i,j;
```

```
    double *m1, *m2, *m3;
    cout << "digite o numero de linhas: \n";
    cin >> l;
    cout << "digite o numero de colunas: \n";
    cin >> c;
    m1 = (double*)malloc(sizeof(double) * l * c);
    m2 = (double*)malloc(sizeof(double) * l * c);
    m3 = (double*)malloc(sizeof(double) * l * c);
    cout << "Digite a primeira matriz: \n";
    digitarmatriz(m1, l ,c);
    cout << "A matriz digitada foi: \n";
    recebermatriz(m1, l, c);
    cout << "Digite a segunda matriz: \n";
    digitarmatriz(m2, l ,c);
    cout << "A matriz digitada foi: \n";
    recebermatriz(m2, l, c);
    cout << "\n";
    somarmatrizes(m1,m2,m3,l,c);
    cout << "A soma das matrizes digitadas eh: \n";
    recebermatriz(m3, l, c);
    return 0;
}
void digitarmatriz(double *matriz, int l, int c)
{
    int i, j;
    double *amatriz;
    amatriz=matriz;
    for(i=0; i<l; i++)
        for(j=0; j<c; j++)
```

```
        scanf("%lf", amatriz++);
}
void recebermatriz(double *matriz, int l, int c)
{
   int i, j;
   double *amatriz;
   amatriz=matriz;
   //cout << "A matriz digitada foi: \n";
   for(i=0; i<l; i++){
      for(j=0; j<c; j++)
         printf("\t %g", *(amatriz+i*c+j)); //antes ("\t %3lf") agora ("\t %g)
      cout << "\n";
   }
}
void somarmatrizes(double *x, double *y, double *z, int l, int c)
{
   int i, j;
   double *pX=x , *pY=y , *pZ=z ;
   for(i=0; i<l; i++)
      for(j=0; j<c; j++)
      *pZ++ = *pX++ - *pY++;
}
```

Código 4: Código do cálculo da multiplicação de matrizes

```
#include <stdio.h>
#include <stdlib.h>
#include <cmath>
int main()
{
```

```
int i, j, lA, cA, lB, cB, X;
printf("\n Informe a quantidade de linhas da matriz A : ");
scanf("%d",&lA);
printf("\n Informe a quantidade de colunas da matriz A : ");
scanf("%d",&cA);
printf("\n Informe a quantidade de linhas da matriz B : ");
scanf("%d",&lB);
printf("\n Informe a quantidade de colunas da matriz B : ");
scanf("%d",&cB);
float matrizA[lA][cA], matrizB[lB][cB], matrizC[lA][cB], Aux=0; // declaração  das matrizes e variaveis
 if(cA==lB)   // condição para haver o produto das matrizes A e B
{
for(i=0; i<lA; i++)   // Leitura dos valores de entrada da matriz A
{
      for(j=0; j<cA; j++)
      {
      printf("\n\n Informe o valor da %d%c Linha e da %d%c Coluna da Matriz A: ", i+1, 167, j+1, 167);
      scanf("%f", &matrizA[i][j]);
      }
      printf("\n");
      }
for(i=0; i<lB; i++) // Leitura dos dados da matriz B
{
      for(j=0; j<cB; j++)
      {
      printf("\n\n Informe o valor da %d%c Linha e da %d%c Coluna da 2%c Matriz B: ", i+1, 167, j+1, 167, 167);
      scanf("%f", &matrizB[i][j]);
      }
```

```
        printf("\n");
        }
printf("Matriz A \n\n");  // impressão dos elmentos da matriz A
for(i=0; i<lA; i++)
{
        for(j=0; j<cA; j++)
        {
        printf("%6.f", matrizA[i][j]);
        }
        printf("\n\n");
        }
printf("Matriz B \n\n");  // impressão dos elmentos da matriz B
for(i=0; i<lB; i++)
{
        for(j=0; j<cB; j++)
        {
        printf("%6.f", matrizB[i][j]);
        }
        printf("\n\n");
        }
for(i=0; i<lA; i++)   // gearando o produto de duas matrizes
{
        for(j=0; j<cB; j++)
        {
                matrizC[i][j]=0;
                for(X=0; X<lB; X++)
                {
                Aux += matrizA[i][X] * matrizB[X][j];
                }
```

```
            matrizC[i][j]=Aux;
            Aux=0;
            }
            }
            printf("\n\n");
printf("Matriz Gerada da Multiplicação: A*B \n\n");  // Escrita na matriz C, gerada pelo produto de A e B
for(i=0; i<lA; i++)
{
      for(j=0; j<cB; j++)
      {
      printf("%6.f", matrizC[i][j]);
      }
      printf("\n\n");
      }
      printf("\n\n");
      }
else
{
printf("\n\n Nao é possivel fazer o produto das duas matrizes, pois, o numero de linhas da matriz A é diferente do numero de linhas da matriz  B \n");   // Caso la seja diferente de Ca, então não é possível muitplicar as matrizes A e B
}
system("pause");
return 0;
}
```

Código 5: Código da transposta de uma matriz

```
#include<iostream>
#include <stdio.h>
```

```
#include <stdlib.h>
using namespace std;
float leMatriz(int m, int n, float **matriz);
void imprimirDinamico(int n, int m, float **matriz);
void transposta(int a, int b, float **matrizA, float **matrizB);
int main(){
   float **matrizA, **matrizB;
   int a,b,i;
   printf("Insira as dimensoes da matriz (linha,coluna): \n");
   scanf("%d %d", &a,&b);
   /* ALOCAÇÃO DE MEMORIA PARA A MATRIZ */
   matrizA = (float**)malloc(a*sizeof(float*));
      for(i=0; i < a; i++){
         matrizA[i] = (float*)malloc(b*sizeof(float));
      }
   matrizB = (float**)malloc(b*sizeof(float*));
      for(i=0; i < b; i++){
         matrizB[i] = (float*)malloc(a*sizeof(float));
      }
      leMatriz(a,b,matrizA);
     // leMatriz(a,b,matrizB);
      cout << " A matriz digitada foi: \n";
      imprimirDinamico(a,b,matrizA);
      cout << " A matriz transposta eh: \n";
      transposta(a, b, matrizA, matrizB);
      imprimirDinamico(b,a,matrizB);
   return 0;
}
float leMatriz(int m, int n, float **matriz){
```

```
    int i,j;
    cout << "Digite a Matriz: \n";
    for(i=0; i < m; i++){
        for(j = 0; j < n; j++){
            //printf("\nInsira a posicao [%d][%d] da matriz: ", i+1, j+1);
            scanf("%f", &matriz[i][j]);
        }
    }
}
void imprimirDinamico(int n, int m, float **matriz){
    int i,j;
    for(i=0; i < n; i++){
        for(j=0;j < m; j++){
            printf("%g \t",matriz[i][j]);
        }
        printf("\n");
    }
}
void transposta(int a, int b, float **matrizA, float **matrizB){
    int i,j;
    for(i=0; i < a; i++){
        for(j=0; j < b; j++){
            matrizB[j][i] = matrizA[i][j];
        }
    }
}
```

Código 6: Código do cálculo do produto escalar

```
#include<iostream>
#include<stdlib.h>
#include<cmath>
#include<stdio.h>
#include <locale.h>
using namespace std;
int main(){
   setlocale(LC_ALL, "Portuguese");
   cout << "\n\t\t\t\tPROGRAMA DE CÁLCULO DE PRODUTO ESCALAR\n\n";
   int tam, i;
   double e=0;
   cout << "Digite a dimensão dos vetores que serão usados para o cálculo do produto escalar:
\n";
   scanf("%d",&tam);
   double a[tam],b[tam],vet[tam];
   cout << "\nDigite o vetor |a|: \n";
   for(i=0; i<tam;i++){
      cin >> a[i];
   }
   cout << "\nO vetor |a| digitado foi: \n";
   for(int i=0; i<tam; i++){
                  cout << "\t" << a[i];
      }
      cout << "\n";
   cout << "\nDigite o vetor |b|: \n";
   for(i=0; i<tam; i++){
      cin >> b[i];
   }
```

```
 cout << "\nO vetor |b| digitado foi: \n";
   for(int i=0; i<tam; i++){
                    cout << "\t" << b[i];
      }
      cout << "\n";
   for(i=0; i<tam; i++){
      vet[i]=a[i]*b[i];
   }
   /*cout << "Numeros a serem somados: ";
   for(int i=0; i<tam; i++){
         if(i==0){
            cout <<  vet[i];
         }else{cout << "+"  << vet[i];}
      }
      */
      cout << "\n";
   for(i=0; i<tam; i++){
      e=e+vet[i];
   }
   cout << "\nO Produto escalar |a|.|b| resulta em: \n\n " << e << "\n\n";
   system("pause");
   return 0;
}
```

Código 7: Código do cálculo do produto vetorial

```
#include <iostream>
#include <stdio.h>
#include <math.h>
```

```
using namespace std;
int main() {
   int tam;
   double vet[3], a[3], b[3];
   int i;
   cout << “Programa de 69xecução de Produto Vetorial\n”;
   cout << “\nDigite o vetor ‘a’: \n”;
   for(i=0; i<3;i++){
      cin >> a[i];
   }
   cout << “\nDigite o vetor ‘b’: \n”;
   for(i=0; i<3; i++){
      cin >> b[i];
   }
   vet[0]= (a[1]*b[2] – b[1]*a[2]);
   vet[1]= (a[2]*b[0] – b[2]*a[0]);
   vet[2]= (a[0]*b[1] – b[0]*a[1]);
   cout << “ Produto vetorial |a ^ b|: \n”;
   cout << “(“ << vet[0] <<”,” ;
   cout << “ “ << vet[1] << “,”;
   cout << “ “ << vet[2] << “ )”;
   return 0;
}
```

Código 8: Código para resolução de sistema linear por eliminação de Gauss

```
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define MAX 10 //O numero de equações (linhas do sistema linear)
```

```
using namespace std;

int main (){

  setlocale(LC_ALL, "Portuguese");

  cout << "\n\n\t\t\t\tRESOLUÇÃO DE SISTEMAS LINEARES - ELIMINAÇÃO DE GAUSS\n\n\n\n\n";

   int  p, j, k,a,b,c,x,y,V;

   float fator, soma;

              /*= {{10, 15, 105, 204.37},

                 {15, 105, 405, 682.15},

                 {105, 405, 2373, 4377.47}};                    */

   cout << "Digite o número de equações do sistema: ";

   cin >> V;

   while(V < 0 || V > MAX){

      cout << "O número de equações não pode ser negativo, e o número de equações não pode ser maior que" << MAX;

      cout << "Digite o número de equações do sistema: ";

      cin >> V;

   }

   int V2=V+1;

   float L[V];

   float M[V][V2];

    cout << "Digite a Matriz do sistema de equações: \n";

       for(x=0 ; x<V ; x++){

         for(y=0 ; y<V2 ; y++){

            cin >> M[x][y];//scanf("%lf", &M[x][y]);

         }

         printf("\n");

       }

   cout << "Matriz do sistema de equações digitada: \n\n";

   for(p = 0; p<V; p++){
```

```
        for(j = 0; j<=V; j++){
        cout << "\t|"  << M[p][j];//printf("\t |%lf", M[p][j]);
        }
        printf("\n");
    }
// ELIMINACAO PROGRESSIVA
    for(k = 0; k<V-1; k++){//Laco exterior se refere a coluna pivo que se altera. (linha pivo ou equacao pivo = equacao referencial para as eliminacoes)
        for(p = (k+1); p<V; p++){//Este laco se refere as linhas que sofrerao as eliminacoes, que sempre iniciarao na linha seguinte da linha pivô
         fator = M[p][k] / M[k][k];//fator de correcao que sera multiplicada pela linha pivo
for(j=0; j<=V; j++){//loco interno se refere a cada coluna da linha que sofrerao modificacoes da linha i
              M[p][j] = M[p][j] - fator*M[k][j];
           }
        }
    }
    L[V-1] = M[V-1][V] / M[V-1][V-1]; // Primeria variavel que obtemos
     cout << "Matriz escalonada: \n";
    printf("\n");
    for(p = 0; p<V; p++){
        for(j = 0; j<=V; j++){
           cout << "\t|" << M[p][j];//printf(" \t|%.3f\t", M[p][j]);
        }
        printf("\n");
    }
// SUBSTITUICAO PROGRESSIVA
    for(p=V-2; p>=0; p--){  //Laço exterior se refere a linha que recebera os valores das variaveis ja encontradas para descobrirmos  as outras. Ela se inicia na penultima linha e vai ate a primeira
        soma = 0;
        for(j=(p+1); j<V; j++){     //Laco interno se refere as colunas da linha i
```

```
            soma = soma + M[p][j]*L[j]; // soma todos os numeros da linha que ja tem o valor de x
        }
        L[p] = (M[p][V] - soma)/M[p][p]; // Calcula o x da coluna i
    }
    printf("\nResultado: " );
    for(p=0; p<V; p++)
        printf("\nL%d = %f\n", p+1, L[p]);
//L[0]=c;
//L[1]=b;
//L[2]=a;
 cout << "valor de a= " << L[2];
        return 0;
}
```

Código 9: Código da resolução de sistemas lineares pelo método de Jacobi

```
#include <stdio.h>
#include <stdlib.h>
#include <cmath>
#include <locale.h>
typedef int boolean;
#define true 1
#define false 0
boolean verificar_linhas(double **matriz, int dimensao);
boolean verificar_colunas(double **matriz, int dimensao);
double testar_convergencia(double *x_anterior, double *x_proximo, int dimensao);
int main(void)
{
setlocale(LC_ALL, "Portuguese");
```

```
 printf("\n\t\t\t\tRESOLUÇÃO DE SISTEMAS LINEARES - MÉTODO DE JACOBI \n\n\n");
printf("\t\t\t\t   Separe as casas decimais com virgula(,)\n\n\n");
   int dimA;
   printf("Digite a dimensão da matriz A: ");
   scanf("%d", &dimA);
   //Cria a matriz
   double **matriz_a = (double **)calloc(dimA, sizeof(double *));
   for (int i = 0; i < dimA; i++)
   {
      matriz_a[i] = (double *)calloc(dimA, sizeof(double));
   }
   double *matriz_b = (double *)calloc(dimA, sizeof(double));
   double *x0 = (double *)calloc(dimA, sizeof(double));
   //Preenche a matriz A:
   printf("Preenchendo a matriz A:\n\n");
   for (int i = 0; i < dimA; i++)
   {
      printf("Preenchendo a linha %d\n", i + 1);
      for (int j = 0; j < dimA; j++)
      {
         printf("Elemento %d: ", j + 1);
         scanf("%lf", &matriz_a[i][j]);
      }
   }
   //Preenche a matriz B:
   printf("\n\nPreenchendo a matriz B:\n");
   for (int i = 0; i < dimA; i++)
   {
      printf("Elemento %d: ", i + 1);
      scanf("%lf", &matriz_b[i]);
```

```
}
//Verificar se é diagonal dominante
//Verificando por linha:
boolean teste = verificar_linhas(matriz_a, dimA);
if (teste == false)
{
   teste = verificar_linhas(matriz_a, dimA);
}
int num_ite = 0;
if (teste == false)
{
   printf("O método não converge pela verificação da diagonal dominante.\n");
   printf("Digite um número de iterações > 0 para aplicar o método mesmo assim: ");
   scanf("%d", &num_ite);
   if (num_ite <= 0)
   {
      return -1;
   }
}
double precisao;
printf("\nDigite a precisão do método: ");
scanf("%lf", &precisao);
//Criando o valor x0 (inicial)
for (int i = 0; i < dimA; i++)
{
   x0[i] = matriz_b[i] / matriz_a[i][i];
}
//Criando vetor de iterações
double *xk = (double *)calloc(dimA, sizeof(double));
```

```
double *xt = (double *)calloc(dimA, sizeof(double));
if (num_ite > 0)
{
   for(num_ite; num_ite > 0; num_ite--){
      //Calcula os valores de xk
      for (int i = 0; i < dimA; i++)
      {
         double soma = matriz_b[i];
         for (int j = 0; j < dimA; j++)
         {
            if (j != i)
            {
               soma -= x0[j] * matriz_a[i][j];
            }
         }
         xk[i] = (1.0 / matriz_a[i][i]) * (soma);
      }
      for (int i = 0; i < dimA; i++)
      {
         xt[i] = x0[i];
         x0[i] = xk[i];
      }
   }
}
else
{
   do
   {
      //Calcula os valores de xk
```

```
        for (int i = 0; i < dimA; i++)
        {
            double soma = matriz_b[i];
            for (int j = 0; j < dimA; j++)
            {
                if (j != i)
                {
                    soma -= x0[j] * matriz_a[i][j];
                }
            }
            xk[i] = (1.0 / matriz_a[i][i]) * (soma);
        }
        for (int i = 0; i < dimA; i++)
        {
            xt[i] = x0[i];
            x0[i] = xk[i];
        }
    } while (testar_convergencia(xt, xk, dimA) > precisao);
}
//Mostra o resultado:
printf("\n\nResultado: \n");
for (int i = 0; i < dimA; i++)
{
    printf("x%d = %lf\n", i, xk[i]);
}
//Libera a matriz
for (int i = 0; i < dimA; i++)
{
    free(matriz_a[i]);
```

```
    }
    free(matriz_a);
    free(matriz_b);
    free(x0);
    free(xk);
    free(xt);
    return 0;
}
boolean verificar_linhas(double **matriz, int dimensao)
{
    for (int i = 0; i < dimensao; i++)
    {
        int soma = 0;
        for (int j = 0; j < dimensao; j++)
        {
            if(i != j){
                soma += matriz[i][j];
            }
        }
        if (soma > matriz[i][i])
            return false;
    }
    return true;
}
boolean verificar_colunas(double **matriz, int dimensao)
{
    for (int i = 0; i < dimensao; i++)
    {
        int soma = 0;
```

```
        for (int j = 0; j < dimensao; j++)
        {
            if(i != j){
                soma += matriz[i][j];
            }
        }
        if (soma > matriz[i][i])
            return false;
    }
    return true;
}
double testar_convergencia(double *x_anterior, double *x_proximo, int dimensao)
{
    double maior_num = 0, maior_den = 0;
    for (int i = 0; i < dimensao; i++)
    {
        if (abs(x_proximo[i] - x_anterior[i]) > maior_num)
        {
            maior_num = abs(x_proximo[i] - x_anterior[i]);
        }
        if (abs(x_proximo[i]) > maior_den)
        {
            maior_den = abs(x_proximo[i]);
        }
    }
    return maior_num / maior_den;
}
```

Código 10: Código do cálculo da matriz inversa

```
#include<iostream>
#define MAX 20
#include <cmath>
using namespace std;
double determinante(double matriz[MAX][MAX], int m);
double cofator(double matriz[MAX][MAX], int m, int linha, int coluna);
int main(){
   int i,j,m,a,b,n,h,w;
   int l=0,c=0;
   double matriz[MAX][MAX],
matrizb[MAX][MAX],cofatores[MAX][MAX],inversa[MAX][MAX],transposta[MAX][MA
X],det;
   cout << "\t\t\t\tPROGRAMA DE CALCULO DE MATRIZ INVERSA\n\n";
   cout<< "Digite a ordem da matriz quadrada: ";
   cin >> m;
   cout << "\n\nDigite a matriz A: \n";
   for(i=0;i<m;i++){
      for(j=0;j<m;j++){
         cin >> matriz[i][j];
      }
   }
   cout << "\n\nA matriz digitada foi: \n";
      for(i=0;i<m;i++){
         for(j=0;j<m;j++){
            cout<< "\t" << matriz[i][j];
         }
         cout <<"\n";
      }
```

```
cout << "\n\n Determinante: " <<determinante(matriz,m)<<"\n";
            //det=determinante(matriza,m);
            //cout << det;
            if(determinante(matriz,m)==0){
               cout << "\n\n\tA MATRIZ NAO PODE SER INVERTIDA!!\n\n";
            }if(m==1){
               matriz[0][0]=pow(matriz[0][0],-1);
            cout << "\n\nA matriz inversa: \n" << matriz[0][0];
            }if(determinante(matriz,m)!=0 && m>=2){   //calculo dos cofatores
         for(i=0;i<m;i++){
            for(j=0;j<m;j++){
               cofatores[i][j] = cofator(matriz,m,i,j);
            }
            cout << "\n";
         }
          cout << "matriz dos cofatores: \n";
         for(i=0;i<m;i++){
            for(j=0;j<m;j++){
               //cout<<"\t" << cofator(matriza,m,i,j);
               cout << "\t" << cofatores[i][j];
            }
            cout << "\n";
         }
         //TRANSPONDO A MATRIZ DOS COFATORES
            for(h=0;h<m;h++){
               for(w=0;w<m;w++){
                  transposta[w][h]=cofatores[h][w];
               }
            }
```

```
            cout << "\n\nA matriz transposta dos cofatores: \n";
        for(w=0;w<m;w++){
            for(h=0;h<m;h++){
                cout<< "\t"  <<transposta[w][h];
            }
            cout <<"\n";
        }
        // CALCULO DA MATRIZ INVERSA
        for(w=0;w<m;w++){
            for(h=0;h<m;h++){
                inversa[w][h]=transposta[w][h]/determinante(matriz,m);
            }
        }
        cout << "\n\nMatriz inversa: \n";
            // if(determinante(matriz,m)==0){
                    //cout << "\n\n\tA MATRIZ NAO PODE SER INVERTIDA!!\n\n";
//}else{
        for(w=0;w<m;w++){
            for(h=0;h<m;h++){
                cout << "\t\t" << inversa[w][h];
            }
            cout << "\n";
        }
            // }
             }
    return 0;
}
double determinante(double matriz[MAX][MAX], int m)
{
```

```
    double det=0.0;
    if (m == 1){
        det = matriz[0][0];
    }else{
        for(int j=0; j<m; j++){
            det = det + matriz[0][j] * cofator(matriz,m, 0, j);
        }
    }
    return det;
}
double cofator(double matriz[MAX][MAX], int m, int linha, int coluna)
{
   double  submatriz[MAX][MAX];
   int n=m-1;
   int a=0;
   int b=0;
    //int j;
   for (int i=0; i<m; i++){
    for (int j=0; j<m; j++){
        if (i !=linha && j !=coluna){
            submatriz[a][b] = matriz[i][j];
            b++;
            if (b >= n){
                a++;
                b=0;
            }
        }
     }
   }
```

```
  return pow(-1,linha+coluna)*determinante(submatriz, n);
}
```