

Universidade Federal dos Vales do Jequitinhonha e
Mucuri

Instituto de Ciência e Tecnologia-ICT
Ciência e Tecnologia-BCT

**Projeto Proae: Elaboração de Material
Didático que Empregue o uso de Software
como Suporte para o Aprendizado de
Álgebra Linear**

(Nº do Registro: 2019.D.2.20.063.0)

por

Mônica Aparecida Cruvinel Valadão (Coordenadora)
Douglas Frederico Guimarães Santiago (Vice-Coordenador)
Anderson Luiz Pedrosa Porto (Colaborador)
Flaviano Luiz Benfca (Bolsista)

Diamantina-MG
Dezembro de 2020

Sumário

1	INTRODUÇÃO	4
2	ROTEIRO DE INSTALAÇÃO <i>OCTAVE</i>	5
2.1	Instalando o Octave no Windows	5
2.2	Instalando o Octave no Linux	7
3	TUTORIAL FUNÇÕES	10
3.1	Construa uma Matriz 1	10
3.2	Construa uma Matriz 2	11
3.3	Soma de Matrizes	12
3.4	Subtração de Matrizes	13
3.5	Produto de Matrizes	13
3.6	Produto de Um Escalar por Uma Matriz	14
3.7	Transposta de Uma Matriz	14
3.8	Forma Escalonada Reduzida de Uma Matriz	15
3.9	Matriz Ampliada e Forma Escalonada Reduzida de Uma Matriz	15
3.10	Verifica Solução Sistema Linear	16
3.11	Solução de Um Sistema de Equações Lineares	18
3.12	Determinate de Uma Matriz	19
3.12.1	Escolhendo Uma Linha Para Calcular o Determinante	20
3.12.2	Escolhendo Uma Coluna Para Calcular o Determinante	21
3.13	Matriz dos Cofatores	22
3.14	Matriz Adjunta	24
3.15	Matriz Inversa	24
3.16	Polinômio Característico	25
3.17	Autovalores	26
3.18	Autovetores Associados a Um Autovalor	26
3.19	Verifica Autovalor	28
3.20	Verifica Autovetor	28
4	ATIVIDADES USANDO O OCTAVE	30
4.1	Atividade 1	31
4.2	Atividade 2	33
4.3	Atividade 3	33
4.4	Atividade 4	34
4.5	Atividade 5	35

4.6	Atividade 6	36
4.7	Atividade 7	36
4.8	Atividade 8	37
4.9	Atividade 9	37
4.10	Atividade 10	37
4.11	Atividade 11	38
4.12	Atividade 12	39
5	CONSIDERAÇÕES FINAIS	40
	REFERÊNCIAS	41
	APÊNDICES	42
	APÊNDICE A – CÓDIGO DAS FUNÇÕES IMPLEMENTADAS . .	43

1 Introdução

A disciplina de Álgebra Linear ocupa um papel de destaque na formação do curso de Ciência e Tecnologia. O conteúdo estudado nessa disciplina é indispensável para disciplinas como Funções de Várias Variáveis, Cálculo Numérico, Fenômenos Mecânicos, Linguagens de Programação, entre outras, presentes nas grades dos cursos lotados no Instituto de Ciência e Tecnologia (PROJETO... , 2020). Dessa forma, o desempenho do discente no decorrer do curso está relacionado com o conhecimento assimilado em Álgebra Linear. Nesse contexto, deve-se pensar em atividades que contribuam com o aprendizado dos discentes nessa disciplina.

Embora os conceitos de Álgebra Linear sejam fundamentais para diversas outras áreas, observa-se que muitos discentes não se sentem estimulados com essa disciplina. Nesse sentido, o uso de software como suporte de ensino de Álgebra Linear é uma alternativa que pode ser adotada para tornar essa disciplina mais atrativa. Várias referências adotadas nessa disciplina já apresentam atividades que envolvem o uso ferramentas computacionais (LAY, 2007; ANTON; RORRES, 2001; POOLE, 2011). Estas atividades podem ser adaptadas para o uso de softwares gratuitos como Octave (EATON DAVID BATEMAN, 2020) ou outra linguagem similar a esta.

O projeto “Proae Elaboração de Material Didático que Empregue o uso de Software como Suporte para o Aprendizado de Álgebra Linear”, com registro na Prograd (Nº do Registro: 2019.D.2.20.063.0) (PROAE... , 2020), é apresentado como uma proposta que permite abordar diferentes conceitos de Álgebra Linear de uma forma mais interessante. Nesse projeto dedicou-se a uma elaboração detalhada de material didático voltado para os tópicos de Matrizes, Sistema de Equações Lineares, Determinantes, Autovalores e Autovetores. Esse material didático é composto por duas partes. A primeira parte corresponde à um conjunto de funções implementadas em linguagem Octave (arquivos “.m”). A outra parte corresponde ao presente texto, o qual apresenta detalhes de como utilizar os arquivos “.m” no Octave. Esse material é acessível aos discentes que não possuem nenhum conhecimento prévio sobre o software adotado. Todo o material do projeto está disponível no endereço <<https://github.com/monicavaladao/Proae2020>>. O restante desse texto é estruturado conforme a seguir. O Capítulo 2 apresenta orientações sobre a instalação do software Octave. Os detalhes de como executar no Octave cada função implementada em arquivo “.m” são apresentados no Capítulo 3; cada subseção deste capítulo está relacionada à um conceito de Álgebra Linear. O Capítulo 4 apresenta diferentes atividades que podem ser realizadas com o auxílio do Octave. As considerações finais sobre o desenvolvimento desse material são apresentadas na Capítulo 5.

2 Roteiro de Instalação *OCTAVE*

Neste capítulo são apresentadas instruções para download e instalação do Octave (GNU..., 2020; EATON DAVID BATEMAN, 2020).

2.1 Instalando o Octave no Windows

1. Em seu navegador acesse o link de dowload do octave <<https://www.gnu.org/software/octave/download.html>>.
2. Em seguida, na tela exibida na Figura 1, escolha o sistema operacional de seu computador.

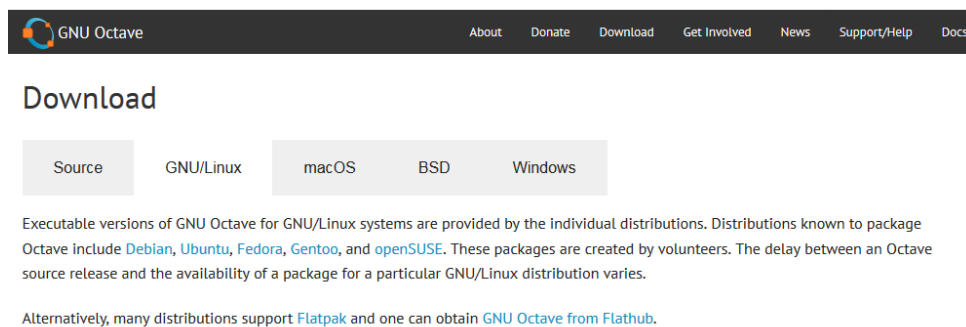


Figura 1 – GNU Octave.

3. Escolha a opção correspondente ao seu sistema operacional, conforme ilustrado na Figura 2, e aguarde o dowload.
4. Após o dowload execute o arquivo de instalação clicando nele. Basta ir avançado que o próprio programa já será instalado com as preferências padrão.

Caso não saiba se seu sistema operacional windows é 32 ou 64 bits, siga os passos seguintes.

1. Aperte e segure a tecla Windows em seguida aperte a tecla “Pause Break” em seu teclado e pule para o tópico 4, caso seu teclado não possua as teclas siga para o passo abaixo.
2. Em seu computador abra a pasta meus documentos, ou a pasta principal escrito meu computador.
3. No canto esquerdo em cima da palavra computador clique com o botão direito e vá em propriedades.

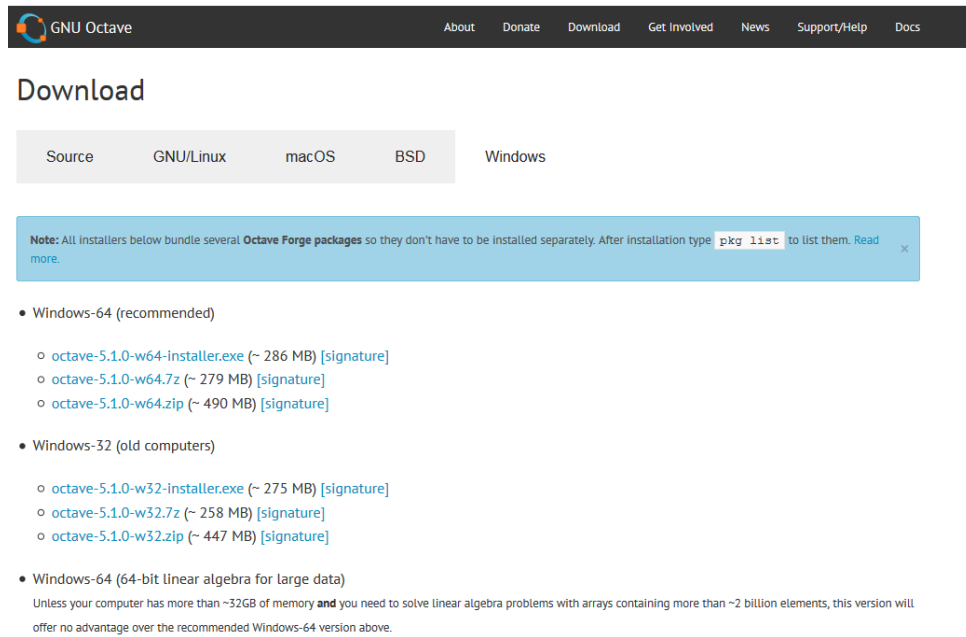


Figura 2 – GNU Octave.

4. Em seguida basta verificar se estará escrito 64 ou 32 bits e escolher a opção correspondente no passo 3.

Caso tenha clicado em meu computador aparecerá “Disco Local C:”, basta clicar com o botão direito em cima também e ir em propriedades.

Observação 1. *As imagens foram cortadas para reduzir o espaço desnecessário e irão diferenciar entre o windows 7, 8, 10 mas o processo em si não sofre alteração.*

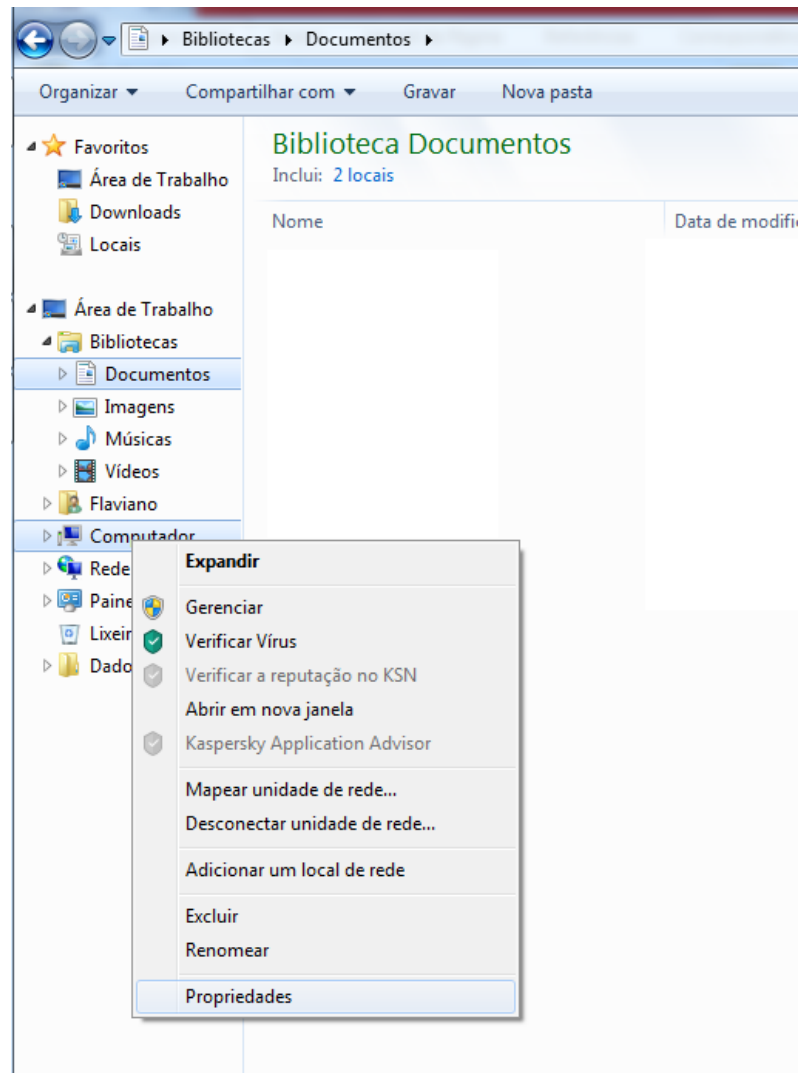


Figura 3 – GNU Octave.

2.2 Instalando o Octave no Linux

A instalação no Linux em geral depende da distribuição Linux que você utiliza. Daremos três opções de instalação baseadas no Ubuntu. A primeira delas através do comando `apt install`, baseado em pacotes nos repositórios oficiais. As outras duas formas representam formas mais recentes de instalação com maior independência em relação a pacotes externos. Estes métodos também são aplicáveis para a maioria das outras distribuições. Nestes casos, sigam os links indicados.

1. Boa parte das distribuições linux fornece o Octave na forma de pacotes alojados em seus repositórios oficiais. É o caso das principais distribuições baseadas em Debian, como Ubuntu. Neste, basta abrir o terminal e digitar o comando a seguir e pressionar enter (será exigida a senha de superusuário):

`sudo apt install octave.`

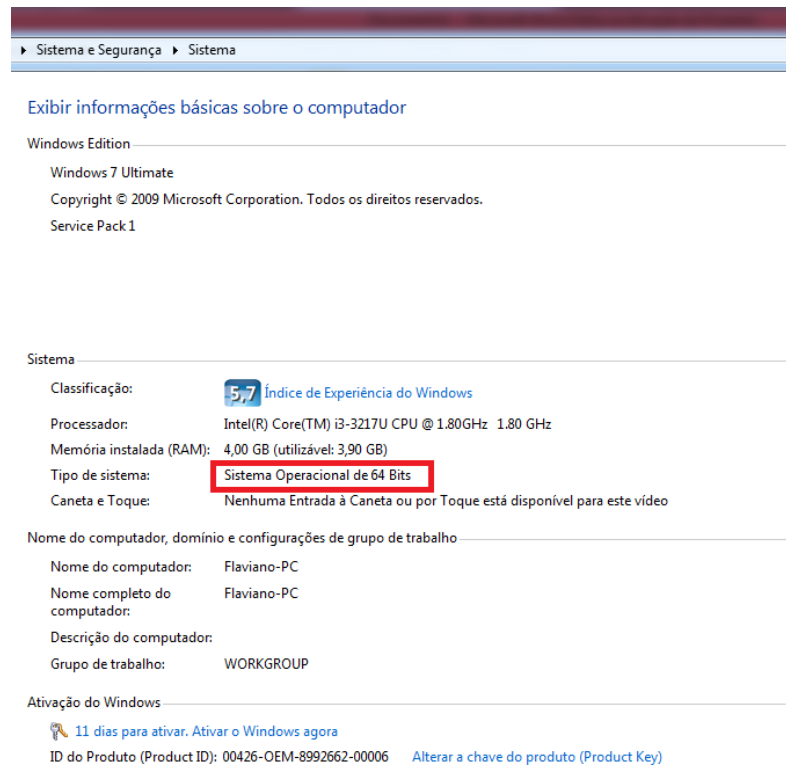


Figura 4 – GNU Octave.

De forma alternativa, pode-se também acessar via ícones o aplicativo de instalação e procurar pelo software.

2. Nas versões mais recentes de diversas distribuições linux, você pode também instalar via snap. No Ubuntu, a partir da versão 16.04 *LTS*, esta opção já vem habilitada. Em versões anteriores ou outras distribuições, você pode habilitar o snap. Para isto, siga as instruções em <https://snapcraft.io/docs/installing-snapd>. Se esta opção já estiver habilitada, basta digitar:

```
sudo snap install octave.
```

3. Uma terceira forma de instalação consiste em utilizar flatpaks. Várias distribuições linux já tem suporte à esta ferramenta. Para maiores detalhes, acesse <https://flatpak.org/setup/>. A partir da versão 18.10, digite:

```
sudo apt install flatpak.
```


Você pode então instalar o plugin que permite que seu aplicativo de instalação trabalhe com flatpaks. Para maiores informações consulte [<https://flatpak.org/setup/Ubuntu/>](https://flatpak.org/setup/Ubuntu/). Não utilizaremos esta ferramenta, pois você pode também instalar via linha de comando no terminal. De qualquer forma, primeiro é necessário adicionar o repositório de aplicativos do flathub através do comando:

```
flatpak remote --add --if-not-exists  
flathub https://flathub.org/repo/flathub.flatpakrepo.
```

Todos os passos anteriores feitos, basta digitar

```
flatpak install flathub org.octave.Octave.
```

Para consultar mais aplicativos possíveis de serem instalados via flatpaks, visite [<https://flathub.org/home>](https://flathub.org/home).

3 Tutorial Funções

Este capítulo apresenta orientações para executar no Octave cada função implementada nos arquivos “.m” disponíveis no endereço <https://github.com/monicavaladao/Proae2020>, conforme apresentado do no Capítulo 1.

3.1 Construa uma Matriz 1

O arquivo “construa_matriz.m” implementa a função

$$[A] = \text{construa_matriz}(\text{regra_ii}, \text{regra_ij}, m, n).$$

As entradas dessa função fornecem a definição para a construção de uma matriz $A = [a_{ij}]_{m \times n}$, onde

regra_ii : regra associada a condição dos índices $i = j$;

regra_ij : regra associada a condição $i \neq j$.

Essas regras devem ser inseridas no formato de função anônima $@(i, j)()$.

Os Exemplos 1 e 2 ilustram a construção de uma matriz $A = [a_{ij}]_{4 \times 7}$ definida por

$$a_{ij} = \begin{cases} i - j, & \text{se } i = j \\ i, & \text{se } i \neq j \end{cases}$$

Exemplo 1.

```
>> [A] = construa_matriz(@(i,j)(i-j), @(i,j)(i), 4, 7)
A =
```

```
0 1 1 1 1 1 1
2 0 2 2 2 2 2
3 3 0 3 3 3 3
4 4 4 0 4 4 4
```

Exemplo 2.

```
>> regra_ii = @(i,j)(i-j)
```

```
regra_ii =
```

```
    @(i,j)(i-j)
```

```
>> regra_ij = @(i,j)(i)
```

```
regra_ij =
```

```
    @(i,j)(i)
```

```
>> m = 4
```

```
m =
```

```
4
```

```
>> n = 7
n =
    7
>> [A] = construa_matriz(regra_ii, regra_ij, m, n)
A =
    0  1  1  1  1  1  1
    2  0  2  2  2  2  2
    3  3  0  3  3  3  3
    4  4  4  0  4  4  4
```

3.2 Construa uma Matriz 2

O arquivo “construa_matriz_caso_geral.m” implementa a função

$[A] = \text{construa_matriz_caso_geral}(\text{regra_ii}, \text{regra_ij}, \text{cond_ij}, \text{valor_cond}, m, n).$

As entradas dessa função fornecem a definição para a construção de uma matriz $A = [a_{ij}]_{m \times n}$, onde

regra_ii : regra associada a condição dos índices $\text{cond_ij} = \text{valor_cond}$;

regra_ij : regra associada a condição dos índices $\text{cond_ij} \neq \text{valor_cond}$.

Essas regras e condições devem ser inseridas no formato de função anônima $@(i, j)()$.

Os Exemplos 3 e 4 ilustram a construção de uma matriz $A = [a_{ij}]_{4 \times 7}$ definida por

$$a_{ij} = \begin{cases} 2.i, & \text{se } i + j = 4 \\ j + 2, & \text{se } i + j \neq 4 \end{cases}$$

Exemplo 3.

```
>> [A] = construa_matriz_caso_geral(@(i,j)(2*i), @(i,j)(j+2), @(i,j)(i+j), 4, 4, 7)
A =
    3  4  2  6  7  8  9
    3  4  5  6  7  8  9
    6  4  5  6  7  8  9
    3  4  5  6  7  8  9
```

Exemplo 4.

```
>> regra_ii = @(i,j)(2*i)
regra_ii =
    @(i,j)(2*i)

>> regra_ij = @(i,j)(j+2)
regra_ij =
    @(i,j)(j+2)
```

```
>> cond_ij = @(i,j)(i + j)
cond_ij =
        @(i,j)(i + j)

>> valor_cond = 4
valor_cond =
        4

>> m = 4
m =
        4

>> n = 7
n =
        7

>> [A] = construa_matriz_caso_geral(regra_ii, regra_ij, cond_ij, valor_cond, m, n)
A =
        3  4  2  6  7  8  9
        3  4  5  6  7  8  9
        6  4  5  6  7  8  9
        3  4  5  6  7  8  9
```

3.3 Soma de Matrizes

O arquivo “soma_matrizes.m” implementa a função

$$[C] = \text{soma_matrizes}(A, B),$$

cujas entradas são matrizes $A = [a_{ij}]_{m \times n}$ e $B = [b_{ij}]_{m \times n}$. A saída dessa função é uma matriz $C = [c_{ij}]_{m \times n}$ dada por $C = A + B$.

Exemplo 5.

```
>> A = [1, 2, 3; 4, 5, 6]
A =
        1  2  3
        4  5  6

>> B = [3, 2, 1; 9, 8, 7]
B =
        3  2  1
        9  8  7

>> C = soma_matrizes(A, B)
C =
        4  4  4
       13 13 13
```

3.4 Subtração de Matrizes

O arquivo “subtracao_matrizes.m” implementa a função

$$[C] = \text{subtracao_matrizes}(A, B),$$

cujas entradas são matrizes $A = [a_{ij}]_{m \times n}$ e $B = [b_{ij}]_{m \times n}$. A saída dessa função é uma matriz $C = [c_{ij}]_{m \times n}$ dada por $C = A - B$.

Exemplo 6.

```
>> A = [1, 2, 3; 4, 5, 6]
A =
     1     2     3
     4     5     6
>> B = [3, 2, 1; 9, 8, 7]
B =
     3     2     1
     9     8     7
>> C = subtracao_matrizes(A, B)
C =
    -2     0     2
    -5    -3    -1
```

3.5 Produto de Matrizes

O arquivo “produto_matrizes.m” implementa a função

$$[C] = \text{produto_matrizes}(A, B),$$

cujas entradas são matrizes $A = [a_{ij}]_{m \times n}$ e $B = [b_{jk}]_{n \times p}$. A saída dessa função é uma matriz $C = [c_{ik}]_{m \times p}$ dada por $C = A \cdot B$.

Exemplo 7.

```
>> A = [1, 2, 3; 4, 5, 6]
A =
     1     2     3
     4     5     6
>> B = [7, 7, 9, 10; 11, 12, 13, 14; 1, 2, 3, 4]
B =
     7     7     9    10
    11    12    13    14
     1     2     3     4
```

```
>> [C] = produto_matrizes(A, B)
C =
    32    37    44    50
    89   100   119   134
```

3.6 Produto de Um Escalar por Uma Matriz

O arquivo “produto_matriz_por_escalar.m” implementa a função

$$[C] = \text{produto_matriz_por_escalar}(A, k).$$

Essa função recebe como entrada uma matriz $A = [a_{ij}]_{m \times n}$ e um escalar k . A saída dessa função é uma matriz $C = [c_{ij}]_{m \times n}$ dada por $C = k \cdot A$.

Exemplo 8.

```
>> A = [1, 2, 3; 4, 5, 6]
A =
     1     2     3
     4     5     6
>> k = 4
k =
     4
>> C = produto_matriz_por_escalar(A, k)
C =
     4     8    12
     8    20    24
```

3.7 Transposta de Uma Matriz

O arquivo “transposta_matriz.m” implementa a função

$$[C] = \text{transposta_matriz}(A).$$

Essa função recebe como entrada uma matriz $A = [a_{ij}]_{m \times n}$. A saída dessa função é uma matriz $C = [c_{ji}]_{n \times m}$ tal que $C = A^T$.

Exemplo 9.

```
>> A = [3, 5, 1; 2, 4, 7; 3, 1, 8]
A =
     3     5     1
     2     4     7
     3     1     8
```

```
>> [C] = transposta_matriz(A)
C =
     3     2     3
     5     4     1
     1     7     8
```

3.8 Forma Escalonada Reduzida de Uma Matriz

O arquivo “forma_escalonada_reduzida_matriz.m” implementa a função

$$[E, pA, nA] = \text{forma_escalonada_reduzida_matriz}(A),$$

cuja a entrada é uma matriz $A = [a_{ij}]_{m \times n}$. As saídas dessa função são:

$E = [e_{ij}]_{m \times n}$: matriz na forma escalonada reduzida equivalente a A ;

ou

$E = []$;

pA : posto de A ;

nA : nulidade de A .

Exemplo 10.

```
>> A = [1, 2, -3, 0; 2, 4, -2, 2; 3, 6, -4, 3]
A =
     1     2    -3     0
     2     4    -2     2
     3     6    -4     3
>> [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
E =
     1     2     0     0
     0     0     1     0
     0     0     0     1
pA = 3
nA = 1
```

3.9 Matriz Ampliada e Forma Escalonada Reduzida de Uma Matriz

O arquivo “escalonada_reduzida_matriz_ampliada.m” implementa a função

$$[A, E] = \text{escalonada_reduzida_matriz_ampliada}(C, B).$$

As entradas das dessa função são uma matriz $C = [c_{ij}]_{m \times n}$ e $B = [b_{jk}]_{m \times 1}$. As matrizes C e B , representam, respectivamente, a matriz dos coeficientes e matriz dos termos independentes associados a um sistema de equações lineares. As saídas dessa função são uma

matriz $A = [a_{ij}]_{m \times (n+1)}$ e uma matriz $E = [e_{ij}]_{m \times (n+1)}$ que representam, respectivamente, a matriz ampliada e matriz na forma escalonada reduzida associadas ao sistema linear em questão. No caso de ocorrer erro numérico ao usar `rref()` retorna-se $E = []$.

Exemplo 11.

```
>> C = [3, 2, -5; 2, -4, -2; 1, -2, -3]
C =
     3     2    -5
     2    -4    -2
     1    -2    -3
>> B = [8; -4; -4]
B =
     8
    -4
    -4
>> [A, E] = escalonada_reduzida_matriz_ampliada(C, B)
A =
     3     2    -5     8
     2    -4    -2    -4
     1    -2    -3    -4
E =
     1     0     0     3
     0     1     0     2
     0     0     1     1
```

3.10 Verifica Solução Sistema Linear

O arquivo “`verifica_solucao_sistema_linear.m`” implementa a função

$$[R, B] = \text{verifica_solucao_sistema_linear}(C, B, X).$$

As entradas dessa função são matrizes da forma $C = [c_{ij}]_{n \times n}$, $B = [b_{jk}]_{n \times 1}$ e $X = [x_{jk}]_{n \times 1}$. As matrizes C , B e X representam, respectivamente, a matriz dos coeficientes, a matriz dos termos independentes e a matriz a ser verificada ser ou não solução do sistema associado $A.X = B$. As saídas dessa função são as matrizes $R = A.X$ e B .

Exemplo 12.

```
>> C = [3, 2, -5; 2, -4, -2; 1, -2, -3]
C =
     3     2    -5
     2    -4    -2
     1    -2    -3
```



```
>> B = [8; -4; -4]
B =
     8
    -4
    -4
>> X = [-2; 3; 1]
X =
    -2
     3
     1
>> [R, B] = verifica_solucao_sistema_linear(C, B, X)
R =
    -5
   -18
   -11
B =
     8
    -4
    -4
```

Exemplo 13. >> C = [3, 2, -5; 2, -4, -2; 1, -2, -3]

```
C =
     3     2    -5
     2    -4    -2
     1    -2    -3
>> B = [8; -4; -4]
B =
     8
    -4
    -4
>> X = [3; 2; 1]
X =
     3
     2
     1
>> [R, B] = verifica_solucao_sistema_linear(C, B, X)
R =
     8
    -4
    -4
```

```
B =
      8
     -4
     -4
```

3.11 Solução de Um Sistema de Equações Lineares

O arquivo “solucao_sistema_linear.m” implementa a função

$$[pA, pC, A, S, E] = \text{solucao_sistema_linear}(C, B).$$

As entradas dessa função são uma matriz $C = [c_{ij}]_{m \times n}$ e $B = [b_{jk}]_{m \times 1}$. As matrizes C e B , representam, respectivamente, a matriz dos coeficientes e matriz dos termos independentes associados a um sistema de equações lineares. As saídas dessa função são:

pA : posto da matriz ampliada;

pC : posto da matriz dos coeficientes;

A : matriz ampliada;

S : solução do sistema no caso de única solução;

E : matriz na forma escalonada reduzida equivalente a matriz A .

Exemplo 14.

```
>> C = [3, 2, -5; 2, -4, -2; 4, -8, -4]
```

```
C =
      3      2     -5
      2     -4     -2
      4     -8     -4
```

```
>> B = [8; -4; -8]
```

```
B =
      8
     -4
     -8
```

```
>> [pA, pC, A, S, E] = solucao_sistema_linear(C, B)
```

Sistema possui infinitas solucoes

```
pA =
      2

pC =
      2

A =
      3      2     -5      8
      2     -4     -2     -4
      4     -8     -4     -8
```

```

S =
    [ ]
E =
    1.0000    0   -1.5000    1.5000
         0    1.0000   -0.2500    1.7500
         0     0         0         0

```

Exemplo 15.

```
>> C = [3, 2, -5; 2, -4, -2; 4, -8, -4]
```

```

C =
     3     2    -5
     2    -4    -2
     4    -8    -4

```

```
>> B = [8; -4; -4]
```

```

B =
     8
    -4
    -4

```

```
>> [pA, pC, A, S, E] = solucao_sistema_linear(C, B)
```

Sistema nao possui solucao

```

pA =
     3
pC =
     2
A =
     3     2    -5     8
     2    -4    -2    -4
     4    -8    -4    -4

```

```

S =
    [ ]

```

```

E =
    1.0000    0   -1.5000    0
         0    1.0000   -0.2500    0
         0     0         0    1.0000

```

3.12 Determinate de Uma Matriz

O arquivo “determinante_matriz.m” implementa a função

$$[\text{detA}] = \text{determinante_matriz}(A),$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é uma constante que denota o determinante da matriz A .

Exemplo 16.

```
>> A = [1, 2, 3; 4, 8, 7; 5, 2, 3]
A =
     1     2     3
     4     8     7
     5     2     3
>> [detA] = determinante_matriz(A)
detA =
    -40
```

Exemplo 17.

```
>> A = [1, 2, 3; 4, 8, 7; 5, 2, 3]
A =
     1     2     3
     4     8     7
     5     2     3
>> [detA] = determinante_matriz(A)
A matriz nao e quadrada!!!
```

3.12.1 Escolhendo Uma Linha Para Calcular o Determinante

O arquivo “determinante_matriz_escolhe_linha.m” implementa a função

$$[\text{detA}] = \text{determinante_matriz_escolhe_linha}(A)$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é uma constante que denota o determinante da matriz A . A linha escolhida é solicitada ao usuário durante a execução do código. Essa função permite visualizar de forma detalhada as submatrizes e cofatores associados.

Exemplo 18.

```
>> A = [1, 2, 3; 4, 8, 7; 5, 2, 3]
A =
     1     2     3
     4     8     7
     5     2     3
>> [detA] = determinante_matriz_escolhe_linha(A)
Matriz A:
A =
     1     2     3
     4     8     7
     5     2     3
```

Número de linhas de A:

3

Número de colunas de A:

3

Escolha uma linha qualquer da matriz A: 2

=====

Identifica cada submatriz e cada cofator

=====

Submatriz A_{21} :

2 3

2 3

Determinante da submatriz A_{21} :

0

Cofator Cof_{21} :

-0

=====

Submatriz A_{22} :

1 3

5 3

Determinante da submatriz A_{22} :

-12

Cofator Cof_{22} :

-12

=====

Submatriz A_{23} :

1 2

5 2

Determinante da submatriz A_{23} :

-8

Cofator Cof_{23} :

8

=====

$\det A = -40$

3.12.2 Escolhendo Uma Coluna Para Calcular o Determinante

O arquivo “determinante_matriz_escolhe_coluna.m” implementa a função

$$[\det A] = \text{determinante_matriz_escolhe_linha}(A)$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é uma constante que denota o determinante da matriz A . A coluna escolhida é solicitada ao usuário durante a execução do código. Essa função permite visualizar de forma detalhada as submatrizes e cofatores associados. As informações retornadas por essa função são semelhantes ao que foi apresentado na Seção 3.12.1.

3.13 Matriz dos Cofatores

O arquivo “cofatores_matriz.m” implementa a função

$$[\text{CofA}] = \text{cofatores_matriz}(A)$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é uma matriz de ordem $n \times n$ com o cofator associado a cada elemento de A .

Exemplo 19.

```
>> A = [1, 2, 3; 4, 8, 7; 5, 2, 3]
A =
     1     2     3
     4     8     7
     5     2     3
>> [CofA] = cofatores_matriz(A)
Matriz A:
     1     2     3
     4     8     7
     5     2     3
Número de linhas de A:
3
Número de colunas de A:
3
=====
Submatriz A11:
     8     7
     2     3
Determinante da submatriz A11:
10
Cofator Cof11:
10
=====
Submatriz A12:
     4     7
     5     3
```

Determinante da submatriz A_{12} :

-23

Cofator Cof_{12} :

23

=====

Submatriz A_{13} :

$4 \ 8$

$5 \ 2$

Determinante da submatriz A_{13} :

-32

Cofator Cof_{13} :

-32

=====

Submatriz A_{21} :

$2 \ 3$

$2 \ 3$

Determinante da submatriz A_{21} :

0

Cofator Cof_{21} :

-0

=====

Submatriz A_{22} :

$1 \ 3$

$5 \ 3$

Determinante da submatriz A_{22} :

-12

Cofator Cof_{22} :

-12

=====

Submatriz A_{23} :

$1 \ 2$

$5 \ 2$

Determinante da submatriz A_{23} :

-8

Cofator Cof_{23} :

8

=====

Submatriz A_{31} :

$2 \ 3$

$8 \ 7$

Determinante da submatriz A_{31} :

```

-10
Cofator Cof31:
-10
=====
Submatriz A32:
1 3
4 7
Determinante da submatriz A32:
-5
Cofator Cof32:
5
=====
Submatriz A33:
1 2
4 8
Determinante da submatriz A33:
0
Cofator Cof33:
0
=====
CofA =
      10   23  -32
      -0  -12   8
     -10   5   0

```

3.14 Matriz Adjunta

O arquivo “adjunta_matriz.m” implementa a função

$$[\text{AdjA}] = \text{adjunta_matriz}(A)$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é uma matriz de ordem $n \times n$, a qual corresponde a transposta da matriz dos cofatores associada a matriz A . As informações retornadas por essa função são semelhantes ao que foi apresentado na Seção 3.13 com a etapa adicional de determinar a transposta da matriz dos cofatores.

3.15 Matriz Inversa

O arquivo “inversa_matriz.m” implementa a função

$$[\text{detA}, \text{invA}] = \text{inversa_matriz}(A)$$

cuja a entrada é uma matriz $A = [a_{ij}]_{n \times n}$. As saídas dessa função são:

detA: determinante da matriz A;

invA: matriz de ordem $n \times n$, caso a inversa de A exista;

invA: [] no caso em A não possui inversa.

Exemplo 20.

```
>> A = [2, 3, -2; 1, 4, 8; 7, 5, 3]
A =
     2     3    -2
     1     4     8
     7     5     3
>> [detA, invA] = inversa_matriz(A)
detA = 149
invA =
    -0.187919   -0.127517    0.214765
     0.355705    0.134228   -0.120805
    -0.154362    0.073826    0.033557
```

Exemplo 21.

```
>> A = [2, 3, -2; 4, 6, -4; 7, 5, 3]
A =
     2     3    -2
     4     6   -4
     7     5     3
>> [detA, invA] = inversa_matriz(A)
warning: matrix singular to machine precision, rcond = 6.97854e - 18
warning: called from
    inversa_matriz at line 39 column 10

A matriz não possui inversa!!!
detA = 9.7700e - 15
invA = [ ](0x0)
```

3.16 Polinômio Característico

O arquivo “coeficientes_polinomio_caracteristico.m” implementa a função

$$[c] = \text{coeficientes_polinomio_caracteristico}(A)$$

cuja entrada é uma matriz $A = [a_{ij}]_{n \times n}$. A saída dessa função é um vetor linha $1 \times (n + 1)$ com os coeficientes do polinômio característico associado a matriz A.

Exemplo 22.

```
>> A = [-3, 1, -1; -2, 5, -1; -6, 7, -8]
A =
    -3     1    -1
    -2     5    -1
    -6     7    -8
>> [c] = coeficientes_polinomio_caracteristico(A)
c =
    1.0000    6.0000   -28.0000   -73.0000
```

3.17 Autovalores

O arquivo “autovalores_matriz.m” implementa a função

$$[\text{autoval}] = \text{autovalores_matriz}(A)$$

A entrada dessa função é uma matriz $A = [a_{ij}]_{n \times n}$ cuja saída autoval é um vetor coluna $n \times 1$ com os autovalores de A .

Exemplo 23.

```
>> A = [-3, 1, -1; -7, 5, -1; -6, 6, -2]
autoval =
    4.0000
   -2.0000
   -2.0000
```

3.18 Autovetores Associados a Um Autovalor

O arquivo “autovetores_matriz.m” implementa a função

$$[M_autvet, M_autoval] = \text{autovetores_matriz}(A, \text{autoval})$$

cuja entrada é uma matriz $A = [a_{ij}]_{n \times n}$ e um autovalor autoval de A . O uso dessa função para encontrar a forma geral dos autovalores associados ao autovalor autval tem um caráter mais acadêmico. Essa estrutura permite ao aluno(a) perceber que dado um autovalor de A , obtém-se a forma geral dos autovalores ao observar a forma escalonada reduzida do sistema

$$(A - \text{autoval} \cdot I_n) \cdot v = o \quad (3.1)$$

onde I_n é a matriz identidade de ordem n e “o” é um vetor nulo $n \times 1$. Entretanto, com uso da função `rref()` podem ocorrer erros numéricos. Dependendo da tolerância `tol` estabelecida para `rref()`, tem-se resultados diferentes. Na prática, usa-se a função `eig()` para determinar autovalores e autovetores. Dessa forma, no contexto desse material, tem-se que:

M_autvet : matriz ampliada associada ao sistema (3.1);

$M_autoval$: autovalor de A ;

ou

M_autvet : matriz com autovetores de A ;

$M_autoval$: matriz diagonal com os autovalores de A .

A partir de M_autvet obtém-se a forma geral dos autovetores da matriz A associados ao autovalor $autoval$.

Exemplo 24.

```
>> A = [-3, 1, -1; -7, 5, -1; -6, 6, -2]
A =
    -3     1    -1
    -7     5    -1
    -6     6    -2
>> autoval = -2
autoval = -2
>> [M_autvet, M_autoval] = autovetores_matriz(A, autoval)
M_autvet =
     1    -1     0     0
     0     0     1     0
     0     0     0     0
M_autoval = -2

>> A = [1, 7, 3; 2, 9, 12; 5, 22, 7]
A =
     1     7     3
     2     9    12
     5    22     7
>> autoval = 25.5548
autoval = 25.5548
>> [M_autvet, M_autoval] = autovetores_matriz(A, autoval)
```

Possível erro associado a tolerância de `rref()`!

Nesse caso, será usada a função `eig()`!

Assim, M_autvet é uma matriz de autovetores de A

e $M_autoval$ é uma matriz cuja diagonal contém os autovalores de A !

```

M_autvet =
    -0.260977  -0.973445  0.189104
    -0.587027  0.228061  -0.581573
    -0.766349  -0.019808  0.791210

M_autoval =
    25.55484  0.00000  0.00000
    0.00000  -0.57893  0.00000
    0.00000  0.00000  -7.97590

```

3.19 Verifica Autovalor

O arquivo “[p] = verifica_autovalor_matriz.m” implementa a função

```
[p] = verifica_autovalor_matriz(A, lambda)
```

cujas entradas são uma matriz $A = [a_{ij}]_{n \times n}$ e um escalar lambda. A saída dessa função é um escalar p, o qual representa o valor “p”(lambda) onde $p(\cdot)$ é o polinômio característico de A.

Exemplo 25.

```

>> A = [-3, 1, -1; -7, 5, -1; -6, 6, -2]
A =
    -3     1    -1
    -7     5    -1
    -6     6    -2
>> lambda = -2
lambda = -2
>> [p] = verifica_autovalor_matriz(A, lambda)
p = -7.1054e - 15

```

3.20 Verifica Autovetor

O arquivo “verifica_autovalor_autovetor_matriz.m” implementa a função

```
[O] = verifica_autovalor_autovetor_matriz(A, lambda, v)
```

cujas entradas são uma matriz $A = [a_{ij}]_{n \times n}$, um escalar lambda e um vetor “v” $n \times 1$. A saída dessa função é um vetor O $n \times 1$, o qual representa a forma geral da matriz resultante do produto

$$(A - \text{lambda} \cdot I_n) \cdot v \quad (3.2)$$

No caso em que lambda não é um autovalor de A ou “v” é um vetor nulo, então a saída da função é $O = []$.

Exemplo 26.

```
>> A = [-3, 1, -1; -7, 5, -1; -6, 6, -2]
```

```
A =
```

```
    -3    1    -1
```

```
    -7    5    -1
```

```
    -6    6    -2
```

```
>> lambda = -2
```

```
lambda = -2
```

```
>> v = [-1; 0; 0]
```

```
v =
```

```
    -1
```

```
     0
```

```
     0
```

```
>> [O] = verifica_autovalor_autovetor_matriz(A, lambda, v)
```

```
O =
```

```
     1
```

```
     7
```

```
     6
```

4 Atividades Usando o Octave

Este capítulo apresenta um conjunto de atividades que podem ser realizadas com Octave. Especificamente, usa-se o Octave para verificar a resolução associada a cada questão apresentada. Ao utilizar essas funções é importante ter atenção ao usar informações retornadas na janela de comandos do Octave. Use os comandos “clear all” e “clc” sempre que for necessário limpar informações da janela de comandos.

Em grande parte das aplicações, o conjunto numérico utilizado é o dos números reais \mathbb{R} . Este conjunto possui infinitos números, mas o computador pode armazenar apenas uma quantidade finita destes. Isto ocasiona erros na representação dos números reais, pois um número que não pode ser representado de forma exata no computador, será aproximado por um outro. Temos então os erros de representação numérica. Em um computador, mesmo que inicialmente tenhamos boas representações numéricas para o conjunto dos números reais (representações com erros pequenos), os erros podem se propagar de forma que ao final de uma série de operações, o resultado encontrado não se aproxima em nada do resultado original esperado. Para minizar os erros, o Octave utiliza precisão dupla na representação de números reais. Isto faz com que o número tenha acurácia nos primeiros 15 dígitos. Por padrão o Octave mostra apenas alguns dígitos de um número. Para ver todos os dígitos representados, você deve digitar

```
>> format long
```

e pressionar enter. Os números serão então mostrados com todos seus dígitos. Para retornar ao padrão anterior, basta digitar

```
>> format short
```

Um experimento interessante para ilustrar esta questão dos erros é tentar fazer a soma do número 0,1 oito vezes. Digite no Octave:

```
>> 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1
```

e pressione enter. Caso o Octave esteja mostrando os números no formato curto a resposta será:

```
ans = 0.80000.
```

Neste caso, a resposta está sendo mostrada com cinco casas decimais. Já houve

um erro numérico que o Octave não está mostrando por conta de um arredondamento. Se você fizer a mesma experiência mas no formato longo, a resposta será:

$$ans = 7.999999999999999e - 01.$$

O que mostra que mesmo para uma conta simples, já se observa um pequeno erro numérico associado.

As atividades apresentadas nesse capítulo são baseadas em diferentes referências adotadas na disciplina de Álgebra Linear ([BOLDRINI et al., 1986](#); [KOLMAN](#); [HILL, 2011](#); [STEINBRUCH](#); [WINTERLE, 1987](#); [SANTOS, 2012](#); [LIPSCHUTZ, 1972](#)).

4.1 Atividade 1

Usando o Octave

- Use a função

$$[A] = \text{construa_matriz}(\text{regra_ii}, \text{regra_ij}, m, n)$$

para verificar a resolução de cada item da Questão [4.1.1](#).

- Use a função

$$[A] = \text{construa_matriz_caso_geral}(\text{regra_ii}, \text{regra_ij}, \text{cond_ij}, \text{valor_cond}, m, n)$$

para verificar a resolução de cada item da Questão [4.1.2](#).

- Use as funções

$$[C] = \text{soma_matrizes}(A, B),$$

$$[C] = \text{produto_matrizes}(A, B),$$

e

$$[C] = \text{transposta_matriz}(A).$$

para verificar cada item da Questão [4.1.3](#).

- Use a função

$$[C] = \text{produto_matrizes}(A, B),$$

para verificar a resolução das Questões [4.1.4](#) e [4.1.5](#).

Questão 4.1.1. *Construa as matrizes a seguir:*

$$(a) \ A = [a_{ij}]_{4 \times 7} \text{ definida por } a_{ij} = \begin{cases} 5i - 3j, & \text{se } i = j \\ (i - j)^2, & \text{se } i \neq j \end{cases}$$

(b) $B = [b_{jk}]_{7 \times 9}$ definida por $b_{jk} = j - k$

(c) $C = [c_{ij}]_{4 \times 7}$ definida por $c_{ij} = \begin{cases} i^2, & \text{se } i = j \\ 2.i - j, & \text{se } i \neq j \end{cases}$

Questão 4.1.2. Construa as matrizes a seguir:

(a) $A = [a_{ij}]_{3 \times 5}$ definida por $a_{ij} = \begin{cases} 2.i + 3.j, & \text{se } i + j = 4 \\ (i - j)^2, & \text{se } i + j \neq 4 \end{cases}$

(b) $B = [a_{ij}]_{3 \times 5}$ definida por $a_{ij} = \begin{cases} (i - j)^3, & \text{se } i.j = 2 \\ -3.j + 1, & \text{se } i.j \neq 2 \end{cases}$

(c) $A = [a_{ij}]_{3 \times 5}$ definida por $a_{ij} = \begin{cases} -i + 4.j, & \text{se } i - j = 1 \\ j - 3, & \text{se } i - j \neq 1 \end{cases}$

Questão 4.1.3. Sejam as matrizes $A = \begin{bmatrix} 4 & 11 & -9 \\ 0 & 3 & 2 \\ -3 & 1 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0 & 5 \\ -4 & 6 & 11 \\ -6 & 4 & 9 \end{bmatrix}$, $C =$

$\begin{bmatrix} 4 & -2 & 7 \\ 6 & -4 & 9 \\ 8 & -6 & 5 \end{bmatrix}$ e $D = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ -2 & 1 \end{bmatrix}$. Verifique que:

(a) $(A + B) + C = A + (B + C)$

(b) $(AB)D = A(BD)$

(c) $(B + C)D = BD + CD$

(d) $(A + B)^T = A^T + B^T$

(e) $(AD)^T = D^T A^T$

Questão 4.1.4. Verifique se a matriz $C = \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ é a inversa da matriz $B =$

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

Questão 4.1.5. Verifique se a matriz $B = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 2 & -1 & 2 \\ 0 & -1 & -7 & 1 \\ 0 & 3 & 3 & 0 \end{bmatrix}$ é a inversa da matriz

$$C = \begin{bmatrix} 3 & 2 & 0 & -1 \\ 2 & 4 & -1 & 2 \\ -6 & -1 & 0 & 0 \\ 1 & 3 & 2 & 1 \end{bmatrix}.$$

4.2 Atividade 2

Usando o Octave

- Use a função

$$[E, pA, nA] = \text{forma_escalonada_reduzida_matriz}(A)$$

para verificar a resolução de cada item da Questão [4.2.1](#).

Questão 4.2.1. Determine o posto e a nulidade de cada matriz a seguir:

$$(a) A = \begin{bmatrix} 4 & 11 & -9 \\ 0 & 3 & 2 \\ -3 & 1 & 1 \end{bmatrix}$$

$$(b) B = \begin{bmatrix} 1 & 0 & 5 \\ -4 & 6 & 11 \\ -6 & 4 & 9 \end{bmatrix},$$

$$(c) C = \begin{bmatrix} 4 & -2 & 7 & 5 \\ 6 & -4 & 9 & 1 \\ 8 & -6 & 5 & -3 \end{bmatrix}$$

$$(d) D = \begin{bmatrix} 3 & 2 & 0 & -1 \\ 2 & 4 & -1 & 2 \\ -2 & -6 & -4 & -2 \\ -6 & -1 & 0 & 0 \\ 1 & 3 & 2 & 1 \end{bmatrix}.$$

4.3 Atividade 3

Usando o Octave

- Use a função

$$[R, B] = \text{verifica_solucao_sistema_linear}(C, B, X).$$

para verificar a resolução da Questão 4.3.1.

Questão 4.3.1. Verifique se $X = \begin{bmatrix} -\frac{3}{2} \\ \frac{5}{4} \\ -2 \end{bmatrix}$ é solução do sistema linear

$$\begin{cases} 3x + 2y - 5z = 8 \\ 2x - 4y - 2z = -4 \\ 4x - 8y - 4z = -8 \end{cases}$$

4.4 Atividade 4

Usando o Octave

- Use a função

$$[pA, pC, A, S, E] = \text{solucao_sistema_linear}(C, B).$$

para verificar a resolução das Questões 4.4.1 e 4.4.2.

Resolva as Questões 4.4.1 e 4.4.2 usando o método de Gauss-Jordan.

Questão 4.4.1. Determine se cada sistema linear homogêneo a seguir tem solução não nula.

$$(a) \begin{cases} x - 2y + 2z = 0 \\ 2x + y - 2z = 0 \\ 3x + 4y - 6z = 0 \\ 3x - 11y + 12z = 0 \end{cases}$$

$$(b) \begin{cases} 2x - 4y + 7z + 4v - 5w = 0 \\ 9x + 3y + 2z - 7v + w = 0 \\ 5x + 2y - 3z + v + 3w = 0 \\ 6x - 5y + 4z - 3v - 2w = 0 \end{cases}$$

Questão 4.4.2. Resolva o sistema

$$\begin{cases} x + 2y - 3z = 4 \\ x + 3y + z = 11 \\ 2x + 5y - 4z = 13 \\ 2x + 6y + 2z = 22 \end{cases}$$

4.5 Atividade 5

Usando o Octave

- Use a Janela de Comandos do Octave para verificar a resolução das Questões 4.5.1 e 4.5.2. Veja como exemplo a representação das operações elementares a seguir na Janela de Comandos.

$$A = \left[\begin{array}{ccc|c} 3 & 2 & -5 & 8 \\ 2 & -4 & -2 & -4 \\ 1 & -2 & -3 & -4 \end{array} \right] \quad L_2 \rightarrow 3L_2 - 2L_1 \quad \left[\begin{array}{ccc|c} 3 & 2 & -5 & 8 \\ 0 & -16 & 4 & -28 \\ 1 & -2 & -3 & -4 \end{array} \right] \quad L_3 \rightarrow 3L_3 - L_1$$

$$\left[\begin{array}{ccc|c} 3 & 2 & -5 & 8 \\ 0 & -16 & 4 & -28 \\ 0 & -8 & -4 & -20 \end{array} \right] \quad L_3 \rightarrow (-16)L_3 - (-8)L_2 \quad \left[\begin{array}{ccc|c} 3 & 2 & -5 & 8 \\ 0 & -16 & 4 & -28 \\ 0 & 0 & 96 & 96 \end{array} \right]$$

```
>> A = [3, 2, -5, 8; 2, -4, -2, -4; 1, -2, -3, -4]
```

```
A =
```

```

3   2   -5   8
2  -4   -2  -4
1  -2   -3  -4
```

```
>> A(2,:) = 3 * A(2,:) - 2 * A(1,:)
```

```
A =
```

```

3   2   -5   8
0 -16    4 -28
1  -2   -3  -4
```

```
>> A(3,:) = 3 * A(3,:) - A(1,:)
```

```
A =
```

```

3   2   -5   8
0 -16    4 -28
0  -8   -4 -20
```

```
>> A(3,:) = (-16) * A(3,:) - (-8)A(2,:)
```

```
A =
```

```

3   2   -5   8
0 -16    4 -28
0   0   96  96
```

Resolva as Questões 4.5.1 e 4.5.2 usando o método de Gauss.

Questão 4.5.1. *Determine se cada sistema linear homogêneo a seguir tem solução não nula.*

$$(a) \begin{cases} x - 2y + 2z = 0 \\ 2x + y - 2z = 0 \\ 3x + 4y - 6z = 0 \\ 3x - 11y + 12z = 0 \end{cases}$$

$$(a) \begin{cases} 2x - 4y + 7z + 4v - 5w = 0 \\ 9x + 3y + 2z - 7v + w = 0 \\ 5x + 2y - 3z + v + 3w = 0 \\ 6x - 5y + 4z - 3v - 2w = 0 \end{cases}$$

Questão 4.5.2. *Resolva o sistema*

$$\begin{cases} x + 2y - 3z = 4 \\ x + 3y + z = 11 \\ 2x + 5y - 4z = 13 \\ 2x + 6y + 2z = 22 \end{cases}$$

4.6 Atividade 6

Usando o Octave

- Use a função

$$[\text{detA}] = \text{determinante_matriz_escolhe_coluna}(\text{A})$$

para verificar a resolução da Questão 4.6.1.

Questão 4.6.1. *Calcule o determinante da matriz $A =$*

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

4.7 Atividade 7

Usando o Octave

- Use a função

$$[\text{detA}] = \text{determinante_matriz_escolhe_linha}(\text{A})$$

para verificar a resolução da Questão 4.7.1.

Questão 4.7.1. Calcule o determinante da matriz $A = \begin{bmatrix} 2 & -9 & 3 & -2 \\ -1 & 4 & 6 & -4 \\ 0 & 7 & 5 & 3 \\ -3 & 0 & -4 & 1 \end{bmatrix}$.

4.8 Atividade 8

Usando o Octave

- Use as funções

$$[\text{Cof}A] = \text{cofatores_matriz}(A)$$

$$[\text{Adj}A] = \text{adjunta_matriz}(A)$$

$$[\text{det}A] = \text{determinante_matriz_escolhe_linha}(A)$$

$$[C] = \text{produto_matriz_por_escalar}(A, k)$$

para verificar a resolução da Questão 4.8.1.

Questão 4.8.1. Encontre a matriz inversa de $A = \begin{bmatrix} 4 & 1 & 2 & 3 \\ 5 & 7 & 9 & -2 \\ 0 & -1 & 3 & 4 \\ 0 & -3 & 5 & 8 \end{bmatrix}$.

4.9 Atividade 9

Usando o Octave

- Use a função

$$[\text{det}A, \text{inv}A] = \text{inversa_matriz}(A)$$

para verificar a resolução da Questão 4.9.1.

Questão 4.9.1. Encontre a matriz inversa de $A = \begin{bmatrix} 4 & 1 & 2 & 3 \\ 5 & 7 & 9 & -2 \\ 0 & -1 & 3 & 4 \\ 0 & -3 & 5 & 8 \end{bmatrix}$.

4.10 Atividade 10

Usando o Octave

- Use a função

$$[c] = \text{coeficientes_polinomio_caracteristico}(A)$$

para verificar a resolução da Questão 4.10.1.

Questão 4.10.1. Encontre o polinômio característico de $A = \begin{bmatrix} 2 & 3 & 0 & 5 & 5 \\ 4 & 5 & 0 & 3 & 2 \\ 4 & 2 & 1 & 1 & 5 \\ 1 & 3 & -5 & 1 & 2 \\ 1 & 2 & 0 & 2 & 5 \end{bmatrix}$.

4.11 Atividade 11

Usando o Octave

- Use as funções

$$[c] = \text{coeficientes_polinomio_caracteristico}(A)$$

$$[\text{autoval}] = \text{autovalores_matriz}(A)$$

$$[O] = \text{verifica_autovalor_autovetor_matriz}(A, \text{lambda}, v)$$

para resolver a Questão 4.11.1.

Questão 4.11.1. Considere a matriz $A = \begin{bmatrix} -9 & 3 & -2 \\ 7 & 5 & 3 \\ 0 & -4 & 0 \end{bmatrix}$.

(a) Encontre o polinômio característico de A .

(b) Encontre os autovalores de A .

(c) Verifique se o vetor $v = [0.73254; 0.51783; 0.54120]$ é um autovetor de A associado ao autovalor $\text{lambda} = \text{autoval}(1)$.

Para resolver o item (c) digite na janela de comandos os seguintes passos:

```
>> A = [-9, 3, -2; 7, 5, 3; 0, -4, 0]
```

```
>> [autoval] = autovalores_matriz(A)
```

```
>> v = [0.73254; 0.51783; 0.54120]
```

```
>> lambda = autoval(1)
```

```
>> [O] = verifica_autovalor_autovetor_matriz(A, lambda, v)
```

4.12 Atividade 12

Usando o Octave

- Use a função

$$[M_autvet, M_autoval] = \text{autovetores_matriz}(A, \text{autoval})$$

para verificar a resolução da Questão [4.12.1](#).

Questão 4.12.1. *Encontre os autovetores de $A = \begin{bmatrix} -1 & 2 & 0 \\ 5 & 6 & 0 \\ -8 & 0 & 7 \end{bmatrix}$ associados ao autovalor $\lambda = 7$.*

5 Considerações Finais

O projeto “Elaboração de Material Didático que Empregue o uso de Software como Suporte para o Aprendizado de Álgebra Linear” resultou no desenvolvimento de um material didático acessível à docentes e discentes que não possuem nenhum conhecimento prévio com o software Octave. Trata-se também de um material diferenciado que pode complementar o ensino-aprendizado da disciplina de Álgebra Linear.

O uso desse material em sala de aula proporciona aos discentes o contato com uma ferramenta computacional, a qual permite explorar diferentes conceitos de Álgebra Linear. Pode-se considerar também que o presente material representa uma oportunidade do docente ter uma experiência em adotar um material que pode tornar a disciplina mais atrativa. Além disso, as diferentes atividades apresentas no material proporcionam uma maior flexibilidade com relação às avaliações a serem aplicadas na disciplina.

Este material é um primeiro passo dos membros deste projeto em direção ao uso de recursos de ensino que estimulem o interesse dos discentes. A partir dessa experiência nesse projeto, espera-se em um futuro próximo desenvolver um outro material didático que explore a resolução de problemas práticos por meio do Octave no contexto de Álgebra Linear.

Referências

- ANTON, H.; RORRES, C. *Álgebra Linear com Aplicações*. 8. ed. [S.l.]: Bookman, 2001. Tradução de Clauss Ivo Doering. Citado na página 4.
- BOLDRINI, J. L. et al. *Álgebra Linear*. [S.l.]: Harbra, 1986. Citado na página 31.
- EATON DAVID BATEMAN, S. H. R. W. J. W. *GNU Octave: A high-level interactive language for numerical computations*. Version 5.2.0. [S.l.], 2020. Disponível em: <<https://octave.org/octave.pdf>>. Citado 2 vezes nas páginas 4 e 5.
- GNU Octave. Scientific Programming Language. 2020. Acessado em: 11 de maio de 2020. Disponível em: <<https://www.gnu.org/software/octave/>>. Citado na página 5.
- KOLMAN, B.; HILL, D. R. *Introdução à Álgebra Linear com Aplicações*. 8. ed. Rio de Janeiro: LTC, 2011. Tradução de Alessandra Bosquilha. Citado na página 31.
- LAY, D. C. *Álgebra Linear e suas Aplicações*. 2. ed. Rio de Janeiro: LTC, 2007. Citado na página 4.
- LIPSCHUTZ, S. *Álgebra Linear*. 2. ed. [S.l.]: McGraw-Hill do Brasil Ltda, 1972. (Coleção Schaum). Tradução de Roberto Ribeiro Baldino. Citado na página 31.
- POOLE, D. *Álgebra Linear*. [S.l.]: Cengage Learning, 2011. Tradução técnica de Martha Salermo Monteiro (coord). [et al]. Citado na página 4.
- PROAE. Programa de Apoio ao Ensino de Graduação. 2020. Acessado em: 12 de maio de 2020. Disponível em: <<http://www.ufvjm.edu.br/prograd/proae.html>>. Citado na página 4.
- PROJETO Pedagógico do Curso de Graduação em Ciência e Tecnologia. 2020. Acessado em: 12 de maio de 2020. Disponível em: <<http://www.ufvjm.edu.br/prograd/projetos-pedagogicos.html>>. Citado na página 4.
- SANTOS, R. J. *Um Curso de Geometria Analítica e Álgebra Linear*. [S.l.]: Imprensa Universitária da UFMG, 2012. Citado na página 31.
- STEINBRUCH, A.; WINTERLE, P. *Álgebra Linear*. 2. ed. São Paulo: Pearson, 1987. Citado na página 31.

Apêndices

APÊNDICE A – Código das Funções Implementadas

```

1  function [A] = construa_matriz(regra_ii,regra_ij,m,n)
2  % CONSTRUA_MATRIZ: Função que retorna a matriz A de ordem mxn
3  % correspondente a definição fornecida pelas entradas da função.
4  % Entrada:
5  %   regra_ii: Regra associada a condição  $i=j$ , sendo que  $i,j$  representam,
6  %   respectivamente, linha e coluna de A. Essa regra deve ser inserida na
7  %   forma de @(i,j)().
8  %   regra_ij: Regra associada a condição  $i \neq j$ , sendo que  $i,j$  representam,
9  %   respectivamente, linha e coluna de A. Essa regra deve ser inserida na
10 %   forma de @(i,j)().
11 %   m: Número de linhas de A.
12 %   n: Número de colunas de A.
13 % Saída:
14 %   A: Matriz A de ordem mxn
15 % Exemplo:
16 %   Duas condições para os índices
17 %   [A] = construa_matriz(@(i,j)(i - j),@(i,j)(i),4,7)
18 %   Apenas uma condição para os índices
19 %   [A] = construa_matriz(@(i,j)(i),@(i,j)(i),4,7)
20 %=====
21 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
22 %   Software como Suporte para o Aprendizado de Álgebra Linear.
23 % Membros do Projeto:
24 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
25 %   Douglas Frederico Guimarães Santiago (Vice-Coordenador)
26 %   Anderson Luiz Pedrosa Porto (Colaborador)
27 %   Flaviano Luiz Benfica (Bolsista)
28 %=====
29
30 % Construa a matriz A
31 for i = 1:m
32     for j = 1:n
33         if i == j
34             A(i,j) = regra_ii(i,j);
35         else
36             A(i,j) = regra_ij(i,j);
37         end
38     end
39 end
40
41
42 end

```

```

1  function [A] = construa_matriz_caso_geral(regra_ii, regra_ij, cond_ij, valor_cond, m, n)
2  % CONSTRUA MATRIZ: Função que retorna a matriz A de ordem mxn
3  % correspondente a definição fornecida pelas entradas da função.
4  % Entrada:
5  %   regra_ii: Regra associada a condição cond_ij = cond_valor, sendo que i,j
6  %   representam, %   respectivamente, linha e coluna de A. Essa regra deve ser
7  %   inserida na forma de @(i,j)().
8  %   regra_ij: Regra associada a condição cond_ij = valor_cond, sendo que i,j
9  %   representam, respectivamente, linha e coluna de A. Essa regra deve ser
10 %   inserida na forma de @(i,j)().
11 %   cond_ij: Condição estabelecida aos índices i,j da matriz. Essa regra deve
12 %   ser inserida na forma de @(i,j)().
13 %   valor_cond: Valor associado a cond_ij.
14 %   m: Número de linhas de A.
15 %   n: Número de colunas de A.
16 % Saída:
17 %   A: Matriz A de ordem mxn
18 % Exemplo:
19 %   Duas condições para os índices
20 %   [A] = construa_matriz_caso_geral(@(i,j) (2*i), @(i,j) (j+2), @(i,j) (i+j), 4, 4, 7)
21 %   Apenas uma condição para os índices
22 %   [A] = construa_matriz_caso_geral(@(i,j) (2*i), @(i,j) (2*i), @(i,j) (i+j), 4, 4, 7)
23 %=====
24 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
25 %   Software como Suporte para o Aprendizado de Álgebra Linear.
26 % Membros do Projeto:
27 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
28 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
29 %   Anderson Luiz Pedrosa Porto (Colaborador)
30 %   Flaviano Luiz Benfica (Bolsista)
31 %=====
32
33 % Construa a matriz A
34 for i = 1:m
35     for j = 1:n
36         if cond_ij(i,j) == valor_cond
37             A(i,j) = regra_ii(i,j);
38         else
39             A(i,j) = regra_ij(i,j);
40         end
41     end
42 end
43
44
45 end

```

```

1  function [C] = soma_matrizes(A,B)
2  % SOMA_MATRIZES: Função que retorna a matriz C = A + B.
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem mxn.
6  % Saída:
7  % C: Matriz C = A+B de ordem mxn.
8  % Exemplo:
9  % [C] = soma_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 % Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 % Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 % Anderson Luiz Pedrosa Porto (Colaborador)
17 % Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m_A,n_A] = size(A);
22
23 % Calcula a ordem da matriz B
24 [m_B,n_B] = size(B);
25
26 % Verifica se a soma está definida
27 if m_A == m_B && n_A == n_B
28     % Calcula a matriz C
29     C = A + B;
30 else
31     fprintf('Soma de matrizes não está definida. Verifique as ordens das matrizes!!!\n')
32     C = [];
33 end
34
35 end

```

```

1  function [C] = subtracao_matrizes(A,B)
2  % SOMA_MATRIZES: Função que retorna a matriz C = A - B.
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem mxn.
6  % Saída:
7  % C: Matriz C = A+B de ordem mxn.
8  % Exemplo:
9  % [C] = subtracao_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 % Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 % Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 % Anderson Luiz Pedrosa Porto (Colaborador)
17 % Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m_A,n_A] = size(A);
22
23 % Calcula a ordem da matriz B
24 [m_B,n_B] = size(B);
25
26 % Verifica se a soma está definida
27 if m_A == m_B && n_A == n_B
28     % Calcula a matriz C
29     C = A - B;
30 else
31     fprintf('Subtração de matrizes não está definida. Verifique as ordens das
32             matrizes!!!\n')
33     C = [];
34 end
35 end

```

```

1  function [C] = produto_matrizes(A,B)
2  % PRODUTO_MATRIZES: Função que retorna a matriz C = A.B
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem nxp.
6  % Saída:
7  % C: Matriz de ordem nxp.
8  % Exemplo:
9  % [C] = produto_matrizes([1,2,3;4,5,6],[7,7,9,10;11,12,13,14;1,2,3,4])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20
21 % Calcula a ordem da matriz A
22 [m_A,n_A] = size(A);
23
24 % Calcula a ordem da matriz B
25 [m_B,n_B] = size(B);
26
27 % Verifica se o produto matricial está definido
28 if n_A == m_B
29     % Calcula a matriz C
30     C = A*B;
31 else
32     fprintf('Produto de matrizes não está definido. Verifique as ordens as matrizes!!!\n')
33     C = [];
34 end
35
36 end

```



```
1  function[C] = produto_matriz_por_escalar(A,k)
2  % PRODUTO_MATRZ_POR_ESCALAR: Função que retorna a matriz C = k.A.
3  % Entrada:
4  %   A: Matriz A de ordem mxn.
5  %   k: Escalar qualquer
6  % Saída:
7  %   C: Matriz C = k.A de ordem mxn.
8  % Exemplo:
9  %   [C] = produto_matriz_por_escalar([1,2,3;4,5,6],-3)
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 %   Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a matriz C
21 C = k.*A;
22
23 end
```

```
1  function[C] = transposta_matriz(A)
2  % TRANSPOSTA_MATRIZ: Função que retorna a matriz matriz transposta de A.
3  % Entrada:
4  %   A: Matriz A de ordem mxn.
5  % Saída:
6  %   C: Matriz C = A^{T} de ordem nxm.
7  % Exemplo:
8  %   [C] = transposta_matriz([1,2,3;4,5,6])
9  %=====
10 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
11 %   Software como Suporte para o Aprendizado de Álgebra Linear.
12 % Membros do Projeto:
13 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
14 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
15 %   Anderson Luiz Pedrosa Porto (Colaborador)
16 %   Flaviano Luiz Benfica (Bolsista)
17 %=====
18
19 % Calcula a matriz C
20 C = A';
21
22 end
```

```

1  function [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
2  % FORMA_ESCALONADA_REDUIZIDA: Função que retorna a forma escalonada reduzida
3  % da matriz A.
4  % Entrada:
5  %   A: Matriz de ordem mxn.
6  % Saída:
7  %   E: Matriz na forma escalonada reduzida equivalente a A, no caso de não
8  %       ocorrer erro ao usar a função rref().
9  %   E: [] no caso de ocorrer erro ao usar a função rref().
10 %   pA: Posto de A.
11 %   nA: Nulidade de A.
12 % Exemplo:
13 %   [E,pA,nA] = forma_escalonada_reduzida_matriz([1,2,-3,0;2,4,-2,2;3,6,-4,3])
14 %=====
15 %Importante: O uso dessa função para encontrar a forma escalonada reduzida em um
16 % caráter mais acadêmico. Com o uso da função rref() podem ocorrer erros numéricos.
17 % Dependendo da tolerância tol estabelecida para rref(), tem-se resultados
18 % diferentes.
19 %=====
20 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
21 % Software como Suporte para o Aprendizado de Álgebra Linear.
22 % Membros do Projeto:
23 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
24 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
25 %   Anderson Luiz Pedrosa Porto (Colaborador)
26 %   Flaviano Luiz Benfica (Bolsista)
27 %=====
28
29 % Determina o número de linhas e de colunas de A
30 [m,n] = size(A);
31
32 % Determina o posto de A
33 pA = rank(A);
34
35 % Determina a nulidade de A
36 nA = n - pA;
37
38 % Determina a forma escalonada reduzida equivalente a A.
39 E = rref(A);
40 % Verifica se pode ter ocorrido erro com o uso de rref
41 m_absE = rank(E);
42 % Verifica se o número de linhas de absE é diferente de pA
43 if m_absE ~= pA
44     fprintf('Possivel erro associado a tolerancia de rref()!\n');
45     E = [];
46 end
47
48 end

```

```

1  function [A,E] = escalonada_reduzida_matriz_ampliada(C,B)
2  % ESCALONADA_REDUZIDA_MATRIZ_AMPLIADA: Função que Retorna a matriz ampliada
3  % e a forma escalonada reduzida associada.
4  % Entrada:
5  %   C: Matriz dos coeficientes de ordem mxn.
6  %   B: Matriz dos termos independentes de ordem mx1.
7  % Saída:
8  %   A: Matriz ampliada de ordem mx(n+1).
9  %   E: Matriz na forma escalonada reduzida equivalente a A.
10 % Exemplo:
11 %   [A,E] = escalonada_reduzida_matriz_ampliada([3,2,-5;2,-4,-2;1,-2,-3],[8;-4;-4])
12 %=====
13 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
14 %   Software como Suporte para o Aprendizado de Álgebra Linear.
15 % Membros do Projeto:
16 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
17 %   Douglas Frederico Guimarães Santiago (Vice-Cordenador)
18 %   Anderson Luiz Pedrosa Porto (Colaborador)
19 %   Flaviano Luiz Benfica (Bolsista)
20 %=====
21
22 % Calcula a ordem da matriz C
23 [m_C,n_C] = size(C);
24
25 % Calcula a ordem da matriz I
26 [m_B,n_B] = size(B);
27
28 % Verifica se o número de colunas de C é igual ao número de linhas de B
29 if m_C == m_B
30     % Identifica a matriz ampliada associada ao sistema linear
31     A = [C,B];
32     % Determina o posto de A
33     pA = rank(A);
34     % Encontra a forma escalonada reduzida associada a matriz A
35     E = rref(A);
36     % Verifica se pode ter ocorrido erro com o uso de rref
37     m_absE = rank(E);
38     % Verifica se o número de linhas de absE é diferente de pA
39     if m_absE ~= pA
40         fprintf('Possível erro associado a tolerância de rref()!\n');
41         E = [];
42     end
43 else
44     fprintf('Verifique as ordens as matrizes!!!')
45     A = [];
46     E = [];
47 end

```

```

1  function [R,B] = verifica_solucão_sistema_linear(C,B,X)
2  % VERIFICA_SOLUCAO_SISTEMA_LINEAR: Função que verifica se o S é solução
3  % do sistema linear representado por  $AX = B$ 
4  % solução. Retorna a solução em caso afirmativo para sistema possível e
5  % determinado. Retorna a matriz na forma escalonada reduzida associada ao
6  % sistema linear em questão.
7  % Entrada:
8  %   C: Matriz dos coeficientes nxn.
9  %   B: Matriz dos termos independentes nx1.
10 %   X: Matriz nx1
11 % Saída:
12 %   R: Matriz resultante  $R = A.X$ .
13 %   B: Matriz dos termos independentes nx1.
14 %=====
15 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
16 % Software como Suporte para o Aprendizado de Álgebra Linear.
17 % Membros do Projeto:
18 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
19 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
20 %   Anderson Luiz Pedrosa Porto (Colaborador)
21 %   Flaviano Luiz Benfica (Bolsista)
22 %=====
23 % Exemplo:
24 % [R,B] = verifica_solucão_sistema_linear([3 2 -5;2 -4 -2;1 -2 -3],[8;-4;-4],[-2;3;1])
25
26 % Calcula a ordem da matriz C
27 [m_C,n_C] = size(C);
28
29 % Calcula a ordem da matriz X
30 [m_X,n_X] = size(X);
31
32 % Verifica se o produto AX está definido
33 if n_C == m_X
34     R = C*X;
35 else
36     fprintf('Verifique as ordens das matrizes\n');
37     R = [];
38
39 end
40

```

```

1  function[pA, pC, A, S, E] = solucao_sistema_linear(C,B)
2  % SOLUCAO_SISTEMA_LINEAR: Função que verifica se o sistema linear possui
3  % solução. Retorna a solução em caso afirmativo para sistema possível e
4  % determinado. Retorna a matriz na forma escalonada reduzida associada ao
5  % sistema linear em questão.
6  % Entrada:
7  %   C: Matriz dos coeficientes mxn.
8  %   B: Matriz dos termos independentes mx1.
9  % Saída:
10 %   pA: Posto da matriz ampliada A .
11 %   pC: Posto da matriz dos coeficientes C.
12 %   A: Matriz ampliada de ordem mx(n+1).
13 %   E:
14 %       Matriz de ordem mx(nx1) na forma escalonada reduzida equivalente a A
15 %       [] se ocorre erro associado a rref().
16 %   S:
17 %       matriz nx1 se Sistema Possível e Determinado.
18 %       [] se Sistema Impossível.
19 %       [] se Sistema Possível e Indeterminado.
20 %       [] se ocorre erro associado a rref().
21 %=====
22 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
23 % Software como Suporte para o Aprendizado de Álgebra Linear.
24 % Membros do Projeto:
25 % Mônica Aparecida Cruvinel Valadão (Coordenadora)
26 % Douglas Frederico Guimarães Santiago (Vice-Coordenador)
27 % Anderson Luiz Pedrosa Porto (Colaborador)
28 % Flaviano Luiz Benfica (Bolsista)
29 %=====
30 % Exemplo:
31 % [pA, pC, A, S, E] = solucao_sistema_linear([3 2 -5;2 -4 -2;1 -2 -3],[8;-4;-4])
32 % Calcula a ordem da matriz C
33 [m_C,n_C] = size(C);
34 % Calcula a ordem da matriz B
35 [m_B,n_B] = size(B);
36 % Verifica se o número de linhas de C é igual ao número de linhas de B
37 if m_C == m_B
38     A = [C,B];
39     % Determina o posto de A
40     pA = rank(A);
41     % Determina o posto de C
42     pC = rank(C);
43     % Identifica o número de variáveis
44     n = n_C;
45     % Verifica se o sistema possui solução
46     if pA == pC
47         if pA == n
48             fprintf('Sistema possui única solucao\n');
49             S = linsolve(C,B);
50         else
51             fprintf('Sistema possui infinitas soluções\n');
52             S = [];
53         end
54     else
55         fprintf('Sistema não possui solução\n');
56         S = [];
57     end
58     % Encontra a forma escalonada reduzida associada a matriz A
59     E = rref(A);
60     % Verifica se pode ter ocorrido erro com o uso de rref
61     m_absE = rank(E);
62     if m_absE ~= pA
63         fprintf('Possível erro associado a tolerância de rref()!\n');
64         E = [];
65     end
66 else
67     fprintf('Verifique as ordens das matrizes!!!\n');
68     pA = [];
69     pC = [];
70     A = [];
71     E = [];
72     S = [];
73 end
74 end

```

```
1  function [detA] = determinante_matriz(A)
2  % DETERMINANTE_MATRIZ: Função que calcula o determinante da matriz A.
3  % Entrada:
4  %   A: Matriz A de ordem nxn.
5  % Saída:
6  %   detA: Determinante da matriz A.
7  % Exemplo:
8  %   [detA] = determinante_matriz([1,2,3;4 8 7;5 2 3])
9  %=====
10 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
11 %   Software como Suporte para o Aprendizado de Álgebra Linear.
12 % Membros do Projeto:
13 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
14 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
15 %   Anderson Luiz Pedrosa Porto (Colaborador)
16 %   Flaviano Luiz Benfica (Bolsista)
17 %=====
18
19 % Calcula a ordem da matriz A
20 [m,n] = size(A);
21
22 % Verifica a condição para calcular o determinante
23 if m == n
24     detA = det(A);
25 else
26     fprintf('A matriz não é quadrada!!!\n');
27     detA = [];
28 end
29
30
31
32 end
```

```
1  function [detA] = determinante_matriz(A)
2  % DETERMINANTE_MATRIZ: Função que calcula o determinante da matriz A.
3  % Entrada:
4  %   A: Matriz A de ordem nxn.
5  % Saída:
6  %   detA: Determinante da matriz A.
7  % Exemplo:
8  %   [detA] = determinante_matriz([1,2,3;4 8 7;5 2 3])
9  %=====
10 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
11 %   Software como Suporte para o Aprendizado de Álgebra Linear.
12 % Membros do Projeto:
13 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
14 %   Douglas Frederico Guimarães Santiago (Vice-Coordenador)
15 %   Anderson Luiz Pedrosa Porto (Colaborador)
16 %   Flaviano Luiz Benfica (Bolsista)
17 %=====
18
19 % Calcula a ordem da matriz A
20 [m,n] = size(A);
21
22 % Verifica a condição para calcular o determinante
23 if m == n
24     detA = det(A);
25 else
26     fprintf('A matriz não é quadrada!!!\n');
27     detA = [];
28 end
29
30
31
32 end
```



```

1  function [detA] = determinante_matriz_escolhe_coluna(A)
2  % DETERMINANTE_MATRIZ_ESCOLHE_COLUNA: Função que calcula o determinante
3  % da matriz A escolhendo uma coluna qualquer de A.
4  % Entrada:
5  %   A: Matriz A de ordem mxm.
6  % Saída:
7  %   detA: Determinante da matriz A.
8  % Exemplo:
9  %   [detA] = determinante_matriz_escolhe_coluna([1,2,3;4 8 7;5 2 3])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m,n] = size(A);
22 % Imprime a matriz A
23 fprintf('\n Matriz A:\n');
24 disp(A)
25 % Imprime o número de linhas de A
26 fprintf('\n Número de linhas de A: \n');
27 disp(m)
28 % Imprime o número de colunas de A
29 fprintf('\n Número de colunas de A: \n');
30 disp(n)
31 % Verifica a condição para calcular o determinante (se A é matriz quadrada)
32 if m == n
33     nL = [1:m];
34     Lna = input('Escolha uma coluna qualquer da matriz A: ','s');
35     L = str2num(Lna);
36     vL = find(nL == L);
37     if isempty(vL)
38         fprintf('\n Índice de coluna incorreto \n');
39         detA = [];
40     else
41         fprintf('===== \n');
42         fprintf('Identifica cada submatriz e cada cofator \n');
43         fprintf('===== \n');
44         Dtnante = 0;
45         for j = 1:n
46             k = strcat(num2str(j),num2str(L));
47             fprintf('Submatriz A_%s:\n\n',k);
48             S = A;
49             S(:,L) = [];
50             S(j,:) = [];
51             disp(S)
52             fprintf('Determinante da submatriz A_%s:\n\n',k);
53             detS = det(S);
54             disp(detS)
55             fprintf('Cofator Cof_%s:\n\n',k);
56             cof = (-1)^(j+L)*detS;
57             disp(cof)
58             Dtnante = Dtnante + A(j,L)*cof;
59             fprintf('===== \n');
60         end
61         detA = Dtnante;
62         %   detA = det(A);
63         %   % Compara Dtnante com detA
64         %   if Dtnante ~= detA
65         %       fprintf('\n Possível erro numérico \n');
66         %   end
67     end
68 else
69     fprintf('A matriz não é quadrada!!!\n');
70     detA = [];
71 end
72
73 end

```

```

1  function [detA] = determinante_matriz_escolhe_linha(A)
2  % DETERMINANTE_MATRIZ_ESCOLHE_LINHA: Função que calcula o determinante
3  % da matriz A escolhendo uma linha qualquer de A.
4  % Entrada:
5  %   A: Matriz A de ordem mxm.
6  % Saída:
7  %   detA: Determinante da matriz A.
8  % Exemplo:
9  %   [detA] = determinante_matriz_escolhe_linha([1,2,3;4 8 7;5 2 3])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m,n] = size(A);
22 % Imprime a matriz A
23 fprintf('\n Matriz A:\n');
24 disp(A)
25 % Imprime o número de linhas de A
26 fprintf('\n Número de linhas de A: \n');
27 disp(m)
28 % Imprime o número de colunas de A
29 fprintf('\n Número de colunas de A: \n');
30 disp(n)
31 % Verifica a condição para calcular o determinante (se A é matriz quadrada)
32 if m == n
33     nL = [1:m];
34     Lna = input('Escolha uma linha qualquer da matriz A: ', 's');
35     L = str2num(Lna);
36     vL = find(nL == L);
37     if isempty(vL)
38         fprintf('\n Índice de linha incorreto \n');
39         detA = [];
40     else
41         fprintf('===== \n');
42         fprintf('Identifica cada submatriz e cada cofator \n');
43         fprintf('===== \n');
44         Dtnante = 0;
45         for j = 1:n
46             k = strcat(num2str(L), num2str(j));
47             fprintf('Submatriz A_%s: \n \n', k);
48             S = A;
49             S(L,:) = [];
50             S(:,j) = [];
51             disp(S)
52             fprintf('Determinante da submatriz A_%s: \n \n', k);
53             detS = det(S);
54             disp(detS)
55             fprintf('Cofator Cof_%s: \n \n', k);
56             cof = (-1)^(L+j)*detS;
57             disp(cof)
58             Dtnante = Dtnante + A(L,j)*cof;
59             fprintf('===== \n');
60         end
61         detA = Dtnante;
62         %   detA = det(A);
63         %   % Compara Dtnante com detA
64         %   if Dtnante ~= detA
65         %       fprintf('\n Possível erro numérico \n');
66         %   end
67     end
68 else
69     fprintf('A matriz não é quadrada!!! \n');
70     detA = [];
71 end
72
73 end

```

```

1  function [A,E] = escalonada_reduzida_matriz_ampliada(C,B)
2  % ESCALONADA_REDUZIDA_MATRIZ_AMPLIADA: Função que Retorna a matriz ampliada
3  % e a forma escalonada reduzida associada.
4  % Entrada:
5  %   C: Matriz dos coeficientes de ordem mxn.
6  %   B: Matriz dos termos independentes de ordem mx1.
7  % Saída:
8  %   A: Matriz ampliada de ordem mx(n+1).
9  %   E: Matriz na forma escalonada reduzida equivalente a A.
10 % Exemplo:
11 %   [A,E] = escalonada_reduzida_matriz_ampliada([3,2,-5;2,-4,-2;1,-2,-3],[8;-4;-4])
12 %=====
13 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
14 % Software como Suporte para o Aprendizado de Álgebra Linear.
15 % Membros do Projeto:
16 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
17 %   Douglas Frederico Guimarães Santiago (Vice-Cordenador)
18 %   Anderson Luiz Pedrosa Porto (Colaborador)
19 %   Flaviano Luiz Benfica (Bolsista)
20 %=====
21
22 % Calcula a ordem da matriz C
23 [m_C,n_C] = size(C);
24
25 % Calcula a ordem da matriz I
26 [m_B,n_B] = size(B);
27
28 % Verifica se o número de colunas de C é igual ao número de linhas de B
29 if m_C == m_B
30     % Identifica a matriz ampliada associada ao sistema linear
31     A = [C,B];
32     % Determina o posto de A
33     pA = rank(A);
34     % Encontra a forma escalonada reduzida associada a matriz A
35     E = rref(A);
36     % Verifica se pode ter ocorrido erro com o uso de rref
37     m_absE = rank(E);
38     % Verifica se o número de linhas de absE é diferente de pA
39     if m_absE ~= pA
40         fprintf('Possível erro associado a tolerância de rref()!\n');
41         E = [];
42     end
43 else
44     fprintf('Verifique as ordens as matrizes!!!')
45     A = [];
46     E = [];
47 end

```

```

1  function [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
2  % FORMA_ESCALONADA_REDUIZIDA: Função que retorna a forma escalonada reduzida
3  % da matriz A.
4  % Entrada:
5  %   A: Matriz de ordem mxn.
6  % Saída:
7  %   E: Matriz na forma escalonada reduzida equivalente a A, no caso de não
8  %       ocorrer erro ao usar a função rref().
9  %   E: [] no caso de ocorrer erro ao usar a função rref().
10 %   pA: Posto de A.
11 %   nA: Nulidade de A.
12 % Exemplo:
13 %   [E,pA,nA] = forma_escalonada_reduzida_matriz([1,2,-3,0;2,4,-2,2;3,6,-4,3])
14 %=====
15 %Importante: O uso dessa função para encontrar a forma escalonada reduzida em um
16 % caráter mais acadêmico. Com o uso da função rref() podem ocorrer erros numéricos.
17 % Dependendo da tolerância tol estabelecida para rref(), tem-se resultados
18 % diferentes.
19 %=====
20 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
21 % Software como Suporte para o Aprendizado de Álgebra Linear.
22 % Membros do Projeto:
23 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
24 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
25 %   Anderson Luiz Pedrosa Porto (Colaborador)
26 %   Flaviano Luiz Benfica (Bolsista)
27 %=====
28
29 % Determina o número de linhas e de colunas de A
30 [m,n] = size(A);
31
32 % Determina o posto de A
33 pA = rank(A);
34
35 % Determina a nulidade de A
36 nA = n - pA;
37
38 % Determina a forma escalonada reduzida equivalente a A.
39 E = rref(A);
40 % Verifica se pode ter ocorrido erro com o uso de rref
41 m_absE = rank(E);
42 % Verifica se o número de linhas de absE é diferente de pA
43 if m_absE ~= pA
44     fprintf('Possível erro associado a tolerancia de rref()!\n');
45     E = [];
46 end
47
48 end

```

```

1  function [detA,invA] = inversa_matriz(A)
2  % INVERSA_MATRIZ: Função que calcula a inversa da matriz A, caso exista.
3  % Entrada:
4  %   A: Matriz A de ordem nxn.
5  % Saída:
6  %   detA: Determinante da matriz A.
7  %   invA:
8  %       Matriz inversa de A de ordem nxn, caso exista.
9  %       [] se A não possui inversa.
10 % Exemplo:
11 %   [detA,invA] = inversa_matriz([2 3 -2;1 4 8;7 5 3])
12 %=====
13 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
14 % Software como Suporte para o Aprendizado de Álgebra Linear.
15 % Membros do Projeto:
16 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
17 %   Douglas Frederico Guimarães Santiago (Vice-Coordenador)
18 %   Anderson Luiz Pedrosa Porto (Colaborador)
19 %   Flaviano Luiz Benfica (Bolsista)
20 %=====
21
22 % Calcula a ordem da matriz A
23 [m,n] = size(A);
24
25 % Verifica a condição para calcular o determinante
26 if m == n
27     % Calcula o determinate da matriz A
28     detA = det(A);
29
30     % Devido a erros numéricos não é aconselhável usar o determinante para verificar
31     % se uma matriz quadrada A possui inversa. Para essa finalidade deve-se verificar
32     % o condicionamento da matriz em questão. Nesse sentido, usa-se aqui a função
33     % inv(), a qual retorna um warning no caso de A não ser bem condicionada.
34
35     % Limpa, caso exista, o último lastwarn retornado
36     lastwarn('');
37     warning('');
38     % Tenta calcular a inversa da matriz A
39     invA = inv(A);
40
41     % Verifica se retornou algum warning
42     [msg, msgid] = lastwarn();
43
44     fprintf('\n\n');
45     if (isempty(msg))
46         invA = invA;
47     else
48         fprintf('A matriz não possui inversa!!!\n');
49         invA = [];
50     end
51
52     %   if detA ~= 0
53     %       % Calcula a inversa da matriz A
54     %       invA = inv(A);
55     %   else
56     %       fprintf('A matriz não possui inversa!!!');
57     %       invA = [];
58     %   end
59
60 else
61     fprintf('A matriz não é quadrada!!!');
62     detA = [];
63     invA = [];
64 end
65
66
67 end

```

```
1  function[C] = produto_matriz_por_escalar(A,k)
2  % PRODUTO_MATRZ_POR_ESCALAR: Função que retorna a matriz C = k.A.
3  % Entrada:
4  %   A: Matriz A de ordem mxn.
5  %   k: Escalar qualquer
6  % Saída:
7  %   C: Matriz C = k.A de ordem mxn.
8  % Exemplo:
9  %   [C] = produto_matriz_por_escalar([1,2,3;4,5,6],-3)
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 %   Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a matriz C
21 C = k.*A;
22
23 end
```

```

1  function [C] = produto_matrizes(A,B)
2  % PRODUTO_MATRIZES: Função que retorna a matriz C = A.B
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem nxp.
6  % Saída:
7  % C: Matriz de ordem nxp.
8  % Exemplo:
9  % [C] = produto_matrizes([1,2,3;4,5,6],[7,7,9,10;11,12,13,14;1,2,3,4])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 %   Anderson Luiz Pedrosa Porto (Colaborador)
17 %   Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20
21 % Calcula a ordem da matriz A
22 [m_A,n_A] = size(A);
23
24 % Calcula a ordem da matriz B
25 [m_B,n_B] = size(B);
26
27 % Verifica se o produto matricial está definido
28 if n_A == m_B
29     % Calcula a matriz C
30     C = A*B;
31 else
32     fprintf('Produto de matrizes não está definido. Verifique as ordens as matrizes!!!\n')
33     C = [];
34 end
35
36 end

```

```
1  function [C] = soma_matrizes(A,B)
2  % SOMA_MATRIZES: Função que retorna a matriz C = A + B.
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem mxn.
6  % Saída:
7  % C: Matriz C = A+B de ordem mxn.
8  % Exemplo:
9  % [C] = soma_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 % Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 % Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 % Anderson Luiz Pedrosa Porto (Colaborador)
17 % Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m_A,n_A] = size(A);
22
23 % Calcula a ordem da matriz B
24 [m_B,n_B] = size(B);
25
26 % Verifica se a soma está definida
27 if m_A == m_B && n_A == n_B
28     % Calcula a matriz C
29     C = A + B;
30 else
31     fprintf('Soma de matrizes não está definida. Verifique as ordens das matrizes!!!\n')
32     C = [];
33 end
34
35 end
```



```

1  function [C] = subtracao_matrizes(A,B)
2  % SOMA_MATRIZES: Função que retorna a matriz C = A - B.
3  % Entrada:
4  % A: Matriz de ordem mxn.
5  % B: Matriz de ordem mxn.
6  % Saída:
7  % C: Matriz C = A+B de ordem mxn.
8  % Exemplo:
9  % [C] = subtracao_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
10 %=====
11 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
12 % Software como Suporte para o Aprendizado de Álgebra Linear.
13 % Membros do Projeto:
14 % Mônica Aparecida Cruvinel Valadão (Coordenadora)
15 % Douglas Frederico Guimarães Santiago (Vice-Coodenador)
16 % Anderson Luiz Pedrosa Porto (Colaborador)
17 % Flaviano Luiz Benfica (Bolsista)
18 %=====
19
20 % Calcula a ordem da matriz A
21 [m_A,n_A] = size(A);
22
23 % Calcula a ordem da matriz B
24 [m_B,n_B] = size(B);
25
26 % Verifica se a soma está definida
27 if m_A == m_B && n_A == n_B
28     % Calcula a matriz C
29     C = A - B;
30 else
31     fprintf('Subtração de matrizes não está definida. Verifique as ordens das
32             matrizes!!!\n')
33     C = [];
34 end
35 end

```

```
1  function[C] = transposta_matriz(A)
2  % TRANSPOSTA_MATRIZ: Função que retorna a matriz matriz transposta de A.
3  % Entrada:
4  %   A: Matriz A de ordem mxn.
5  % Saída:
6  %   C: Matriz C = A^{T} de ordem nxm.
7  % Exemplo:
8  %   [C] = transposta_matriz([1,2,3;4,5,6])
9  %=====
10 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
11 %   Software como Suporte para o Aprendizado de Álgebra Linear.
12 % Membros do Projeto:
13 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
14 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
15 %   Anderson Luiz Pedrosa Porto (Colaborador)
16 %   Flaviano Luiz Benfica (Bolsista)
17 %=====
18
19 % Calcula a matriz C
20 C = A';
21
22 end
```

```

1  function [O] = verifica_autovalor_autovetor_matriz(A,lambda,v)
2  % VERIFICA_AUTOVALOR_AUTOVETOR_MATRIZ: Função que retorna a forma geral da matriz
3  % resultante do produto (A - lambda.*I).v.
4  % Entrada:
5  %   A: Matriz de ordem nxn.
6  %   lambda: Escalar qualquer.
7  %   v: Vetor nx1
8  % Saída:
9  %   O: Matriz resultante do produto (A - lambda.*I).v.
10 % Exemplo:
11 %   [O] = verifica_autovalor_autovetor_matriz([-3,1,-1;-7,5,-1;-6,6,-2],-2, [-1;0;0])
12 %
13 %=====
14 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
15 %   Software como Suporte para o Aprendizado de Álgebra Linear.
16 % Membros do Projeto:
17 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
18 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
19 %   Anderson Luiz Pedrosa Porto (Colaborador)
20 %   Flaviano Luiz Benfica (Bolsista)
21 %=====
22 % Calcula a ordem da matriz A
23 [m,n] = size(A);
24
25 % Verifica a ordem da matriz
26 if m == n
27     % Identifica a matriz idenidade de ordem nxn
28     I = eye(n);
29     % Identifica a matriz nula de ordem nx1
30     O = zeros(n,1);
31     % Identifica a matriz (A - lambda.I)
32     C = (A - lambda.*I);
33     % Verifica se v é não nulo
34     if all(v == 0)
35         fprintf('O vetor v não pode ser nulo \n');
36         O = [];
37     else
38         rC = rank(C);
39         if (rC == m)
40             fprintf('lambda não é um autovalor de A \n');
41             O = [];
42         else
43             % Verifica ordem do vetor v
44             [mv,cv] = size(v);
45             if (mv == m)
46                 % Determina a matriz resultante de (A - autoval.*I).v
47                 O = ((A - lambda.*I)*v);
48             else
49                 fprintf('O vetor v deve ter m linhas \n');
50                 O = [];
51             end
52         end
53     end
54 else
55     fprintf('A matriz não é quadrada!!!');
56 end
57
58 end

```

```

1  function [p] = verifica_autovalor_matriz(A, lambda)
2  % VERIFICA_AUTOVALOR_MATRIZ: Função que retorna o valor p(lambda) onde p(.) é o
3  % polinômio característico de A
4  % Entrada:
5  %   A: Matriz de ordem nxn.
6  %   lambda: Escalar qualquer.
7  % Saída:
8  %   p: Valor dado por p(lambda).
9  % Exemplo:
10 %   [p] = verifica_autovalor_matriz([-3,1,-1;-7,5,-1;-6,6,-2],-2)
11 %
12 %=====
13 % Projeto Proae: Elaboração de Material Didático que Empregue o uso de
14 % Software como Suporte para o Aprendizado de Álgebra Linear.
15 % Membros do Projeto:
16 %   Mônica Aparecida Cruvinel Valadão (Coordenadora)
17 %   Douglas Frederico Guimarães Santiago (Vice-Coodenador)
18 %   Anderson Luiz Pedrosa Porto (Colaborador)
19 %   Flaviano Luiz Benfica (Bolsista)
20 %=====
21 % Calcula a ordem da matriz A
22 [m,n] = size(A);
23
24 % Verifica a ordem da matriz
25 if m == n
26     % Encontra os coeficientes do polinômio característico
27     c = poly(A);
28
29     % Retorna os coeficientes considerando o determinante de (A - lambda.*I)
30     c = -c;
31     c = c(:);
32
33     % Identifica a quantidade de coeficientes
34     nc = length(c);
35
36     % Verifica se lambda é solução do polinômio característico
37     xlambda = lambda*ones(1,(nc-1));
38     e = [1:(nc-1)];
39     [ex,~] = sort(e,'descend');
40     xlambda_ex = xlambda.^ex;
41     xlambda_ex = xlambda_ex(:);
42     p = (c(1:(nc - 1)))'*xlambda_ex + c(nc);
43 else
44     fprintf('A matriz não é quadrada!!!');
45     p = [];
46 end
47
48
49
50 end

```