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 Benfica (Bolsista)

Diamantina-MG
2020

Sumário

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 livres 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.

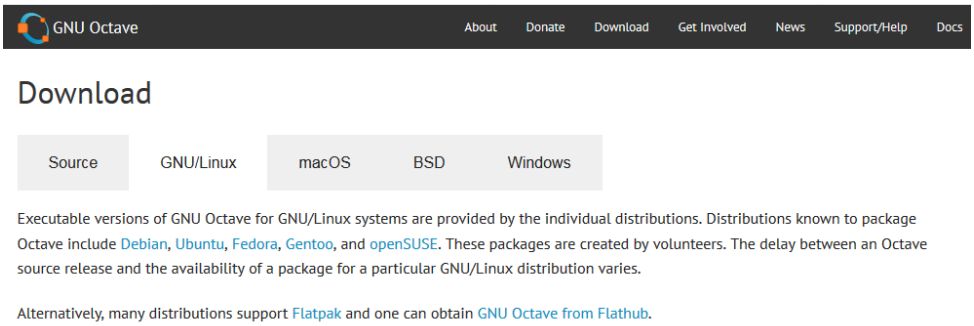

Figura 1 – GNU Octave.

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.

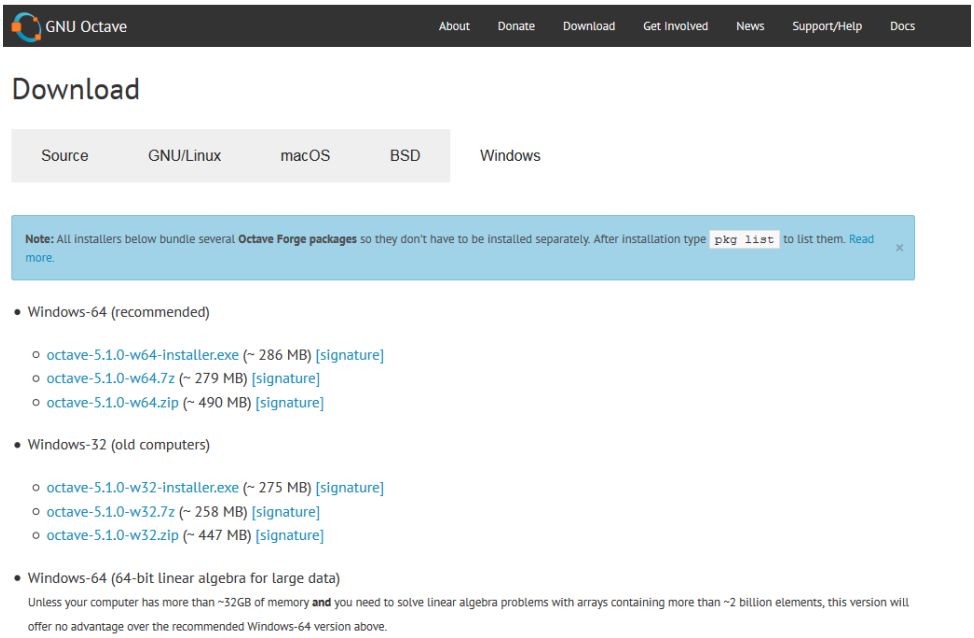

Figura 2 – GNU Octave.

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.*

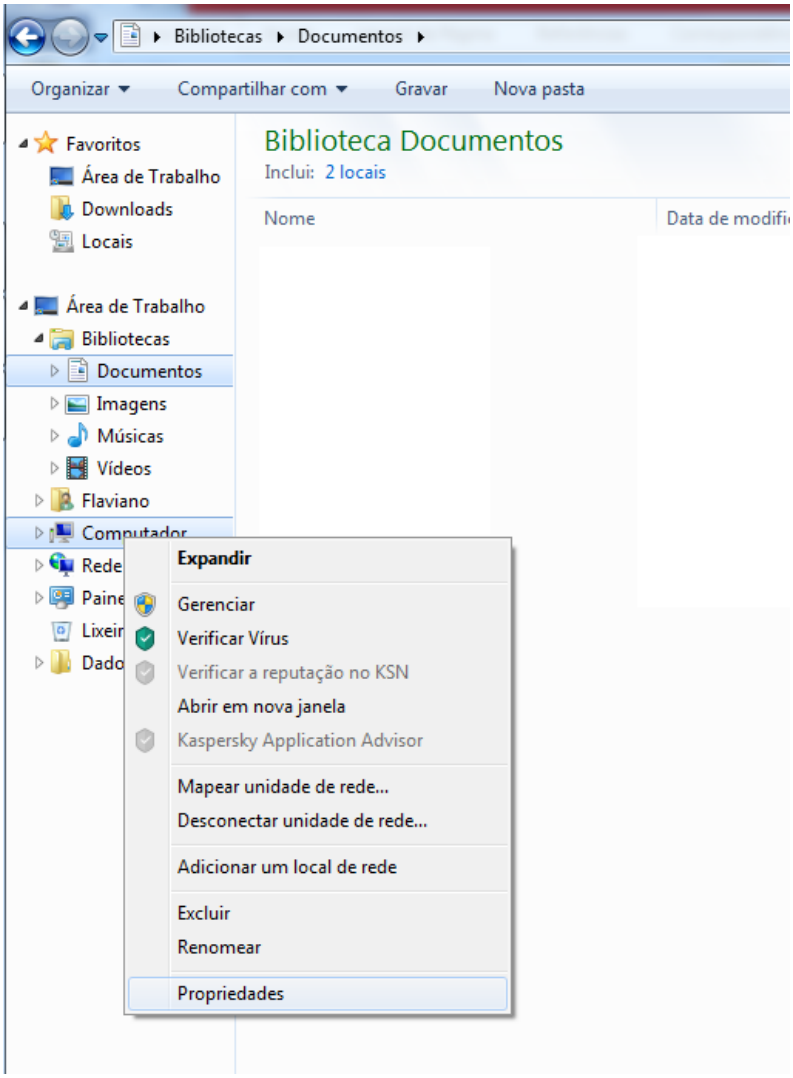

Figura 3 – GNU Octave.

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 dar enter (será exigida a senha de superusuário):

   sudo apt install octave.

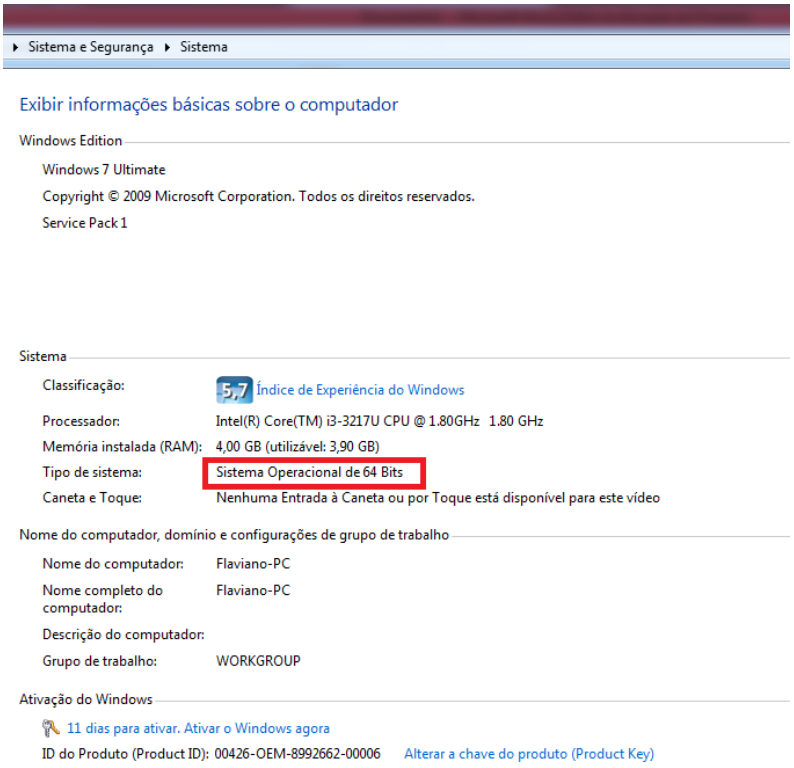

Figura 4 – GNU Octave.

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/>. Não utilizaremos esta ferramenta, pois você pode também instalar via linha de comando no terminal. De qualquer forma, é antes necessario 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>.

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

$$[\mathrm{A}] = \mathrm{construa\_matriz}(\mathrm{regra\_ii}, \mathrm{regra\_ij}, \mathrm{m}, \mathrm{n}).$$

As entradas dessa função fornecem a definição para a construção de uma matriz $\mathrm{A} = [a_{ij}]_{m\times n}$, 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 contrução de uma matriz $\mathrm{A} = [a_{ij}]_{4\times 7}$ 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 =

$$\begin{matrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 0 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 0 & 4 & 4 & 4 \end{matrix}$$

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] = construa_matriz_caso_geral(regra_ii, regra_ij, cond_ij, 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 cond_ij = valor_cond;
regra_ij : regra associada a condição dos ínidces cond_ij $\neq$ valor_cond.
Essas regras e condições devem ser inseridas no formato de função anônima $@(i,j)()$.

Os Exemplos 3 e 4 ilustram a contrução 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

$$[\mathrm{C}] = \mathrm{soma\_matrizes}(\mathrm{A}, \mathrm{B}),$$

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

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

$$[\mathrm{C}] = \mathrm{subtracao\_matrizes}(\mathrm{A}, \mathrm{B}),$$

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

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

$$[\mathrm{C}] = \mathrm{produto\_matrizes}(\mathrm{A}, \mathrm{B}),$$

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

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

$$[\mathrm{C}] = \text{produto\_matriz\_por\_escalar}(\mathrm{A}, \mathrm{k}).$$

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

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

$$[\mathrm{C}] = \text{transposta\_matriz}(\mathrm{A}).$$

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

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 =

$$\begin{matrix} 3 & 2 & 3 \\ 5 & 4 & 1 \\ 1 & 7 & 8 \end{matrix}$$

3.8 Forma Escalonada Reduzida de Uma Matriz

O arquivo "forma_escalonada_reduzida_matriz.m" implementa a função

$$[\mathrm{E}, \mathrm{pA}, \mathrm{nA}] = \mathrm{forma\_escalonada\_reduzida\_matriz(A)},$$

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

$$\begin{matrix} 1 & 2 & -3 & 0 \\ 2 & 4 & -2 & 2 \\ 3 & 6 & -4 & 3 \end{matrix}$$

>> [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
E =

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

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

$$[\mathrm{A}, \mathrm{E}] = \mathrm{escalonada\_reduzida\_matriz\_ampliada(C, B)}.$$

As entradas das dessa função são uma matriz $\mathrm{C} = [c_{ij}]_{m\times n}$ e $\mathrm{B} = [b_{jk}]_{m\times 1}$. 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 =

$$\begin{matrix} 3 & 2 & -5 \\ 2 & -4 & -2 \\ 1 & -2 & -3 \end{matrix}$$

>> B = [8; −4; −4]

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

>> [A, E] = escalonada_reduzida_matriz_ampliada(C, B)

A =

$$\begin{matrix} 3 & 2 & -5 & 8 \\ 2 & -4 & -2 & -4 \\ 1 & -2 & -3 & -4 \end{matrix}$$

E =

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

3.10 Verifica Solução Sistema Linear

O arquivo "verifica_solucao_sistema_linear.m" implementa a função

[R, B] = verifica_solucao_sistema_linear(C, B, X).

As entradas das 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 =

$$\begin{matrix} 3 & 2 & -5 \\ 2 & -4 & -2 \\ 1 & -2 & -3 \end{matrix}$$

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

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

$>>$ X $= [-2; 3; 1]$

X =

$$\begin{matrix} -2 \\ 3 \\ 1 \end{matrix}$$

$>>$ [R, B] = verifica_solucao_sistema_linear(C, B, X)

R =

$$\begin{matrix} -5 \\ -18 \\ -11 \end{matrix}$$

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

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

C =

$$\begin{matrix} 3 & 2 & -5 \\ 2 & -4 & -2 \\ 1 & -2 & -3 \end{matrix}$$

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

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

$>>$ X $= [3; 2; 1]$

X =

$$\begin{matrix} 3 \\ 2 \\ 1 \end{matrix}$$

$>>$ [R, B] = verifica_solucao_sistema_linear(C, B, X)

R =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

3.11 Solução de Um Sistema de Equações Lineares

O arquivo "solucao_sistema_linear.m" implementa a função

$$[\mathrm{pA}, \mathrm{pC}, \mathrm{A}, \mathrm{S}, \mathrm{E}] = \mathrm{solucao\_sistema\_linear}(\mathrm{C}, \mathrm{B}).$$

As entradas das dessa função são uma matriz $\mathrm{C} = [c_{ij}]_{m\times n}$ e $\mathrm{B} = [b_{jk}]_{m\times 1}$. 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.

$>> \mathrm{C} = [3, 2, -5; 2, -4, -2; 4, -8, -4]$

C =

$$\begin{matrix} 3 & 2 & -5 \\ 2 & -4 & -2 \\ 4 & -8 & -4 \end{matrix}$$

$>> \mathrm{B} = [8; -4; -8]$

B =

$$\begin{matrix} 8 \\ -4 \\ -8 \end{matrix}$$

$>> [\mathrm{pA}, \mathrm{pC}, \mathrm{A}, \mathrm{S}, \mathrm{E}] = \mathrm{solucao\_sistema\_linear}(\mathrm{C}, \mathrm{B})$

Sistema possui infinitas solucoes

pA =

$$\begin{matrix} 2 \end{matrix}$$

pC =

$$\begin{matrix} 2 \end{matrix}$$

A =

$$\begin{matrix} 3 & 2 & -5 & 8 \\ 2 & -4 & -2 & -4 \\ 4 & -8 & -4 & -8 \end{matrix}$$

S =

[ ]

E =

$$\begin{matrix} 1.0000 & 0 & -1.5000 & 1.5000 \\ 0 & 1.0000 & -0.2500 & 1.7500 \\ 0 & 0 & 0 & 0 \end{matrix}$$

Exemplo 15.

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

C =

$$\begin{matrix} 3 & 2 & -5 \\ 2 & -4 & -2 \\ 4 & -8 & -4 \end{matrix}$$

>> B = [8; −4; −4]

B =

$$\begin{matrix} 8 \\ -4 \\ -4 \end{matrix}$$

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

Sistema nao possui solucao

pA =

3

pC =

2

A =

$$\begin{matrix} 3 & 2 & -5 & 8 \\ 2 & -4 & -2 & -4 \\ 4 & -8 & -4 & -4 \end{matrix}$$

S =

[ ]

E =

$$\begin{matrix} 1.0000 & 0 & -1.5000 & 0 \\ 0 & 1.0000 & -0.2500 & 0 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

3.12 Determinate de Uma Matriz

O arquivo “determinante_matriz.m” implementa a função

[detA] = 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
>> [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

[detA] = 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
==========
detA = -40

3.12.2 Escolhendo Uma Coluna Para Calcular o Determinante

O arquivo "determinante_matriz_escolhe_coluna.m" implementa a função

[detA] = 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 A_{11}:

8 7

2 3

Determinante da submatriz A_{11}:

10

Cofator Cof_{11}:

10

==========

Submatriz A_{12}:

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 Cof_{31}*:*

-10

==========

Submatriz A_{32}*:*

1 3

4 7

Determinante da submatriz A_{32}*:*

-5

Cofator Cof_{32}*:*

5

==========

Submatriz A_{33}*:*

1 2

4 8

Determinante da submatriz A_{33}*:*

0

Cofator Cof_{33}*:*

0

==========

CofA =

$$\begin{matrix} 10 & 23 & -32 \\ -0 & -12 & 8 \\ -10 & 5 & 0 \end{matrix}$$

3.14 Matriz Adjunta

O arquivo "adjunta_matriz.m" implementa a função

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

cuja a entrada é uma matriz $\text{A} = [a_{ij}]_{n\times n}$. 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 trasposta 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] = 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 =

$$\begin{matrix} -3 & 1 & -1 \\ -2 & 5 & -1 \\ -6 & 7 & -8 \end{matrix}$$

$>>$ [c] = coeficientes_polinomio_caracteristico(A)

c =

$$\begin{matrix} 1.0000 & 6.0000 & -28.0000 & -73.0000 \end{matrix}$$

3.17 Autovalores

O arquivo "autovalores_matriz.m" implementa a função

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

A entrada dessa função é uma matriz $\text{A} = [a_{ij}]_{n\times n}$ 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 =

$$\begin{matrix} 4.0000 \\ -2.0000 \\ -2.0000 \end{matrix}$$

3.18 Autovetores Associados a Um Autovalor

O arquivo "autovetores_matriz.m" implementa a função

$$[\text{M\_autvet}, \text{M\_autoval}] = \text{autovetores\_matriz(A, autoval)}$$

cuja entrada é uma matriz $\text{A} = [a_{ij}]_{n\times n}$ 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

$$(\text{A} - \text{autoval}.\text{I}_n).\text{v} = \text{o} \tag{3.1}$$

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 =

$$\begin{matrix} -3 & 1 & -1 \\ -7 & 5 & -1 \\ -6 & 6 & -2 \end{matrix}$$

$>>$ autoval $= -2$

autoval $= -2$

$>>$ [M_autvet, M_autoval] $=$ autovetores_matriz(A, autoval)

M_autvet =

$$\begin{matrix} 1 & -1 & 0 & -0 \\ 0 & 0 & 1 & -0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

M_autoval $= -2$

$>>$ A $= [1, 7, 3; 2, 9, 12; 5, 22, 7]$

A =

$$\begin{matrix} 1 & 7 & 3 \\ 2 & 9 & 12 \\ 5 & 22 & 7 \end{matrix}$$

$>>$ 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 =

$$\begin{matrix} -0.260977 & -0.973445 & 0.189104 \\ -0.587027 & 0.228061 & -0.581573 \\ -0.766349 & -0.019808 & 0.791210 \end{matrix}$$

M_autoval =

$$\begin{matrix} 25.55484 & 0.00000 & 0.00000 \\ 0.00000 & -0.57893 & 0.00000 \\ 0.00000 & 0.00000 & -7.97590 \end{matrix}$$

3.19 Verifica Autovalor

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

$$[\text{p}] = \text{verifica\_autovalor\_matriz}(\text{A}, \text{lambda})$$

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

Exemplo 25.

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

A =

$$\begin{matrix} -3 & 1 & -1 \\ -7 & 5 & -1 \\ -6 & 6 & -2 \end{matrix}$$

>> 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

$$[\text{O}] = \text{verifica\_autovalor\_autovetor\_matriz}(\text{A}, \text{lambda}, \text{v})$$

cujas entradas são uma matriz $\text{A} = [a_{ij}]_{n\times n}$, 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

$$(\text{A} - \text{lambda}.\text{I}_n).\text{v} \tag{3.2}$$

No caso quem 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 $=$

$$\begin{matrix} -3 & 1 & -1 \\ -7 & 5 & -1 \\ -6 & 6 & -2 \end{matrix}$$

$>>$ lambda $= -2$

lambda $= -2$

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

v $=$

$$\begin{matrix} -1 \\ 0 \\ 0 \end{matrix}$$

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

O $=$

$$\begin{matrix} 1 \\ 7 \\ 6 \end{matrix}$$

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 real 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 dar 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 dê 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
$$[\mathrm{A}] = \mathrm{construa\_matriz}(\mathrm{regra\_ii}, \mathrm{regra\_ij}, \mathrm{m}, \mathrm{n})$$
para verificar a resolução de cada item da Questão 4.1.1.

- Use a função
$$[\mathrm{A}] = \mathrm{construa\_matriz\_caso\_geral}(\mathrm{regra\_ii}, \mathrm{regra\_ij}, \mathrm{cond\_ij}, \mathrm{valor\_cond}, \mathrm{m}, \mathrm{n})$$
para verificar a resolução de cada item da Questão 4.1.2.

- Use as funções
$$[\mathrm{C}] = \mathrm{soma\_matrizes}(\mathrm{A}, \mathrm{B}),$$
$$[\mathrm{C}] = \mathrm{produto\_matrizes}(\mathrm{A}, \mathrm{B}),$$
e
$$[\mathrm{C}] = \mathrm{transposta\_matriz}(\mathrm{A}).$$
para verificar cada item da Questão 4.1.3.

- Use a função
$$[\mathrm{C}] = \mathrm{produto\_matrizes}(\mathrm{A}, \mathrm{B}),$$
para verificar a resolução das Questões 4.1.4 e 4.1.5.

Questão 4.1.1. *Construa as matrizes a seguir:*

(a) $A = [a_{ij}]_{4\times 7}$ *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

$$[\mathrm{E}, \mathrm{pA}, \mathrm{nA}] = \mathrm{forma\_escalonada\_reduzida\_matriz(A)}$$

para verificar a resolução de cada item da Questão 4.2.1.

Questão 4.2.1. *Determine o posto e a nulidade de cada matrize 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

$$[\mathrm{R}, \mathrm{B}] = \mathrm{verifica\_solucao\_sistema\_linear}(\mathrm{C}, \mathrm{B}, \mathrm{X}).$$

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

$$[\mathrm{pA}, \mathrm{pC}, \mathrm{A}, \mathrm{S}, \mathrm{E}] = \mathrm{solucao\_sistema\_linear}(\mathrm{C}, \mathrm{B}).$$

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.

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

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

```
>> 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

$$[\text{detA}] = \text{determinante\_matriz\_escolhe\_coluna(A)}$$

para verificar a resolução da Questão 4.6.1.

Questão 4.6.1. *Calcule o determinante da matriz* $A = \begin{bmatrix} 2 & 3 & 0 & 5 & 5 \\ 4 & 5 & 0 & 3 & 2 \\ 4 & 2 & 1 & 1 & 5 \\ 1 & 3 & -5 & 1 & 2 \\ 1 & 2 & 0 & 2 & 5 \end{bmatrix}$.

4.7 Atividade 7

Usando o Octave

- Use a função

$$[\text{detA}] = \text{determinante\_matriz\_escolhe\_linha(A)}$$

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

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

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

$$[\text{detA}] = \text{determinante\_matriz\_escolhe\_linha(A)}$$

$$[\text{C}] = \text{produto\_matriz\_por\_escalar(A, k)}$$

  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

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

  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

$$[\mathrm{c}] = \mathrm{coeficientes\_polinomio\_caracteristico(A)}$$

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

$$[\mathrm{c}] = \mathrm{coeficientes\_polinomio\_caracteristico(A)}$$

$$[\mathrm{autoval}] = \mathrm{autovalores\_matriz(A)}$$

$$[\mathrm{O}] = \mathrm{verifica\_autovalor\_autovetor\_matriz(A, lambda, v)}$$

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 $\mathrm{v} = [0.73254; 0.51783; 0.54120]$ *é um autovetor de* A *associado ao autovalor* $\mathrm{lambda} = \mathrm{autoval}(1)$.

Para resolver o item (c) digite na janela de comandos os seguintes passos:
$>>$ $\mathrm{A} = [-9, 3, -2; 7, 5, 3; 0, -4, 0]$
$>>$ $[\mathrm{autoval}] = \mathrm{autovalores\_matriz(A)}$
$>>$ $\mathrm{v} = [0.73254; 0.51783; 0.54120]$
$>>$ $\mathrm{lambda} = \mathrm{autoval}(1)$
$>>$ $[\mathrm{O}] = \mathrm{verifica\_autovalor\_autovetor\_matriz(A, lambda, v)}$

4.12 Atividade 12

Usando o Octave

- Use a função

$$[\text{M\_autvet}, \text{M\_autoval}] = \text{autovetores\_matriz}(\text{A}, \text{autoval})$$

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

```
function [A] = construa_matriz(regra_ii,regra_ij,m,n)
% CONSTRUA_MATRIz: Função que retorna a matriz A de ordem mxn
% correspondente a definição fornecida pelas entradas da função.
% Entrada:
%   regra_ii: Regra associada a condição i=j, sendo que i,j representam,
%   respectivamente, linha e coluna de A. Essa regra deve ser inserida na
%   forma de @(i,j)().
%   regra_ij: Regra associada a condição i~=j, sendo que i,j representam,
%   respectivamente, linha e coluna de A. Essa regra deve ser inserida na
%   forma de @(i,j)().
%   m: Número de linhas de A.
%   n: Número de colunas de A.
% Saída:
%   A: Matriz A de ordem mxn
% Exemplo:
%   Duas condições para os índices
%       [A] = construa_matriz(@(i,j)(i - j),@(i,j)(i),4,7)
%   Apenas uma condição para os índices
%       [A] = construa_matriz(@(i,j)(i),@(i,j)(i),4,7)
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Construa a matriz A
for i = 1:m
    for j = 1:n
        if i == j
            A(i,j) = regra_ii(i,j);
        else
            A(i,j) = regra_ij(i,j);
        end
    end
end


end
```

```
function [A] = construa_matriz_caso_geral(regra_ii,regra_ij,cond_ij,valor_cond,m,n)
% CONSTRUA_MATRIz: Função que retorna a matriz A de ordem mxn
% correspondente a definição fornecida pelas entradas da função.
% Entrada:
%   regra_ii: Regra associada a condição cond_ij = cond_valor, sendo que i,j
%   representam, %   respectivamente, linha e coluna de A. Essa regra deve ser
%   inserida na forma de @(i,j)().
%   regra_ij: Regra associada a condição cond_ij = valor_cond, sendo que i,j
%   representam, respectivamente, linha e coluna de A. Essa regra deve ser
%   inserida na forma de @(i,j)().
%   cond_ij: Condicao estabeleciada aos indices i,j da matriz. Essa regra deve
%   ser inserida na forma de @(i,j)().
%   valor_cond: Valor associado a cond_ij.
%   m: Número de linhas de A.
%   n: Número de colunas de A.
% Saída:
%   A: Matriz A de ordem mxn
% Exemplo:
%   Duas condições para os índices
%   [A] = construa_matriz_caso_geral(@(i,j)(2*i),@(i,j)(j+2),@(i,j)(i+j),4,4,7)
%   Apenas uma condição para os índices
%   [A] = construa_matriz_caso_geral(@(i,j)(2*i),@(i,j)(2*i),@(i,j)(i+j),4,4,7)
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Construa a matriz A
for i = 1:m
    for j = 1:n
        if  cond_ij(i,j) == valor_cond
            A(i,j) = regra_ii(i,j);
        else
            A(i,j) = regra_ij(i,j);
        end
    end
end


end
```

```
function [C] = soma_matrizes(A,B)
% SOMA_MATRIZES: Função que retorna a matriz C = A + B.
% Entrada:
% A: Matriz de ordem mxn.
% B: Matriz de ordem mxn.
% Saída:
% C: Matriz C = A+B de ordem mxn.
% Exemplo:
%   [C] = soma_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a ordem da matriz A
[m_A,n_A] = size(A);

% Calcula a ordem da matriz B
[m_B,n_B] = size(B);

% Verifica se a soma está definida
if m_A == m_B && n_A == n_B
    % Calcula a matriz C
    C = A + B;
else
    fprintf('Soma de matrizes não está definida. Verifique as ordens das matrizes!!!\n')
    C = [];
end

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-Coordenador)
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
        matrizes!!!\n')
32      C = [];
33  end
34
35  end
```

```
function [C] = produto_matrizes(A,B)
% PRODUTO_MATRIZES: Função que retorna a matriz C = A.B
% Entrada:
% A: Matriz de ordem mxn.
% B: Matriz de ordem nxp.
% Saída:
% C: Matriz de ordem nxp.
% Exemplo:
%   [C] = produto_matrizes([1,2,3;4,5,6],[7,7,9,10;11,12,13,14;1,2,3,4])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================


% Calcula a ordem da matriz A
[m_A,n_A] = size(A);

% Calcula a ordem da matriz B
[m_B,n_B] = size(B);

% Verifica se o produto matricial está definido
if n_A == m_B
    % Calcula a matriz C
    C = A*B;
else
    fprintf('Produto de matrizes não está definido. Verifique as ordens as matrizes!!!\n')
    C = [];
end

end
```

```
function[C] = produto_matriz_por_escalar(A,k)
% PRODUTO_MATRZ_POR_ESCALAR: Função que retorna a matriz C = k.A.
% Entrada:
%   A: Matriz A de ordem mxn.
%   k: Escalar qualquer
% Saída:
%   C: Matriz C = k.A de ordem mxn.
% Exemplo:
%   [C] = produto_matriz_por_escalar([1,2,3;4,5,6],-3)
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a matriz C
C = k.*A;

end
```

```
function[C] = transposta_matriz(A)
% TRANSPOSTA_MATRIZ: Função que retorna a matriz matriz transposta de A.
% Entrada:
%   A: Matriz A de ordem mxn.
% Saída:
%   C: Matriz C = A^{T} de ordem nxm.
% Exemplo:
%   [C] = transposta_matriz([1,2,3;4,5,6])
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a matriz C
C = A';

end
```

```
function [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
% FORMA_ESCALONADA_REDUZIDA: Função que retorna a forma escalonda reduzida
% da matriz A.
% Entrada:
%   A: Matriz de ordem mxn.
% Saída:
%   E: Matriz na forma escalonada reduzida equivalente a A, no caso de não
%      ocorrer erro ao usar a função rref().
%   E: [] no caso de ocorrer erro ao usar a função rref().
%   pA: Posto de A.
%   nA: Nulidade de A.
% Exemplo:
%   [E,pA,nA] = forma_escalonada_reduzida_matriz([1,2,-3,0;2,4,-2,2;3,6,-4,3])
%=========================================================================
%Importante: O uso dessa função para encontrar a forma escalonada reduzida em um
% caráter mais acadêmico. Com o uso da função rref()podem ocorrer erros numérios.
% Dependendo da tolerância tol estabelecida para rref(), tem-se resultados
% diferentes.
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Determina o número de linhas e de colunas de A
[m,n] = size(A);

% Determina o posto de A
pA = rank(A);

% Determina a nulidade de A
nA = n - pA;

% Determina a forma escalonada reduzida equivalente a A.
E = rref(A);
% Verifica se pode ter ocorrido erro com o uso de rref
m_absE = rank(E);
% Verifica se o número de linhas de absE é diferente de pA
if m_absE ~= pA
    fprintf('Possivel erro associado a tolerancia de rref()!\n');
    E = [];
end

end
```

```
function [A,E] = escalonada_reduzida_matriz_ampliada(C,B)
% ESCALONADA_REDUZIDA_MATRIZ_AMPLIADA: Função que Retorna a matriza ampliada
% e a forma escalonada reduzida associada.
% Entrada:
%   C: Matriz dos coeficientes de ordem mxn.
%   B: Matriz dos termos independentes de ordem mx1.
% Saída:
%   A: Matriz ampliada de ordem mx(n+1).
%   E: Matriz na forma escalonada reduzida equivalente a A.
% Exemplo:
%   [A,E] = escalonada_reduzida_matriz_ampliada([3,2,-5;2,-4,-2;1,-2,-3],[8;-4;-4])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz C
[m_C,n_C] = size(C);

% Calcula a ordem da matriz I
[m_B,n_B] = size(B);

% Verifica se o número de colunas de C é igual ao número de linhas de B
if m_C == m_B
    % Identifica a matriz ampliada associadao ao sitema linear
    A = [C,B];
    % Determina o posto de A
    pA = rank(A);
    % Encontra a forma escalonada reduzida associada a matriz A
    E = rref(A);
    % Verifica se pode ter ocorrido erro com o uso de rref
    m_absE = rank(E);
    % Verifica se o número de linhas de absE é diferente de pA
    if m_absE ~= pA
        fprintf('Possivel erro associado a tolerancia de rref()!\n');
        E = [];
    end
else
    fprintf('Verifique as ordens as matrizes!!!')
    A = [];
    E = [];
end
```

```
function[R,B] = verifica_solucao_sistema_linear(C,B,X)
% VERIFICA_SOLUCAO_SISTEMA_LINEAR: Função que verifica se o S é solução
% do sistema linear representado por AX = B
% solução. Retorna a solução em caso afirmativo para sistema possível e
% determinado. Retorna a matriz na forma escalonada reduzida associada ao
% sistema linear em questão.
% Entrada:
%   C: Matriz dos coeficientes nxn.
%   B: Matriz dos termos indepentendes nx1.
%   X: Matriz nx1
% Saída:
%   R: Matriz resultante R = A.X.
%   B: Matriz dos termos indepentendes nx1.
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================
% Exemplo:
% [R,B] = verifica_solucao_sistema_linear([3 2 -5;2 -4 -2;1 -2 -3],[8;-4;-4],[-2;3;1])

% Calcula a ordem da matriz C
[m_C,n_C] = size(C);

% Calcula a ordem da matriz X
[m_X,n_X] = size(X);

% Verifica se o produto AX está definido
if n_C == m_X
  R = C*X;
else
  fprintf('Verifique as ordens das matrizesn\n');
  R = [];

end

```

```
function[pA, pC, A, S, E] = solucao_sistema_linear(C,B)
% SOLUCAO_SISTEMA_LINEAR: Função que verifica se o sistema linear possui
% solução. Retorna a solução em caso afirmativo para sistema possível e
% determinado. Retorna a matriz na forma escalonada reduzida associada ao
% sistema linear em questão.
% Entrada:
%   C: Matriz dos coeficientes mxn.
%   B: Matriz dos termos indepentendes mx1.
% Saída:
%   pA: Posto da matriz ampliada A .
%   pC: Posto da matriz dos coeficientes C.
%   A: Matriz ampliada de ordem mx(n+1).
%   E:
%       Matriz de ordem mx(nx1) na forma escalonada reduzida equivalente a A
%       [] se ocorre erro associado a rref().
%   S:
%       matriz nx1 se Sistema Possível e Determinado.
%       [] se Sistema Impossível.
%       [] se Sistema Possível e Indeterminado.
%       [] se ocorre erro associado a rref().
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================
% Exemplo:
%   [pA, pC, A, S, E] = solucao_sistema_linear([3 2 -5;2 -4 -2;1 -2 -3],[8;-4;-4])
% Calcula a ordem da matriz C
[m_C,n_C] = size(C);
% Calcula a ordem da matriz B
[m_B,n_B] = size(B);
% Verifica se o número de linhas de C é igual ao número de linhas de B
if m_C == m_B
    A = [C,B];
    % Determina o posto de A
    pA = rank(A);
    % Determina o posto de C
    pC = rank(C);
    % Identifica o número de variáveis
    n = n_C;
    % Verifica se o sitema possui solução
    if pA == pC
        if pA == n
            fprintf('Sistema possui única solucao\n');
            S = linsolve(C,B);
        else
            fprintf('Sistema possui infinitas soluções\n');
            S = [];
        end
    else
        fprintf('Sistema não possui solução\n');
        S = [];
    end
    % Encontra a forma escalonada reduzida associada a matriz A
    E = rref(A);
    % Verifica se pode ter ocorrido erro com o uso de rref
    m_absE = rank(E);
    if m_absE ~= pA
        fprintf('Possível erro associado a tolerância de rref()!\n');
        E = [];
    end
else
    fprintf('Verifique as ordens das matrizes!!!\n');
    pA = [];
    pC = [];
    A = [];
    E = [];
    S = [];
end
end
```

```
function [detA] = determinante_matriz(A)
% DETERMINANTE_MATRIZ: Função que calcula o determinante da matriz A.
% Entrada:
%   A: Matriz A de ordem nxn.
% Saída:
%   detA: Determinante da matriz A.
% Exemplo:
%   [detA] = determinante_matriz([1,2,3;4 8 7;5 2 3])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz A
[m,n] = size(A);

% Verifica a condição para calcular o determinante
if m == n
    detA = det(A);
else
    fprintf('A matriz não é quadrada!!!\n');
    detA = [];
end



end
```

```
function [detA] = determinante_matriz(A)
% DETERMINANTE_MATRIZ: Função que calcula o determinante da matriz A.
% Entrada:
%   A: Matriz A de ordem nxn.
% Saída:
%   detA: Determinante da matriz A.
% Exemplo:
%   [detA] = determinante_matriz([1,2,3;4 8 7;5 2 3])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz A
[m,n] = size(A);

% Verifica a condição para calcular o determinante
if m == n
    detA = det(A);
else
    fprintf('A matriz não é quadrada!!!\n');
    detA = [];
end



end
```

```
function [detA] = determinante_matriz_escolhe_coluna(A)
% DETERMINANTE_MATRIZ_ESCOLHE_COLUNA: Função que calcula o determinante
% da matriz A escolhendo uma coluna qualquer de A.
% Entrada:
%   A: Matriz A de ordem mxm.
% Saída:
%   detA: Determinante da matriz A.
% Exemplo:
%   [detA] = determinante_matriz_escolhe_coluna([1,2,3;4 8 7;5 2 3])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz A
[m,n] = size(A);
% Imprime a matriz A
fprintf('\n Matriz A:\n');
disp(A)
% Imprime o número de linhas de A
fprintf('\n Número de linhas de A: \n');
disp(m)
% Imprime o número de colunas de A
fprintf('\n Número de colunas de A: \n');
disp(n)
% Verifica a condição para calcular o determinante (se A é matriz quadrada)
if m == n
    nL = [1:m];
    Lna = input('Escolha uma coluna qualquer da matriz A:  ','s');
    L = str2num(Lna);
    vL = find(nL == L);
    if isempty(vL)
      fprintf('\n Indice de coluna incorreto \n');
      detA = [];
    else
      fprintf('=============================================\n');
      fprintf('Identifica cada submatriz e cada cofator\n');
      fprintf('=============================================\n');
      Dtnante = 0;
      for j = 1:n
        k = strcat(num2str(j),num2str(L));
        fprintf('Submatriz A_%s:\n\n',k);
        S = A;
        S(:,L) = [];
        S(j,:) = [];
        disp(S)
        fprintf('Determinante da submatriz A_%s:\n\n',k);
        detS = det(S);
        disp(detS)
        fprintf('Cofator Cof_%s:\n\n',k);
        cof = (-1)^(j+L)*detS;
        disp(cof)
        Dtnante = Dtnante + A(j,L)*cof;
        fprintf('==============\n');
      end
      detA = Dtnante;
%      detA = det(A);
%      % Compara Dtnante com detA
%      if Dtnante ~= detA
%        fprintf('\n Possível erro numérico \n');
%      end
    end
else
    fprintf('A matriz não é quadrada!!!\n');
    detA = [];
end

end
```

```
function [detA] = determinante_matriz_escolhe_linha(A)
% DETERMINANTE_MATRIZ_ESCOLHE_LINHA: Função que calcula o determinante
% da matriz A escolhendo uma linha qualquer de A.
% Entrada:
%   A: Matriz A de ordem mxm.
% Saída:
%   detA: Determinante da matriz A.
% Exemplo:
%   [detA] = determinante_matriz_escolhe_linha([1,2,3;4 8 7;5 2 3])
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a ordem da matriz A
[m,n] = size(A);
% Imprime a matriz A
fprintf('\n Matriz A:\n');
disp(A)
% Imprime o número de linhas de A
fprintf('\n Número de linhas de A: \n');
disp(m)
% Imprime o número de colunas de A
fprintf('\n Número de colunas de A: \n');
disp(n)
% Verifica a condição para calcular o determinante (se A é matriz quadrada)
if m == n
    nL = [1:m];
    Lna = input('Escolha uma linha qualquer da matriz A:  ','s');
    L = str2num(Lna);
    vL = find(nL == L);
    if isempty(vL)
      fprintf('\n Indice de linha incorreto \n');
      detA = [];
    else
      fprintf('=============================================\n');
      fprintf('Identifica cada submatriz e cada cofator\n');
      fprintf('=============================================\n');
      Dtnante = 0;
      for j = 1:n
        k = strcat(num2str(L),num2str(j));
        fprintf('Submatriz A_%s:\n\n',k);
        S = A;
        S(L,:) = [];
        S(:,j) = [];
        disp(S)
        fprintf('Determinante da submatriz A_%s:\n\n',k);
        detS = det(S);
        disp(detS)
        fprintf('Cofator Cof_%s:\n\n',k);
        cof = (-1)^(L+j)*detS;
        disp(cof)
        Dtnante = Dtnante + A(L,j)*cof;
        fprintf('==============\n');
      end
      detA = Dtnante;
%      detA = det(A);
%      % Compara Dtnante com detA
%      if Dtnante ~= detA
%        fprintf('\n Possível erro numérico \n');
%      end
    end
else
    fprintf('A matriz não é quadrada!!!\n');
    detA = [];
end

end
```

```
function [A,E] = escalonada_reduzida_matriz_ampliada(C,B)
% ESCALONADA_REDUZIDA_MATRIZ_AMPLIADA: Função que Retorna a matriza ampliada
% e a forma escalonada reduzida associada.
% Entrada:
%   C: Matriz dos coeficientes de ordem mxn.
%   B: Matriz dos termos independentes de ordem mx1.
% Saída:
%   A: Matriz ampliada de ordem mx(n+1).
%   E: Matriz na forma escalonada reduzida equivalente a A.
% Exemplo:
%   [A,E] = escalonada_reduzida_matriz_ampliada([3,2,-5;2,-4,-2;1,-2,-3],[8;-4;-4])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz C
[m_C,n_C] = size(C);

% Calcula a ordem da matriz I
[m_B,n_B] = size(B);

% Verifica se o número de colunas de C é igual ao número de linhas de B
if m_C == m_B
    % Identifica a matriz ampliada associadao ao sitema linear
    A = [C,B];
    % Determina o posto de A
    pA = rank(A);
    % Encontra a forma escalonada reduzida associada a matriz A
    E = rref(A);
    % Verifica se pode ter ocorrido erro com o uso de rref
    m_absE = rank(E);
    % Verifica se o número de linhas de absE é diferente de pA
    if m_absE ~= pA
        fprintf('Possivel erro associado a tolerancia de rref()!\n');
        E = [];
    end
else
    fprintf('Verifique as ordens as matrizes!!!')
    A = [];
    E = [];
end
```

```
function [E, pA, nA] = forma_escalonada_reduzida_matriz(A)
% FORMA_ESCALONADA_REDUZIDA: Função que retorna a forma escalonda reduzida
% da matriz A.
% Entrada:
%   A: Matriz de ordem mxn.
% Saída:
%   E: Matriz na forma escalonada reduzida equivalente a A, no caso de não
%      ocorrer erro ao usar a função rref().
%   E: [] no caso de ocorrer erro ao usar a função rref().
%   pA: Posto de A.
%   nA: Nulidade de A.
% Exemplo:
%   [E,pA,nA] = forma_escalonada_reduzida_matriz([1,2,-3,0;2,4,-2,2;3,6,-4,3])
%=========================================================================
%Importante: O uso dessa função para encontrar a forma escalonada reduzida em um
% caráter mais acadêmico. Com o uso da função rref()podem ocorrer erros numérios.
% Dependendo da tolerância tol estabelecida para rref(), tem-se resultados
% diferentes.
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Determina o número de linhas e de colunas de A
[m,n] = size(A);

% Determina o posto de A
pA = rank(A);

% Determina a nulidade de A
nA = n - pA;

% Determina a forma escalonada reduzida equivalente a A.
E = rref(A);
% Verifica se pode ter ocorrido erro com o uso de rref
m_absE = rank(E);
% Verifica se o número de linhas de absE é diferente de pA
if m_absE ~= pA
    fprintf('Possivel erro associado a tolerancia de rref()!\n');
    E = [];
end

end
```

```
function [detA,invA] = inversa_matriz(A)
% INVERSA_MATRIZ: Função que calcula a inversa da matriz A, caso exista.
% Entrada:
%   %   A: Matriz A de ordem nxn.
% Saída:
%   detA: Determinante da matriz A.
%   invA:
%       Matriz inversa de A de ordem nxn, caso exista.
%       [] se A não possui inversa.
% Exemplo:
%   [detA,invA] = inversa_matriz([2 3 -2;1 4 8;7 5 3])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================

% Calcula a ordem da matriz A
[m,n] = size(A);

% Verifica a condição para calcular o determinante
if m == n
    % Calcula o determinate da matriz A
    detA = det(A);

    % Devido a erros numéricos não é aconselhável usar o determinante para verificar
    % se uma matriz quadrada A possui inversa. Para essa finalizade deve-se verificar
    % o condicionamento da matriz em questão. Nesse sentido, usa-se aqui a função
    % inv(), a qual retorna um warning no caso de A não ser bem condicionada.

    % Limpa, caso exista, o último lastwarn retornado
    lastwarn('');
    warning('');
    % Tenta calcular a inversa da matriz A
    invA = inv(A);

    % Verifica se retornou algum warning
    [msg, msgid] = lastwarn();

    fprintf('\n\n');
    if (isempty(msg))
      invA = invA;
    else
      fprintf('A matriz não possui inversa!!!\n');
      invA = [];
    end

%    if detA ~= 0
%        % Calcula a inversa da matriz A
%        invA = inv(A);
%    else
%        fprintf('A matriz não possui inversa!!!');
%        invA = [];
%    end

else
    fprintf('A matriz não é quadrada!!!');
    detA = [];
    invA = [];
end


end
```

```
function[C] = produto_matriz_por_escalar(A,k)
% PRODUTO_MATRZ_POR_ESCALAR: Função que retorna a matriz C = k.A.
% Entrada:
%   A: Matriz A de ordem mxn.
%   k: Escalar qualquer
% Saída:
%   C: Matriz C = k.A de ordem mxn.
% Exemplo:
%   [C] = produto_matriz_por_escalar([1,2,3;4,5,6],-3)
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a matriz C
C = k.*A;

end
```

```
function [C] = produto_matrizes(A,B)
% PRODUTO_MATRIZES: Função que retorna a matriz C = A.B
% Entrada:
% A: Matriz de ordem mxn.
% B: Matriz de ordem nxp.
% Saída:
% C: Matriz de ordem nxp.
% Exemplo:
%   [C] = produto_matrizes([1,2,3;4,5,6],[7,7,9,10;11,12,13,14;1,2,3,4])
%==========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%==========================================================================


% Calcula a ordem da matriz A
[m_A,n_A] = size(A);

% Calcula a ordem da matriz B
[m_B,n_B] = size(B);

% Verifica se o produto matricial está definido
if n_A == m_B
    % Calcula a matriz C
    C = A*B;
else
    fprintf('Produto de matrizes não está definido. Verifique as ordens as matrizes!!!\n')
    C = [];
end

end
```

```
function [C] = soma_matrizes(A,B)
% SOMA_MATRIZES: Função que retorna a matriz C = A + B.
% Entrada:
% A: Matriz de ordem mxn.
% B: Matriz de ordem mxn.
% Saída:
% C: Matriz C = A+B de ordem mxn.
% Exemplo:
%   [C] = soma_matrizes([1,2,3;4,5,6],[3,2,1;9,8,7])
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a ordem da matriz A
[m_A,n_A] = size(A);

% Calcula a ordem da matriz B
[m_B,n_B] = size(B);

% Verifica se a soma está definida
if m_A == m_B && n_A == n_B
    % Calcula a matriz C
    C = A + B;
else
    fprintf('Soma de matrizes não está definida. Verifique as ordens das matrizes!!!\n')
    C = [];
end

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-Coordenador)
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
       matrizes!!!\n')
32     C = [];
33 end
34
35 end
```

```
function[C] = transposta_matriz(A)
% TRANSPOSTA_MATRIZ: Função que retorna a matriz matriz transposta de A.
% Entrada:
%   A: Matriz A de ordem mxn.
% Saída:
%   C: Matriz C = A^{T} de ordem nxm.
% Exemplo:
%   [C] = transposta_matriz([1,2,3;4,5,6])
%=========================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%=========================================================================

% Calcula a matriz C
C = A';

end
```

```
function [O] = verifica_autovalor_autovetor_matriz(A,lambda,v)
% VERIFICA_AUTOVALOR_AUTOVETOR_MATRIZ: Função que retorna a forma geral da matriz
% resultante do produto (A - lambda.*I).v.
% Entrada:
%   A: Matriz de ordem nxn.
%   lambda: Escalar qualquer.
%   v: Vetor nx1
% Saída:
%   O: Matriz resultante do produto (A - lambda.*I).v.
% Exemplo:
%   [O] = verifica_autovalor_autovetor_matriz([-3,1,-1;-7,5,-1;-6,6,-2],-2,[-1;0;0])
%
%================================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%================================================================================
% Calcula a ordem da matriz A
[m,n] = size(A);

% Verifica a ordem da matriz
if m == n
    % Identifica a matriz idenidade de ordem nxn
    I = eye(n);
    % Identifica a matriz nula de ordem nx1
    O = zeros(n,1);
    % Identifica a matriz (A - lambda.I)
    C = (A - lambda.*I);
    % Verifica se v é não nulo
    if all(v == 0)
      fprintf('O vetor v não pode ser nulo \n');
      O = [];
    else
      rC = rank(C);
      if (rC == m)
        fprintf('lambda não é um autovalor de A \n');
        O = [];
      else
        % Verifica ordem do vetor v
        [mv,cv] = size(v);
        if (mv == m)
          % Determina a matriz resultante de (A - autoval.*I).v
          O = ((A - lambda.*I)*v);
        else
          fprintf('O vetor v deve ter m linhas \n');
          O = [];
        end
      end
    end
else
    fprintf('A matriz não é quadrada!!!');
end

end
```

```
function [p] = verifica_autovalor_matriz(A,lambda)
% VERIFICA_AUTOVALOR_MATRIZ: Função que retorna o valor p(lambda) onde p(.) é o
% polinômio característico de A
% Entrada:
%   A: Matriz de ordem nxn.
%   lambda: Escalar qualquer.
% Saída:
%   p: Valor dado por p(lambda).
% Exemplo:
%   [p] = verifica_autovalor_matriz([-3,1,-1;-7,5,-1;-6,6,-2],-2)
%
%================================================================================
% Projeto Proae: Elaboração de Material Didático que Empregue o uso de
%   Software como Suporte para o Aprendizado de Álgebra Linear.
% Membros do Projeto:
%   Mônica Aparecida Cruvinel Valadão (Coordenadora)
%   Douglas Frederico Guimarães  Santiago (Vice-Coordenador)
%   Anderson Luiz Pedrosa Porto (Colaborador)
%   Flaviano Luiz Benfica (Bolsista)
%================================================================================
% Calcula a ordem da matriz A
[m,n] = size(A);

% Verifica a ordem da matriz
if m == n
    % Encontra os coeficientes do polinômio característico
    c = poly(A);

    % Retorna os coeficientes considerando o determinante de (A - lambda.*I)
    c = -c;
    c = c(:);

    % Identifica a quantidade de coeficientes
    nc = length(c);

    % Verifica se lambda é solução do polinômio característico
    xlambda = lambda*ones(1,(nc-1));
    e = [1:(nc-1)];
    [ex,~] = sort(e,'descend');
    xlambda_ex = xlambda.^ex;
    xlambda_ex = xlambda_ex(:);
    p = (c(1:(nc - 1)))'*xlambda_ex + c(nc);
else
    fprintf('A matriz não é quadrada!!!');
    p = [];
end



end
```