UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA ELÉTRICA E BIOMÉDICA
PROGRAMA DE EDUCAÇÃO TUTORIAL DE ENGENHARIA ELÉTRICA

MINICURSO DE OCTAVE

Básico

BELÉM - 2022

Sumário

1. Introdução

O Octave é um software de código aberto (Open-Source), muito utilizado para operações matemáticas complexas como cálculo matricial, processamento de sinais, montagem de gráficos 2D e 3D, análise numérica etc. Além disso, é possível adicionar outras funções ao programa por meio de pacotes*, dessa maneira permitindo a utilização de lógica Fuzzy, simulações complexas de fluidos, Arduino, entre outros.

Também muito conhecido por sua sintaxe simples e totalmente compatível com o Matlab, cuja utilização se destaca no mercado, porém fraca academicamente, devido ao preço para adquirir a sua licença. Dessa forma, os conhecimentos do Octave visto nesse minicurso, mesmo que básicos, são essenciais para a iniciação na área científica.

*Olhar: https://packages.octave.org/

2. Como Instalar

Como dito anteriormente, o Octave é um programa de uso gratuito e sua instalação pode ser feita através do site https://octave.org/download .

2.1. Instalação no Windows

Através do link acima você deve selecionar qual versão do Windows é utilizada no computador no qual se deseja realizar a instalação. Após isso se deve escolher qual o formato do arquivo, é recomendável arquivos do tipo executável (.exe).

Em seguida você deve selecionar como local de instalação, a área de trabalho do seu computador. O download do será executado em seguida, e após o seu fim, selecione para executar o arquivo que você acabou de baixar.

Figura 1 – Bem-vindo ao GNU Octave Setup

A seguinte imagem deve aparecer, apenas selecione a opção Next, até aparecer o programa lhe perguntar para quais usuários você deseja instalar o programa, selecione o que for mais adequado e então aperte Next. Então, aperte Install para dar início a instalação do programa, e aguarde até o fim.

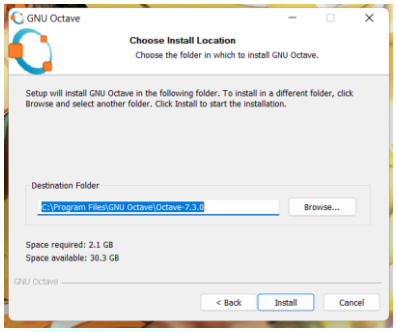

Figura 2 – Escolha a localização de instalação

Com a instalação concluída, mais uma tela aparecerá, então selecione a opção Finish, e perceba que agora existiram 2 versões do Octave instaladas na área de trabalho. A versão que você deseja utilizar para esse curso é GNU Octave (GUI).

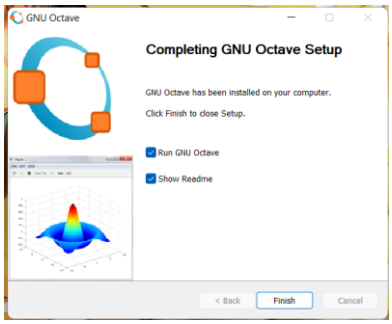

Figura 3 – Terminando a instalação

3. *Interface do programa.*

À primeira vista a interface do GNU Octave parece mais complicada que outros compiladores, mas ao longo desse tópico iremos dissecar o seu ambiente de trabalho em **cinco** áreas mais importantes, que serão vistas a partir do tópico 3.1, por enquanto, existem alguns pontos para se destacar.

A interface possui uma mecânica um tanto quanto diferente dos demais compiladores, pois essa trabalha com um sistema de Acoplar/Desacoplar janelas, de forma a 'personalizar' o ambiente a maneira que o usuário preferir. Acima de cada uma dessas janelas existe uma barra com dois ícones, e ao colocar o mouse em cima de um deles é possível ver o que este representa, sendo um deles para Acoplar/Desacoplar a janela e outro para ocultar a mesma. Além disso na barra é possível enxergar o nome da janela, e ao clicar e segurar o botão esquerdo do mouse também é possível 'arrastar' a janela para uma nova posição, e ao 'soltar' a janela 'em cima' de outra, as duas são agregadas, e somente uma será visualizada por vez, podendo alterná-las através de uma nova barra que surgirá abaixo da janela.

É importante destacar que logo acima do programa existe algo chamado 'Barra de ferramentas', nela existem alguns ícones de botões que quando apertados permitem que você, copie e cole textos que estão na sua área de transferência, desfaça a última alteração feita pelo usuário, além de manter você informado sobre qual o diretório está sendo utilizado para salvar aquele programa.

3.1. Janela de comandos

Essa é a área mais importante do programa, pois nela serão executados todos os comandos que o seu programa utilizar. Diferentemente de muitos compiladores, no Octave, os comandos serão executados em uma janela diferente, linha a linha, isso significa que, todo e qualquer texto que for escrito dentro da janela de comando, deve ser um comando, e cada linha será executada individualmente assim que o usuário apertar 'Enter'.

Em outras palavras, na Janela de Comandos, NÃO entrarão comentários, e cada linha de comando será executada à medida que for escrita. Por esses motivos, não é comum escrever um programa na janela de comandos, e sim no Script/Editor, o qual iremos investigar em seguida.

3.2. Histórico de comandos

Como o nome diz, nessa janela teremos apenas um histórico de comandos utilizados, ou seja, assim que o usuário inserir um comando na Janela de comandos e executá-los, esse mesmo comando aparecerá logo em seguida no histórico. Essa janela é um jeito eficiente de manter uma noção de quais comandos estão sendo executados, e caso ocorra algum erro na sintaxe do código, através dessa janela é possível identificá-lo com facilidade.

3.3. Ambiente de trabalho

Essa janela é muito importante para o programa que está sendo escrito, pois nela é possível visualizar as variáveis que foram criadas até o presente momento, podendo ver também qual o tipo de cada uma e os valores que foram atribuídos as mesmas. Logo, essa janela, de suma importância, nos permite navegar pelo programa criado e explorar as instâncias criadas.

Exercício 1:
-Na janela de comandos, crie as variáveis a e b, com valores da sua mente, em seguida execute o produto de ambas.

Resolução:
1º Passo: Digite na Janela de comandos a=2.5 e aperte Enter, dessa forma você criara a variável 'a' e seu valor será 2,5.
2º Passo: Insira agora b=4 e aperte Enter.
3º Passo: agora escreva a*b e aperte Enter. Note que utilizamos o operador * (asterisco), para simbolizar o produto de duas variáveis

É importante perceber que ao longo do exercício, é exibido no Histórico de Comandos exatamente aquilo que o usuário digitou na Janela de Comandos, ou seja, a=2.5, b=4 e a*b. Além disso, as variáveis criadas, a e b aparecem no Ambiente de Trabalho, bem como seu valor e tipo de variável, e após executado o último comando, surge uma nova

variável ‘ans’ que significa **Resposta** (do inglês, Answer), essa recebe o resultado da operação, a **Resposta** da operação.

3.4. Script/Editor

Conforme dito anteriormente no tópico 3.1, a janela de comandos permite a execução de uma linha de código por vez, logo, essa não é a mais recomendada para que você escreva seu código, por esse motivo, utilizamos o Script/Editor.

Esse recurso do Octave permite que nós escrevamos um código inteiro acrescentando comentários e utilizando várias linhas, e quando finalizamos, podemos executar o código contido no script de uma vez só.

Para começar a utilizá-lo devemos apertar um botão chamado ‘Novo Script’ presente na barra de ferramentas, e imediatamente em seguida, será criado uma nova janela chamada ‘Editor’, aparentemente essa janela parece ter ‘tomado’ o lugar da Janela de Comandos, porém é possível alternar entre as duas através de uma pequena barra abaixo localizada na parte de baixo do editor.

Exercício 2:
-Repita o Exercício 1, agora utilizando Script, e lembre-se de limpar as variáveis e a janela de comando! (Utilize os comandos clear e clc)

Resolução:
1º Passo: Digite no Editor o comando clear, isso excluirá as variáveis já criadas, e ‘limpará’ o Ambiente de Trabalho. Em seguida digite clc para limpar a Janela de Comandos de todos os comandos executados.
2º Passo: Agora digite todos os comandos utilizados no Exercício anterior, linha por linha.
3º Passo: Execute o programa. Isso pode ser feito de duas formas, a primeira, selecione todo o texto com o seu mouse, e aperte F9, ou botão direito do mouse, em seguida aperte em ‘Executar seleção’, a outra forma é apertar no botão de engrenagem que fica na barra de ferramentas diretamente acima do Editor, onde aparecerá a frase “Save File and Run/Continue”.

3.5. Navegador de Arquivos

Essa janela funciona como uma forma de visualizar diretórios no seu computador, permitindo que você navegue entre os documentos e abra um arquivo que está entre eles. Por exemplo, um arquivo de texto contendo um código pronto, no qual você deseja copiar e adicionar ao seu programa.

4. Comandos básicos

4.1. Comandos help, clc, clear, ‘%’, ‘:’ e comando ‘...’.

- help: Digitar help + ‘Comando’, ajuda você a obter mais informações sobre o comando escolhido, por exemplo, sua função, a biblioteca da qual esse faz parte, bem como suas variações;
- clc: ‘Limpa’ a Janela de Comandos, colocando o cursor de volta na sua posição inicial, ajuda para diminuir a poluição visual;
- clear: Exclui todas as variáveis criadas e permite que você consiga recomeçar o programa sem que haja inconsistências com valores antigos de variáveis de mesmo nome, também desocupa memória do computador, ao eliminar variáveis que não estão sendo utilizadas;
- %: Permite que seja feita uma linha de comentários no seu código, essa linha e tudo que houver nela não será executada sendo completamente ignorada pelo programa. É útil para que programadores possam explicar entre si o porquê de algo ter sido feito de tal forma, bem como pode ser usado como anotação de uma informação que você mesmo queira guarda para si;
- ‘:’: O comando : serve para buscar valores em matrizes e os exibe de forma ordenada. Isso ficará mais claro quando chegarmos ao Tópico 6.3 – Indexação de Valores;
- ‘...’: Ao Inserir ‘...’ no final de uma linha de comentário, e no começo da próxima linha, permite que ambas as linhas se tornem uma linha de comentário, sem precisar inserir %. Isso serve para evitar que o comentário seja tão grande que ‘escape’ da tela, dessa maneira você pode manter o seu código mais organizado.

4.2. Operadores matemáticos básicos

Conforme dito no tópico 1, o Octave é muito importante para simplificar processos matemáticos, para começar a entender mais sobre, começaremos aprendendo a montar expressões básicas. Você consegue utilizar os seguintes símbolos como operadores matemáticos para realizar operações matemáticas básicas.

Tabela 1 – Operadores matemáticos básicos

Operação	Operador	Exemplo
Soma	+	5+9
Subtração	-	16-15
Multiplicação	*	24*2
Divisão	/ ou \	64/8 ou 8\64
Potenciação	^	2^5
Igualdade	==	X==5
Diferença	!=	X! =5

Observações:

Note que a mudança de ‘/’ para ‘\’ afeta o resultado. Por exemplo, 10/2 significa “10 sendo dividido por 2”, e seu resultado é 5, porém, 10\2 significa “10 está dividindo 2”, logo o resultado será 0,2. Ou seja, a inversão correta seria 2\10, “2 está dividindo 10”, cujo resultado será 5.

O fato de serem utilizados dois sinais de igual para determinar uma igualdade será abordado quando falarmos sobre o comando *if*. O uso de apenas 1 sinal de ‘=’, diz ao programa que um

valor está sendo atribuído a variável, quando 2 são usados, significa um teste para determinar se a variável possui o valor que está sendo dito.

É importante destacar que existe uma ordem para a resolução das operações, porém, você pode alterar a ordem usando parênteses, assim como em expressões comuns. A ordem de resolução é:

1º. Potenciação
2º. Multiplicação e divisão
3º. Soma e subtração

Exercício 3: Resolva as seguintes operações utilizando o Octave. Lembre-se de utilizar parênteses.

a) $34+2^2 = 38$
b) $(15+5) \cdot (23+17) = 800$
c) $\frac{4^5}{2^4} = 64$
d) $2 \cdot (27+1) = 56$

4.3.Funções Matemáticas

É possível utilizar alguns comandos para calcular o resultado de algumas funções básicas utilizadas na engenharia, algumas funções trigonométricas por exemplo. Em contrapartida com outras linguagens de programação, no Octave, não é necessário a adição de outras bibliotecas, nem a criação de funções recursivas, o que simplifica e muito o uso desse programa.

Tabela 2 – Funções Matemáticas.

COMANDO	DESCRIÇÃO
acos(x)	Arco Cosseno
asin(x)	Arco Seno
atan(x)	Arco Tangente
cos(x)	Cosseno
sin(x)	Seno
tan(x)	Tangente
exp(x)	Exponencial de x
sqrt(x)	Raiz Quadrada

log(x)	Logaritmo
log10(x)	Logaritmo na base 10

Exercício 4: Utilizando a tabela acima, resolva as equações no Octave. Escreva 'pi' para utilizar π, pois as funções trigonométricas aceitam ângulos em **Radianos**.

a) $\text{sen}\left(\frac{\pi}{6}\right)$
b) $\tan\left(\frac{\pi}{4}\right)$
c) $\sqrt{121}$
d) $log_{10}2$
e) e^2

4.4.Transformação Radiano/Graus.

Conforme visto no exercício anterior, o Octave trabalha com valores em radianos, isso não significa que não é possível utilizar valores em Graus. Para realizar essa transformação, utilizamos duas funções:

- rad2deg(x): Recebe o ângulo em **Radianos** e converte para **Graus**.
- deg2rad(x): Recebe o ângulo em **Graus** e converte para **Radianos**.

Exercício 5: Calcule o valor do seno de 19,75º, e converta o ângulo θ = 1.75 rad para graus.

Resolução:

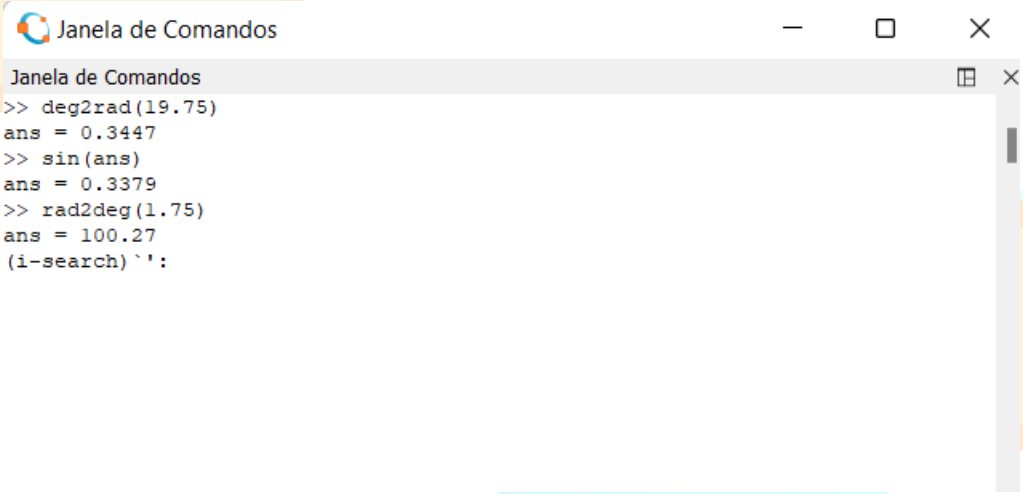

Figura 4 – Resolução do Exercício 5.

Observação:

Note que ans, como já visto antes, é uma variável criada pelo programa para armazenar a resposta, e como essa variável já foi criada, podemos utilizá-la para a operação seguinte com o comando sin(ans), ela então é substituída pelo novo resultado da última operação. Para variáveis com ans, chamamos de **Variáveis Temporárias**.

Uma forma mais direta de se calcular as relações trigonométricas com entrada em graus é acrescentando um ***d*** ao final dos comandos já apresentados na Tabela 2, para lhe ajudar a compreender, a próxima imagem é a resolução do Exercício 5, utilizando esse atributo.

Resolução 2:

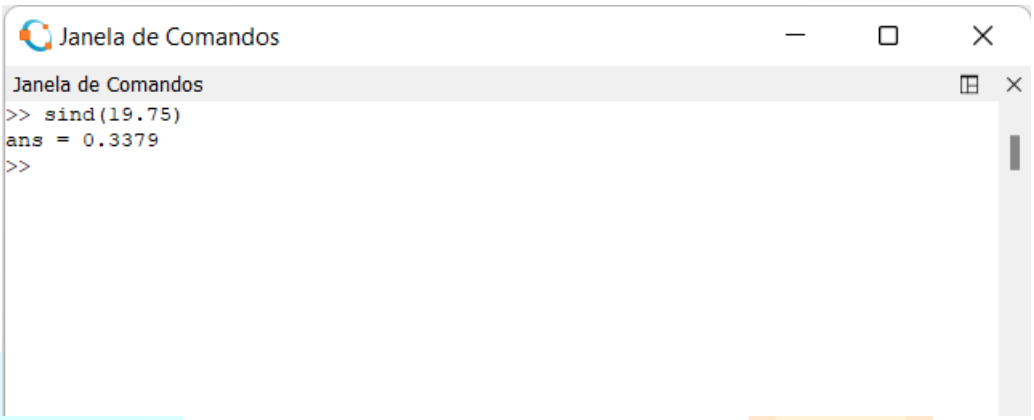

Figura 5 – Resolução alternativa do Exercício 5.

5. *Tipos e Formatações de Variáveis*

5.1. Tipos de Variáveis.

Os tipos de variáveis que serão abordados estão presentes na Tabela a seguir.

Tabela 3 – Tipos de variáveis.

DENOMINAÇÃO	EXEMPLO
Numérica	X = 3
String	nome = 'Tom Cruise'
Complexa	Z = 3+ 4i
Simbólica	syms b
Vetor	V = [1 2 4]
Matriz	M = [1 2 3; 4 5 6; 7 8 9]

Exercício 6: Crie duas variáveis numéricas (a e b) e realize as operações de soma, subtração, divisão e multiplicação. Armazenando cada resultado em uma variável distinta e apresentado os resultados de uma única vez, ao final do programa.

Resolução:

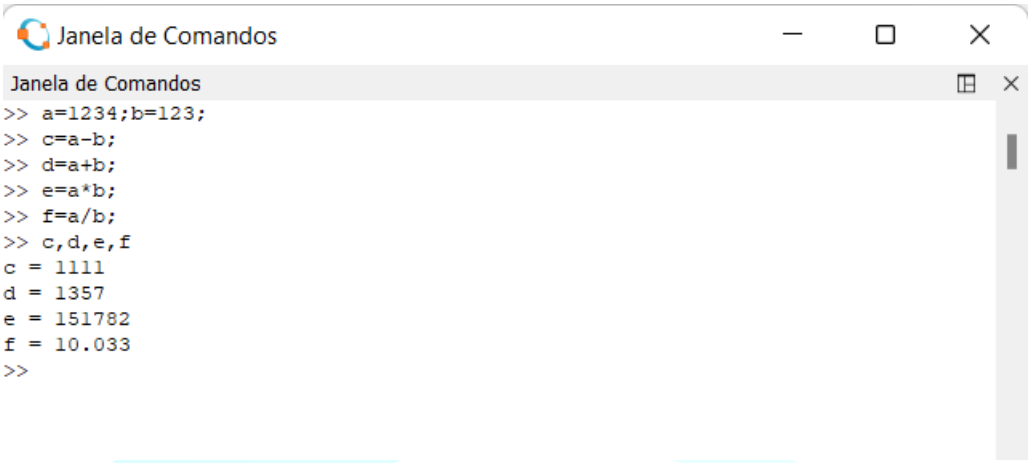

Figura 6 – Resolução do Exercício 6.

Observação:
Note que o uso de Ponto e Vírgula (;) impede que a janela de comando exiba o valor das variáveis após defini-las, é uma forma de evitar a poluição visual.

5.2.Formatação de Variáveis.

O usuário pode optar pelo formato que a variável se apresentará, embora não seja necessário declarar o tipo de variável a ser utilizado, onde se aplicam os comandos mostrados na Tabela a seguir.

Tabela 4 – Formatos de Variáveis.

FORMATO	DESCRIÇÃO	EXEMPLO
format long	**Apresenta o número com precisão de 16 casas decimais.**	pi = 3.141592653589793
format short	**Apresenta o número com precisão de 4 casas decimais.**	pi = 3.1416
format short e	**Notação científica com precisão de até 4 casas decimais.**	pi = 3.1416e+000
format long e	**Notação científica com precisão de até 16 casas decimais.**	pi = 3.141592653589793e+000
format +	**Retorna apenas o sinal da variável.**	Se x = - 2, " - " Se x = 2, " + " Se x = 0, " "
format rat	**Retorna uma razão.**	$e = \frac{1457}{536}$

6. Vetores e Matrizes.

6.1.Vetores.

Vetor é um conjunto de variáveis homogêneas (do mesmo tipo) que são identificadas por um nome. Geralmente são formados por uma linha e 'n' colunas ou uma coluna e 'm' linhas, um vetor pode ser considerado uma matriz de 1 linha ou de 1 coluna.

A primeira forma de se criar um vetor, é elemento por elemento, feito manualmente:

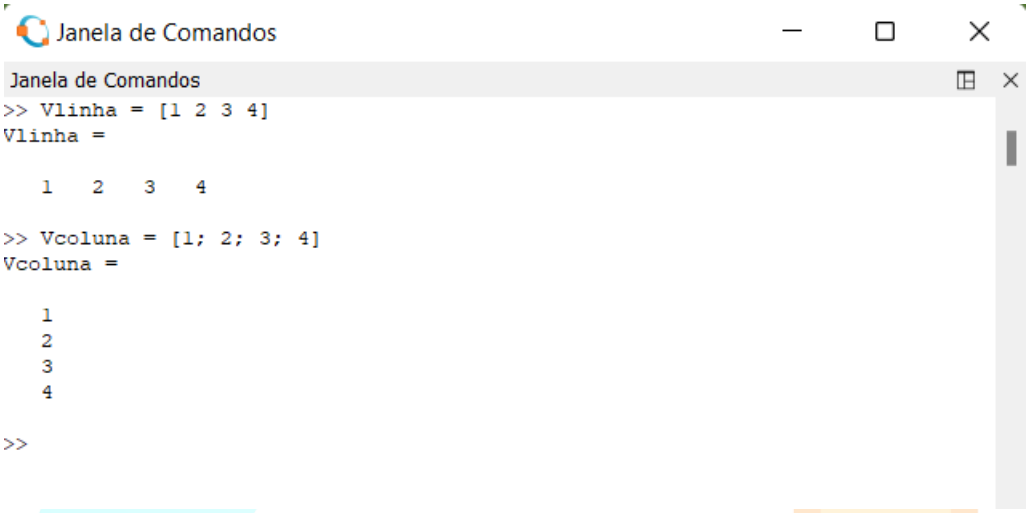

Figura 7 – Primeira forma de se criar um vetor no Octave.

Ou podem ser criados através de comandos básicos, como os comandos exibidos nas tabelas a seguir.

Tabela 5 – Criando Vetores com comandos.

Sintaxe	DESCRIÇÃO	EXEMPLO
linspace(x,y,n)	**Cria um vetor com valor inicial x, valor final y, com n elementos.**	**X = linspace(1,3,4) X= [1; 1,6667; 2,3333; 3]**
X= a:b	**Cria um vetor X com valor inicial a, e valor final b, sendo que o elemento seguinte, é o anterior mais um.**	**X = 1:3 X= [1; 2; 3]**
T= a : i : b	**Cria um vetor T, com valor inicial a, e valor final b, sendo que o elemento seguinte é soma do anterior mais um.**	**X = 0:2:10 X = [0; 2; 4; 6; 8; 10]**

Exercício 7: Crie vetores conforme pedidos:

a) Crie um vetor linha com 10 elementos.
b) Crie um vetor coluna com 4 elementos.
c) Crie um vetor com valor inicial 0, e que possua 20 elementos, cuja razão de progressão seja 1,5.

6.2.Matrizes

Uma matriz pode ser entendida como um conjunto de vetores com o mesmo número de elementos. A nomeação de uma matriz é bem parecida com a nomeação de um vetor, porém, a indicação do final de uma linha e o início de outra é feita por meio do símbolo **';'**, conforme a Figura.

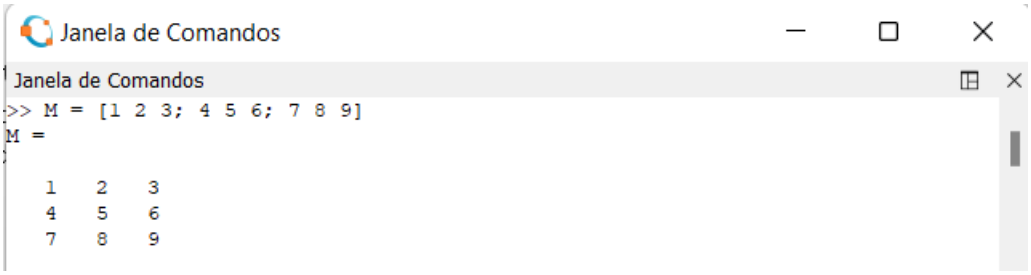

Figura 8 – Exemplo de Matriz.

Também podem ser criados através de comandos, como os exemplificados na tabela a seguir:

Tabela 6 – Comandos para a criação de Matrizes

SINTAXE	DESCRIÇÃO	EXEMPLO
X = ones(m,n)	Cria uma Matriz de **m** linhas e **n** colunas, com todos os elementos iguais a 1.	X = ones(2,2) $X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
X = zeros(m,n)	Cria uma Matriz de **m** linhas e **n** colunas, com todos os elementos iguais a 0.	X = zeros(2,2) $X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
X = rand(m,n)	Cria uma Matriz de **m** linhas e **n** colunas, com todos os elementos aleatórios e menores que um.	X = rand(2,2) $X = \begin{bmatrix} 0{,}023937 & 0{,}476490 \\ 0{,}500259 & 0{,}560220 \end{bmatrix}$
X = eye(n)	Cria uma Matriz identidade de ordem **n**.	X = eye(2) $X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
X = diag([a,b,c])	Cria uma Matriz diagonal, com elementos da diagonal principal iguais a **a**, **b** e **c**.	X =diag([1,2,3]) $X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

Observação:

Note que também é possível criar vetores usando os comando ones(m,n); zeros(m,n) e rand(m,n). Para isso, basta substituir um dos valores de m e n para 1, dessa forma criando um vetor linha, ou vetor coluna.

6.3.Indexação de Valores.

Anteriormente nessa apostila foi dito sobre o operador de indexação ‘:’, agora iremos decorrer sobre sua funcionalidade e quais possibilidades ele oferece.

Para a tabela a seguir, assuma $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Tabela 7 – Uso de indexação.

SINTAXE	DESCRIÇÃO	EXEMPLO
		C = M(:)

M(:)	Dispõem todos os valores em forma de vetor coluna	C = 1 2 3 4 5 6 7 8 9
M(:, a:b)	Dispõem os valores presentes da coluna **a**, até a coluna **b**, em todas as linhas	C = M(:, 2:3) $C=\begin{bmatrix}2 & 3\\5 & 6\\8 & 9\end{bmatrix}$
M(a:b,:)	Dispõem os valores presentes da linha **a**, até a linha **b**, em todas as colunas	C = M(1:2, :) $C=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$

6.4. Operações com Matrizes.

As operações básicas como a soma, subtração e multiplicação são realizadas respeitando as dimensões das matrizes. Isso significa que não é possível efetuar a soma, ou subtração entre matrizes de dimensões diferentes.

- **Soma ou Subtração por escalar:** A soma de uma matriz com um escalar qualquer, nesse caso chamado de 'b', é feita conforme a figura:

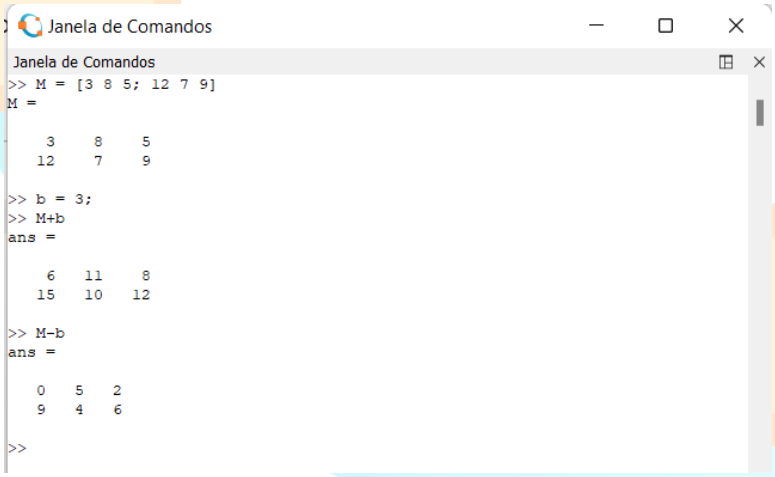

Figura 8 – Soma/Subtração de Matriz por escalar

- **Soma ou Diferença de Matrizes:** A soma entre matrizes é realizada da seguinte forma. Lembrando que para que seja feita a operação, as matrizes devem ter dimensões iguais.

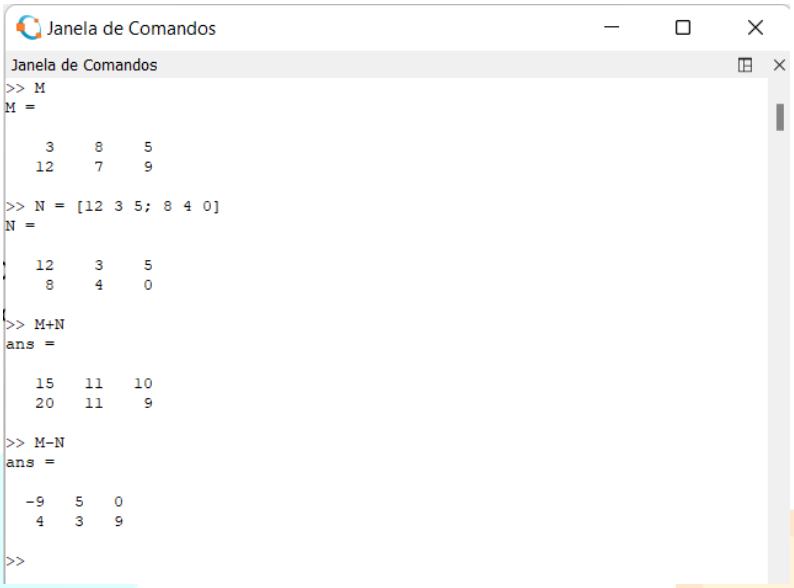

Figura 9 – Soma/Subtração entre matrizes

- **Produto (Multiplicação):** O produto entre matrizes deve obedecer a relação Linha/Colunas, assim como na matemática analítica.

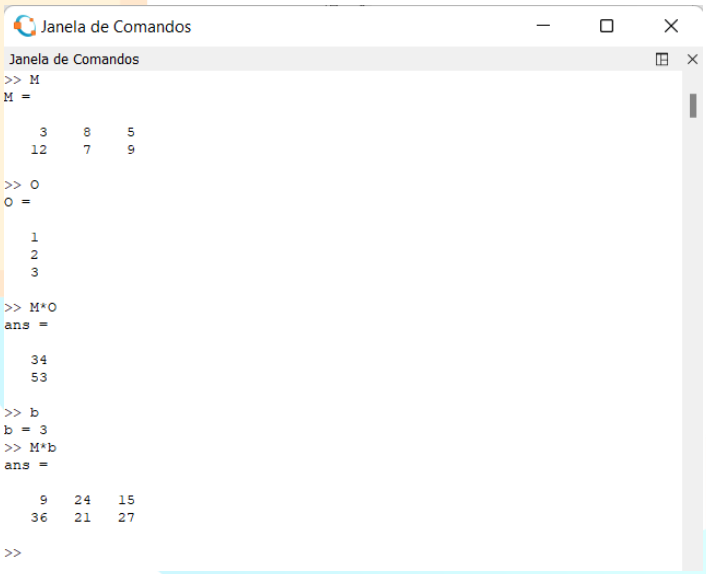

Figura 10 – Produto entre matrizes e escalares

- **Outras Operações:** Algumas outras operações importantes que também são muito conhecidas podem ser vistas na tabela a seguir.

Tabela 8 – Outras operações de matrizes.

SINTAXE	DESCRIÇÃO	EXEMPLO
Inv(M)	Retorna a inversa da Matriz M	$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $M^{-1} = \begin{bmatrix} -2 & 1 \\ 1{,}5 & -0{,}5 \end{bmatrix}$
M’	Calcula a transposta de M	$M' = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
det(M)	Calcula o determinante de M	det(M) = -2

trace(M)	Calcula a soma dos elementos da diagonal principal de M	trace(M) = 5
sum(M)	Soma todos os elementos presentes nas colunas, e retorna uma matriz linha	sum(M) = $[4 \quad 6]$

Exercício 8: Crie as seguintes Matrizes:

A = $\begin{bmatrix}1 & 2\\ 3 & 0\end{bmatrix}$ $B = \begin{bmatrix}7 & 4\\ 2 & 1\end{bmatrix}$ $C = [1 \quad 2 \quad 4]$ D = $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$ $E =$

$$\begin{bmatrix} 1{,}4 & 2{,}5 & 3 & 0\\ 4{,}5 & 5{,}6 & 9{,}6 & 2\\ 11{,}3 & 0 & 1 & 4\\ 1 & 4 & 5 & 7{,}4\end{bmatrix}$$

E Calcule:

a) CxD
b) Determinante de E
c) A^2
d) B^{-1}
e) Transposta de D
f) Determinante de E-(6 x E^{-1})
g) O determinante de E somado com a diagonal principal de A

Resolução

a)$C * D = [37 \quad 44 \quad 51]$ b) det(E) = 597,77 c)A^2 = $\begin{bmatrix}7 & 2\\ 3 & 6\end{bmatrix}$ d) inv(B) =$\begin{bmatrix}-1 & 4\\ 2 & -7\end{bmatrix}$

e) $D' = \begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$ f) det(E-6*inv(E))=665,14 g) trace(A) + det(E) = 598,77

6.5.Concatenação de Matrizes

O verbo *'Concatenar'* significa agrupar, ou juntar, numa sequência lógica. No caso de matrizes, isso significa que nós iremos agrupar uma matriz dentro de outra, por exemplo: Existindo uma Matriz $A = [1 \quad 2]$ e uma Matriz $B = [3 \quad 4]$, então é possível criar $C = \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$, utilizando os seguintes comandos.

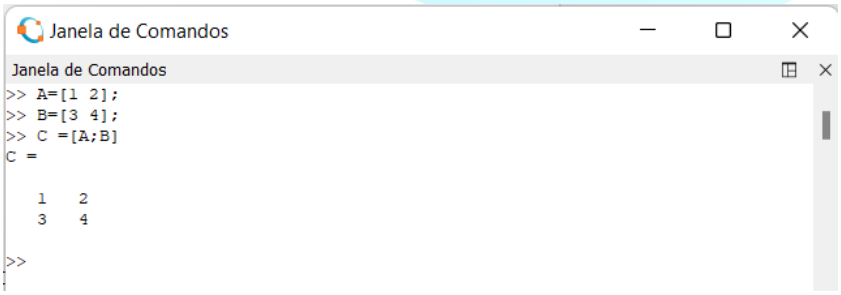

Figura 11 – Concatenação de Matrizes

Exercício 9: Dada as Matrizes:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 \\ 2 & 3 & 4 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \quad C = [1 \quad 2 \quad 3] \qquad D = [4 \quad 5]$$

Utilize concatenação para obter as matrizes:

a) Obtenha $E = [1 \quad 2 \quad 3 \quad 4 \quad 5]$

b) Obtenha $F = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 6 & 7 & 1 & 1 \\ 2 & 3 & 0 & 0 \end{bmatrix}$

Resolução:

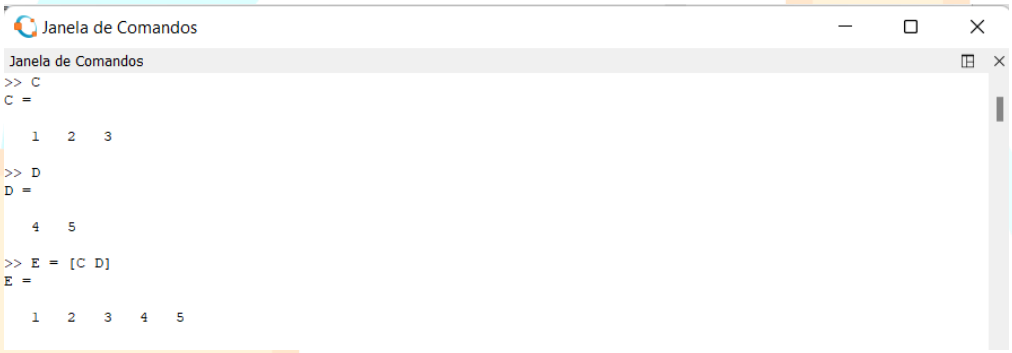

Figura 12 – Resolução da letra a)

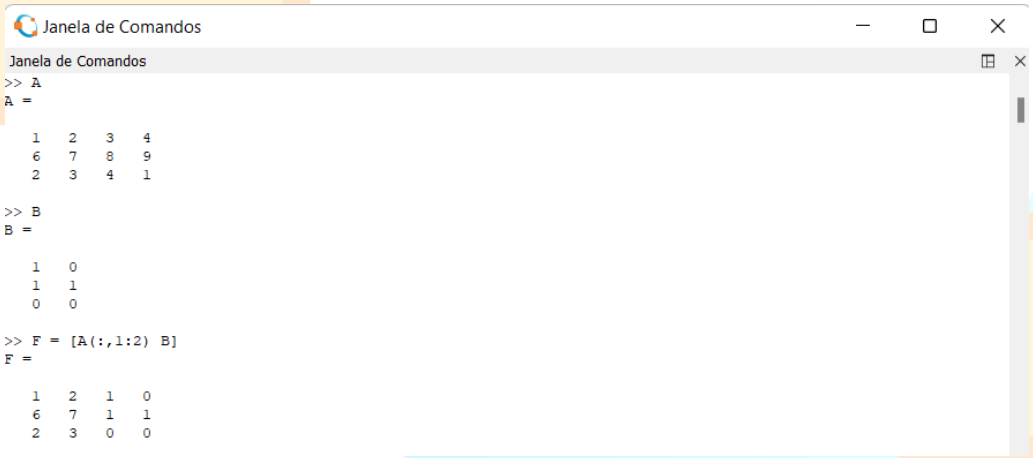

Figura 13 – Resolução da letra b)

7. Criando Scripts/Editor.

A partir deste momento, os programas que você irá criar ficarão um pouco mais complicados, por isso é necessário aprender sobre essa ferramenta muito importante, e podem ser utilizados da seguinte forma.

A primeira forma consiste em utilizá-los como se fossem blocos de notas, o que facilita em muito o trabalho de um programador, pois visto que no Octave é preciso utilizar a Janela de comandos para executar o código, caso seja cometido algum erro durante a criação do código, todo o programa que já foi inserido precisa ser reescrito na janela de comando, além disso também é necessário utilizar o comando clear, para limpar as variáveis.

A segunda forma é como programas executáveis. Ao criarmos programas muito complexos, é comum segmentarmos algumas partes desse programa e isolá-las do código principal, transformando ela em arquivo que pode ser chamado na janela de comando, e então todo código que foi escrito no Script será executado.

Para criar um Script, precisamos utilizar a barra de ferramentas para encontrar o botão 'Novo Script', e então uma nova janela chamada 'Editor' irá aparecer. Ao terminar de escrever um código no Script, podemos executar ele de duas formas, ou selecionando o código que se deseja executar e apertando o botão F9, note que ao fazer isso o código selecionado irá aparecer na janela de comando, ou salvando o arquivo e o chamando por nome na janela de comandos.

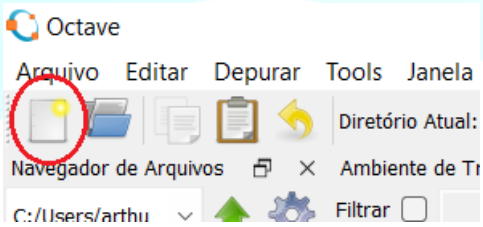

Figura 14 – Botão Novo Script

8. *Análise Polinomial*

8.1. Nomeação de Polinômios.

Um dos muitos usos do Octave, é para o cálculo de funções, de forma a prever valores possíveis, ou eventualmente, exibir gráficos.

Se o desejado for prever o valor de uma função *f(x)* para um valor qualquer de x, então primeiramente é necessário determinar qual o valor x desejado, e em seguida definir o polinômio *f(x)*. É preciso seguir essa ordem para que o programa entenda que existe uma variável x já definida, do contrário, caso o programa não encontre a variável citada, a função retornará uma mensagem de erro.

Para os exemplos a seguir, suponha que se quer trabalhar com a função $f(x) = 4x^2 + 3x + 5$.

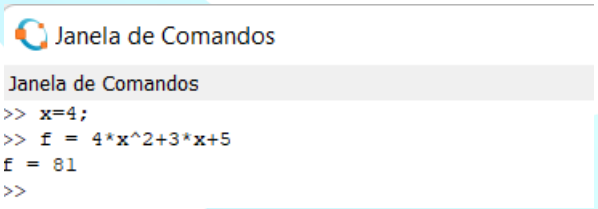

Figura 15 – Determinação do valor do polinômio, para x escalar.

Caso x for um conjunto de valores, o domínio da função, é necessário utilizar '.' em seguida a utilizar o x, de forma a evitar que o programa veja a operação como produto matricial. Como no exemplo:

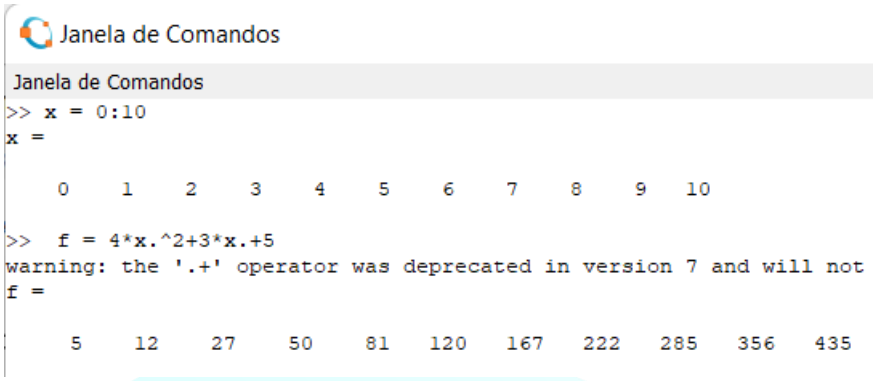

Figura 16 – Determinação da Imagem da função, utilizando um domínio de x.

8.2. Função 'poly'

A função 'poly' pode ser utilizada para determinar os coeficientes de um polinômio, sabendo apenas de suas raízes. Por exemplo, se possuímos uma matriz R com valores que temos conhecimento de serem as raízes de um polinômio *f(x)*, então utilizamos poly(R), e a função retornará os valores dos coeficientes de *f(x)*, dessa forma podemos determinar qual o grau desse polinômio e montar uma função para esse.

Suponhamos que se deseja encontrar a função *g(x)*, da qual sabemos que suas raízes são um conjunto $R = [-1 \quad 1]$, dessa forma executamos o seguinte processo:

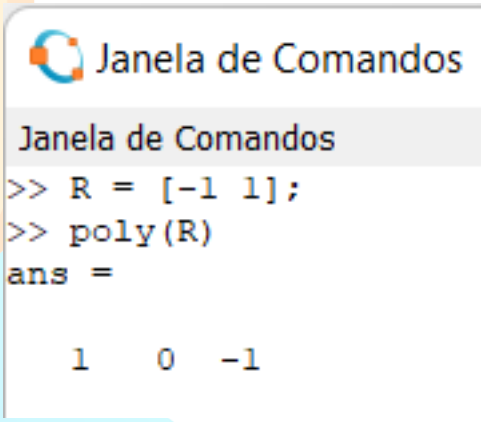

Figura 17 – Uso de poly para determinar coeficientes.

Note que a função nos retorna um vetor linha, o tamanho desse vetor varia de acordo com o número de raízes. Cada valor corresponde ao coeficiente de um dos membros do polinômio, sendo o primeiro elemento da esquerda o coeficiente do membro de maior ordem. Nesse caso, a função que possui raízes $R = [-1 \quad 1]$ é uma função de segundo grau, sendo o primeiro elemento da esquerda correspondente ao coeficiente do termo ax^2, em seguida o de bx, e por último, o termo independente.

Logo, determinamos que o polinômio buscado foi a função $f(x) = 1x^2 - 1$, e para podermos confirmar isso, bastar usar os conhecimentos adquiridos no tópico passado para aplicar os valores de R a função *f(x)*, e ambos os valores retornados devem ser iguais a 0.

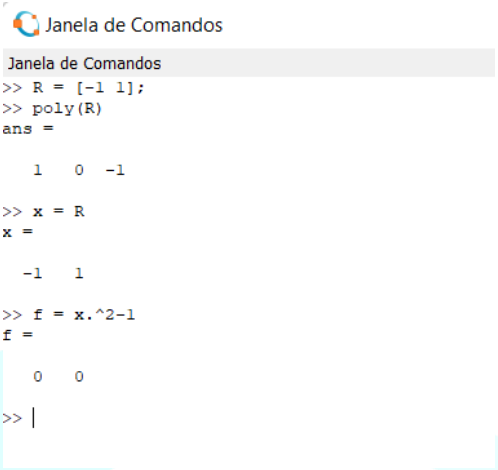

Figura 18 – Determinação da função *f(x)*

8.3.Função roots

O objetivo dessa função é determinar as raízes de um polinômio, utilizando apenas os seus coeficientes. Se usarmos como exemplo o polinômio $g(x) = x^2 + 2x - 15$, para determinar os valores no qual *g(x)* é igual a zero, ou seja, suas raízes, precisamos criar uma matriz $G = [1 \quad 2 \quad -15]$ que contém os coeficientes de *g(x)*, dessa forma, utilizamos roots(G), para obter suas raízes.

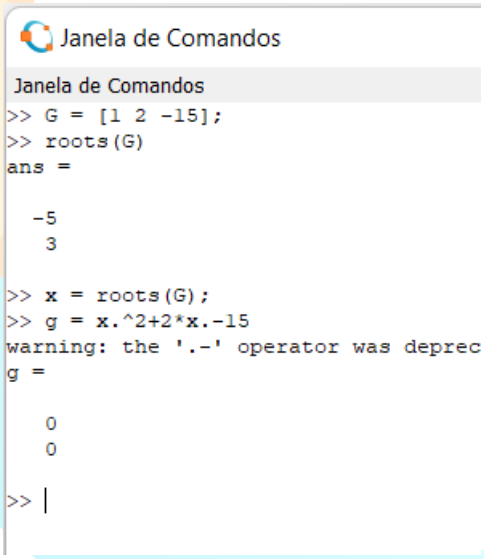

Figura 19 – Determinação das raízes de *g(x)*

Exercício 10: Determine os polinômios cujas raízes são:

a) $A = [1 \quad 2]$
b) $B = [3 \quad 4 \quad 0]$
c) $C = [1 + i \quad 1 - i]$
d) $D = [0 \quad 8 \quad 1]$

Resolução:

a) $f(x) = x^2 - 3x + 2$ b) $f(x) = x^3 - 7x^2 + 12x$ c) $f(x) = x^2 - 2x + 2$
d) $f(x) = x^3 - 9x^2 + 8x$

Exercício 11: Determine as raízes dos seguintes polinômios.

a) $A(x) = x^3 + 2x^2 + x + 2$
b) $B(x) = x^2 + 2x + 1$
c) $C(x) = x^3 + x + 1$
d) $D(x) = x^5 + x + 2$

8.4.Operações matemáticas com polinômios

Para este tópico, iremos tratar polinômios como matrizes do tipo $P = [C_1 \quad C_2 \quad C_3 \quad \dots \quad C_n]$, onde cada elemento da matriz é o coeficiente de um dos membros do polinômio, começando em C_1, sendo o coeficiente do membro de maior grau, e C_n o termo independente.

- **Soma e Subtração:** Funciona da mesma forma que uma soma e subtração de vetores, de forma que o vetor obtido ao final da operação contém os coeficientes do polinômio resultante.

Exercício 12: Efetue a soma dos polinômios $f(x) = x^3 + 2x^2 - 1$ e $g(x) = x^3 + x^2 + 2x - 1$

Resolução:

1º Passo: Retirar os coeficientes e construir as matrizes que representam os polinômios: $F = [1 \quad 2 \quad 0 \quad -1]$ e $G = [1 \quad 1 \quad 2 \quad -1]$, note que o maior grau é 3, portanto, o vetor deve possuir 4 membros.

2º Passo: Realizar a soma de matrizes.

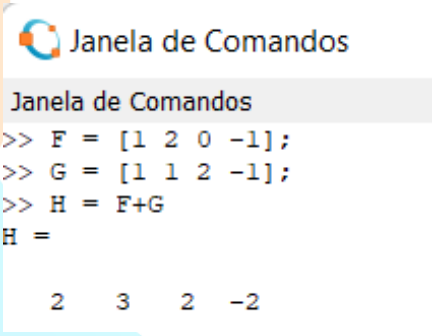

Figura 20 – Soma de Polinômios.

3º Passo: Escrever o polinômio, utilizando os coeficientes obtidos. $h(x) = 2x^3 + 3x^2 + 2x - 2$

- **Produto e Divisão:** Para este tópico, utilizaremos duas funções novas, lembrando que o produto entre dois polinômios é chamado de convolução, para o produto utilizaremos conv(F,G), e para divisão, deconv(F,G), sendo F e G dois vetores com os coeficientes dos polinômios quaisquer f(x) e g(x).

Utilizando como exemplo as funções *f(x)* e *g(x)* do Exercício 12, caso desejássemos obter as funções *y(x)* e *z(x)* através do produto e divisão de ambas, os comandos precisariam ser inseridos da seguinte forma:

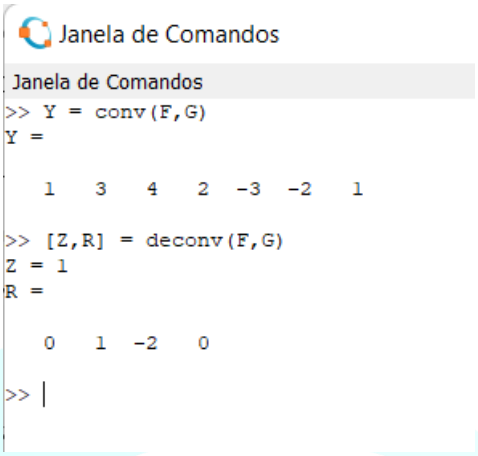

Figura 21 – Comandos conv e deconv

Onde, Y é a matriz que contém os coeficientes do polinômio *y(x)*, Z contém os coeficientes do quociente polinomial e R contém o resto do polinômio de menor ordem.

- **Avaliação Polinomial:** Além da maneira já vista anteriormente para determinar o resultado da expressão para um determinado valor de x, o processo para avaliar um polinômio pode ser feito utilizando o comando polyval(F,x), onde F é a matriz contendo os coeficientes do polinômio qualquer *f(x)* e 'x' é o valor(es) no qual se deseja avaliar a expressão, esse comando retorna o vetor cujos elementos correspondem ao resultado da expressão para o valor de x na mesma coluna do vetor 'x'.

Exercício 13: Avalia o polinômio $f(x) = 5x^2 + 2x + 7$, para o intervalo $x = [0 \quad 1 \quad 2 \quad 3]$ utilizando a função **polyval**, verifique se os valores estão corretos.

Resolução:

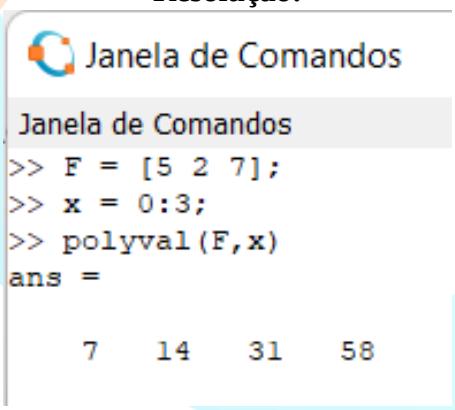

Figura 22 – Utilização da função polyval.

9. *Cálculo Diferencial e Integral.*

9.1. Limite

Para calcular limites, primeiramente é necessário entender o que são variáveis simbólicas. Anteriormente tratamos funções como puramente polinômios e expressões, onde avaliaríamos para valores de x, sendo 'x' uma matriz contendo valores exatos e definidos.

Porém, para o cálculo diferencial, funções e variáveis são tratados como símbolos, conceitos, ao invés de valores duramente definidos, ou seja, ao longo do tópico 8, iremos tratar 'x' como um conceito de variável que **pode** assumir valores diferentes, ao invés de um conjunto de valores definidos.

Antes de prosseguir, utilizaremos o pacote *'symbolic'* do Octave, para isso, você precisará inserir na janela de comandos *'pkg load symbolic'*, isso é necessário, pois, esse pacote não é previamente carregado pelo programa.

Primeiramente, definimos a variável que utilizaremos utilizando o comando syms, nesse caso, utilizaremos 'x', logo, o comando inserido, seria *syms x*. Feito isso, poderemos a partir de agora, criar funções utilizando 'x' sem nos preocuparmos em definir um valor real para esse. Em seguida, para calcular um limite na forma $\lim_{x\to a} f(x)$ utilizamos *limit(f,x,a)*.

Exercício 14: Calcule $\lim_{x\to 2} f(x)$, sendo $f(x) = x^2 - 3x + 1$.

Resolução:

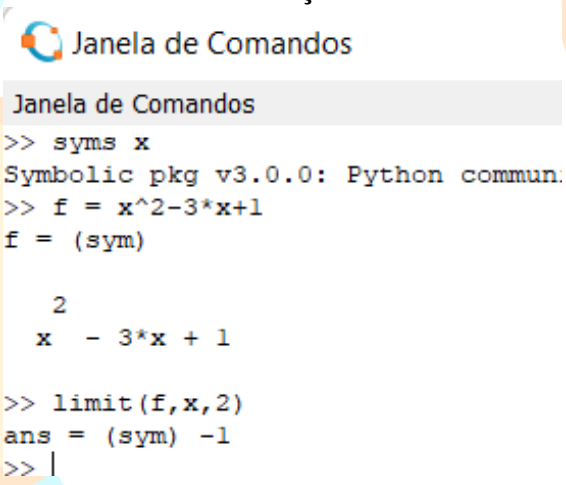

Figura 23 – Limite da função.

Para calcular **limites laterais**, ou seja, determinar o valor da função quando o limite se aproxima por uma direção específica, basta inserir a direção utilizando *'left'*, quando se aproxima pela esquerda, *'right'*, quando pela direita, ou seja, o comando ficará *limit(f,x,a,'direção')*.

Exercício 15: Calcule os limites laterais da função $g(x) = \frac{1}{x}$, quando x se aproxima de 0.

Resolução:

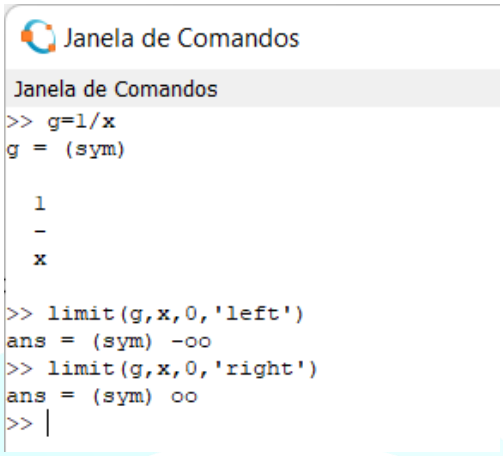

Figura 24 – Limites Laterais.

9.2. Derivada

Tendo já definido como simbólica a variável que será utilizada, uma forma de determinar a sua derivada, é através do comando *diff(função)*. Caso, você esteja derivando uma função de várias variáveis, basta inserir *diff(função, variável)*, para obter a derivada parcial da função, em relação a variável inserida.

Exercício 16: Calcule

a) $\frac{d}{dx}(x^3 + 5x^2 + 8x + 10)$

b) $\frac{d}{du}\left(\frac{1}{u}\right)$

c) $\frac{\partial}{\partial x}(x^2 + xy + y^2 - 2)$

d) $\frac{\partial}{\partial y}(x^2 + xy + y^2 - 2)$

Resolução:

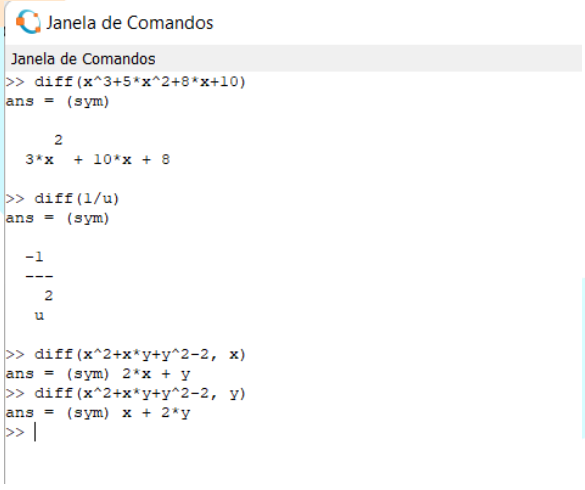

Figura 25 – Derivadas.

Caso deseje-se derivar uma função na forma polinomial, isto é, $P = [C_1 \quad C_2 \quad C_3 \quad \dots \quad C_n]$, o comando a ser utilizado é *polyder(P)*, e ela retornará os coeficientes do polinômio que representa a derivada da função anterior.

9.3.Integral

Aqui utilizaremos o comando *int(função)*, para calcular a integral indefinida, e *int(função, a, b)* para integrais definidas, onde a e b são os limites de integração, inferior e superior, respectivamente. Para integrais de funções com mais de uma variável, usamos *int(função, variável)* e *int(função,variável,a,b)*.

Exercício 17: Calcule

a) $\int 3x^2 + 8x + 2\,dx$

b) $\int_0^2 3x^2 + 8x + 2\,dx$

c) $\int\int 2xy + 9x^2 + 2y\,dx\,dy$

Resolução:

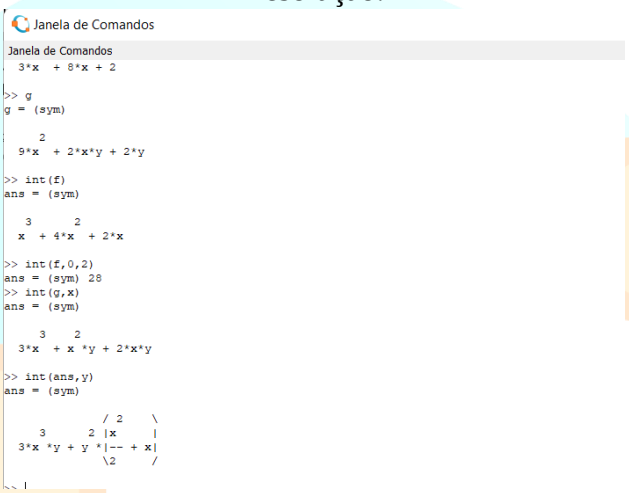

Figura 26 – Resolução do Exercício 17.

10. Trabalhando com gráficos.

10.1. Comando plot.

O principal comando na hora de exibir gráficos, *plot*, trabalha com valores reais, e não simbólicos, dessa forma, para que seja possível utilizá-lo são necessárias duas etapas. Primeiramente, a declaração da variável a ser utilizada, lembrando que ela não será uma variável simbólica, e sim um vetor com os valores que abrangem o domínio da função que iremos trabalhar. Segundamente, será definida a função a ser plotada, e que também irá operar dentro do domínio da variável.

Tendo realizado esses passos, o comando irá funcionar da seguinte forma, *plot(variável,função)*, também é possível utilizar o comando na forma *plot(função)*, caso ambas, variável e função, já tiverem sido declaradas.

Note que a função sempre retornará um conjunto de valores organizados na forma de vetor, ou seja, o comando *plot*, organiza esses pontos no gráfico e os conecta. Portanto, é possível realizar esse comando utilizando apenas um vetor.

Exercício 18: Plote o gráfico da função $\mathrm{sen}(2x)$, no intervalo $x = [-\pi, \pi]$ e taxa de amostragem de 0.1. Faça isso de ambas as formas citadas acima.

Resolução:

```
Janela de Comandos
>> x=[-pi:0.1:pi];
>> f=sin(2*x);
>> plot(x,f)
>> clear
>> plot(x=-pi:0.1:pi,sin(2*x))
>>
```

Figura 27 – Comandos *plot*.

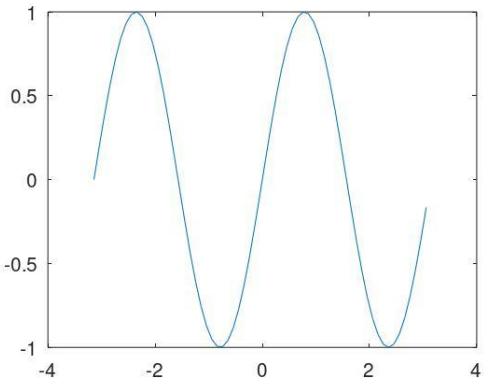

Figura 28 – Gráfico do Exercício 18

10.2. Comando stem.

Esse comando funciona de forma semelhante a ao *plot*, porém, dessa vez não será exibida uma função, e sim uma sequência de sinal discreto. Isso significa, que o comando organiza esses valores, e não os conecta, formando um conjunto de pontos organizados nos instantes do domínio.

Exercício 19: Plote o gráfico da função $f(x) = -x^2 + 25$ na forma de sinal discreto, utilizando o intervalo $x = [-5,5]$ com amostragem de 0.5.

Resolução:

```
Janela de Comandos
>> x=-5:0.5:5;
>> f=-x.^2+25;
>> stem(x,f)
>>
```

Figura 29 – Comando *stem*.

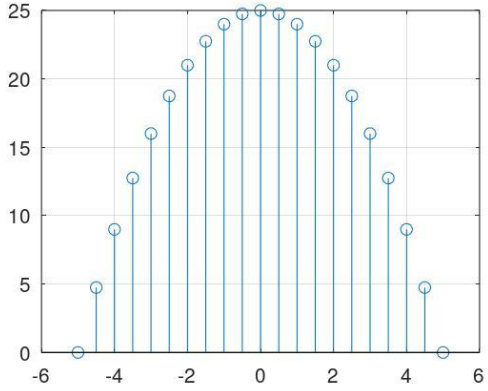

Figura 30 – Gráfico do Exercício 19.

10.3. Outros comandos de gráficos.

- **Comando *bar*:** Consegue plotar gráficos em barra, utilizando vetores com valores predefinidos, na forma, *bar(vetor1, vetor2)*, onde o *vetor1* será a base, e o *vetor2*, o quantitativo.
- **Comando *stairs*:** Exibe gráficos no formato de escadas, e é utilizado da forma *stairs(vetor1, vetor2)*, onde o *vetor1* representa os valores no eixo das abcissas e o *vetor2*, o valor no eixo das ordenadas.
- **Comando *pie*:** Monta o gráfico em formato de pizza, e seu comando é *pie(Vetor)*. Nesse comando, não é necessário mais de um vetor, pois o programa irá gerar um gráfico utilizando como valores os componentes do *vetor*, desde que a soma de todos resultem em 100%.
- **Comando *compass*:** Forma um diagrama polar, e exibe vetores dentro dele, funciona da maneira *compass(x,y)*. Nesse comando cada vetor será representado por uma seta, onde o seu início é a origem do gráfico, e seu fim é as coordenadas (x,y), também pode ser inserido dois vetores, ao invés de x e y, onde cada elemento será uma coordenada de x ou y. Também, é possível utilizar da maneira *compass(z)*, onde z é um número complexo qualquer, o gráfico exibido será a forma polar de z.
- **Comando *loglog*:** Produz um gráfico 2D, com escalas logarítmicas para ambos os eixos. Funciona na forma *loglog(vetor1,vetor2)*.
- **Comando *semilogx* e *semilogy*:** Produz um gráfico 2D com escalas logarítmicas em relação ao eixo x, e ao eixo y, respectivamente. Funcionam na forma *semilogx(vetor1,vetor2)* e *semilogy(vetor1,vetor2)*.

É importante destacar que nos exercícios a seguir, você terá que plotar diversos gráficos, e ao pedir para o programa exibir dois gráficos em seguida, o 2º irá substituir o 1º. Para evitar isso, é utilizado o comando *figure(número)*, que gera uma janela extra para exibir gráficos, cada janela irá ser chamada pelo número que foi inserido.

Exercício 20: Suponha que exista uma pequena empresa que monitore acidentes de trabalho ao longo dos anos 2019, 2020, 2021 e 2022, e durante um levantamento foi constatado que nesses respectivos anos ocorreram 12, 20, 9 e 17 acidentes. Monte o gráfico em barras relacionando o ano e os acidentes.

Exercício 21: Num determinado parque, o aluguel das bicicletas escala conforme a hora de uso pelo cliente, custando 20, 25, 35 e 40, para a 1ª, 2ª, 3ª e 4ª hora. Monte o gráfico em escada que relacione o preço com a hora de uso.

Exercício 22: Mostre no gráfico o número complexo $z = 2 + i$.

Exercício 23: Demonstre o gráfico da função $f(x) = 2e^{(2x)}$, como x no intervalo de [0,10], com amostragem de 0.1, utilizando *loglog*, *semilogx* e *semilogy*.

Resolução:

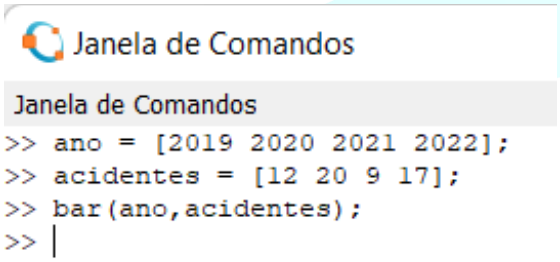

Figura 31 – Comando *bar*

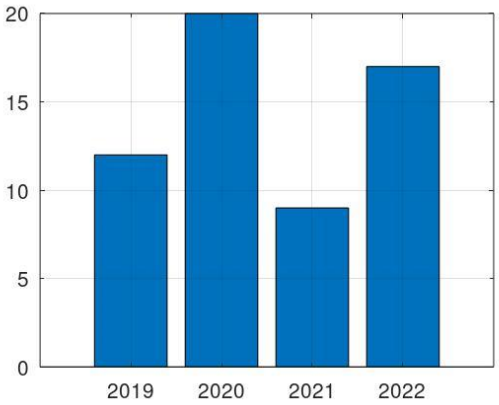

Figura 32 – Gráfico do exercício 20.

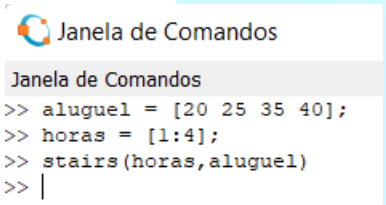

Figura 33 – Comando *stairs*.

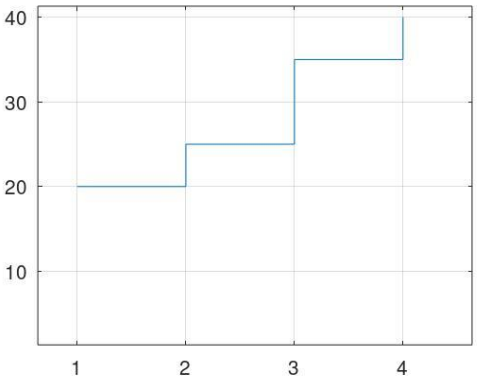

Figura 34 – Gráfico do Exercício 21.

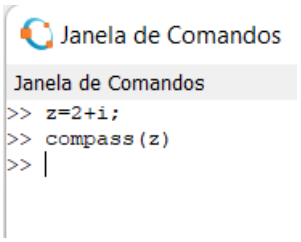

Figura 35 – Comando *compass*.

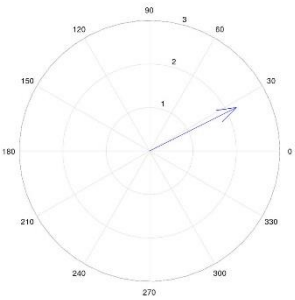

Figura 36 -Gráfico do Exercício 22.

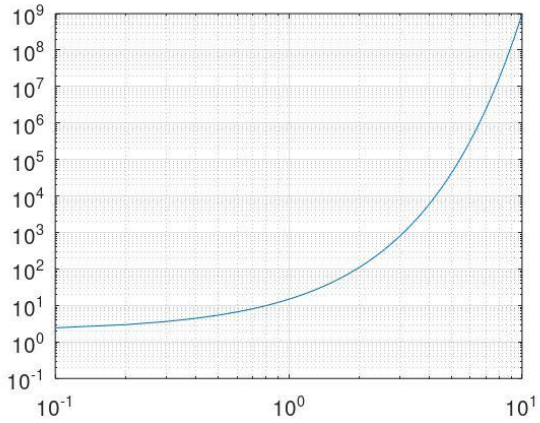

Figura 37 – Gráfico *loglog*.

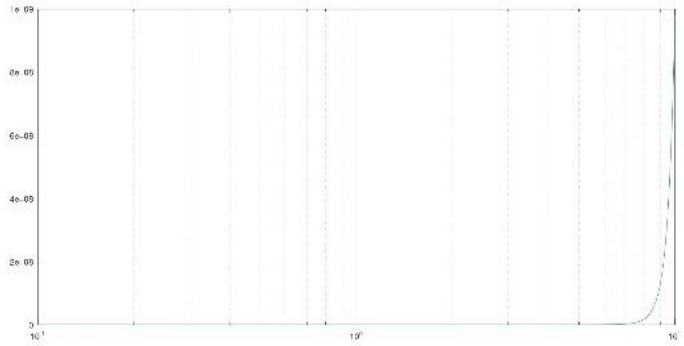

Figura 38 –Gráfico *semilogx*.

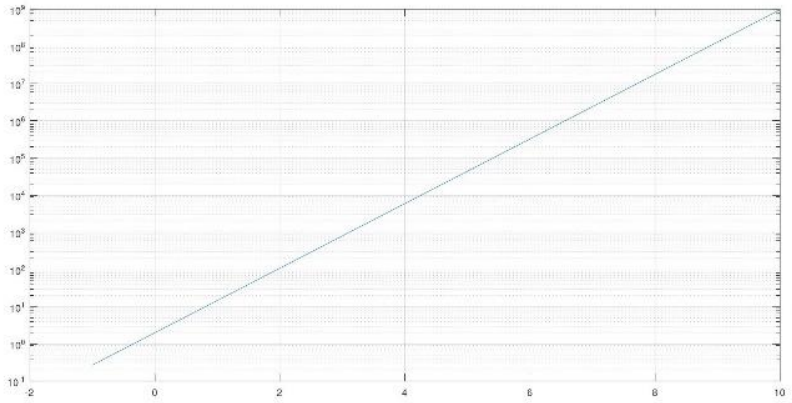

Figura 39 – Gráfico *semilogy*.

10.4. Personalizando os gráficos.

- **Exibindo mais de uma função no gráfico**: É possível que mais funções possam ser exibidas no mesmo gráfico, a fim de comparação visual, para isso basta inserir o comando *plot*(x_1,f_1,...,x_n,f_n), onde f_1 está em função de x_1, da mesma forma que f_n está

em função de x_n. Caso, todas as funções estejam relacionadas a mesma variável, ainda será necessário especificar na forma *plot(x, f_1, x, f_2, ..., x, f_n)*.

Exercício 24: Um transformador opera com fases que obedecem às equações: $Va(t) = 10sen(120\pi)$, $Vb(t) = 10sen\left(120\pi - \frac{2\pi}{3}\right)$ e $Vc(t) = 10sen\left(120\pi + \frac{2\pi}{3}\right)$. Monte o gráfico em relação a t no intervalo [0, 0.1] com amostragem de 0.0001.

Resolução:

```
Janela de Comandos
>> t=0:0.0001:0.1;
>> va=10*sin(120*pi*t);
>> vb=10*sin(120*pi*t-2*pi/3);
>> vc=10*sin(120*pi*t+2*pi/3);
>> plot(t,va,t,vb,t,vc)
>>
```

Figura 40 – Comandos do Exercício 24.

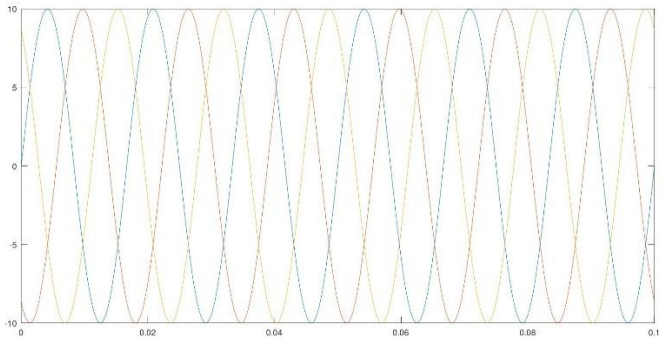

Figura 41 – Gráfico do Exercício 24.

- **Exibindo mais de um gráfico na mesma janela:** Para isso, é utilizado o comando *subplot(linhas, colunas, posição)*. Ao inserir esse comando, o programa automaticamente cria uma janela com uma grade de com o mesmo número e linhas que foi inserido, dessa forma. As posições são ditas da seguinte forma, elas são preenchidas por linha, de forma crescente começando pela esquerda, por exemplo, se fosse inserido o comando *subplot(2,2,1)*, isso indica que existe uma grade de 2 linhas e 2 colunas e que o gráfico está na 1ª posição, ou seja no canto superior esquerdo.

Exercício 25: Utilize o comando *subplot* para exibir os 3 seguintes gráficos na mesma janela: $f(x) = \log(x)$, $g(x) = x^2$ e $h(x) = x^3$.

Resolução:

```
Janela de Comandos
>> x=-10:10;

>> f=log(x);

>> g=x.^2;

>> h=x.^3;

>> plot(x,f,subplot(2,2,1))
>> plot(x,g,subplot(2,2,2))
>> plot(x,h,subplot(2,2,3))
>>
```

Figura 42 – Comandos do exercício 25.

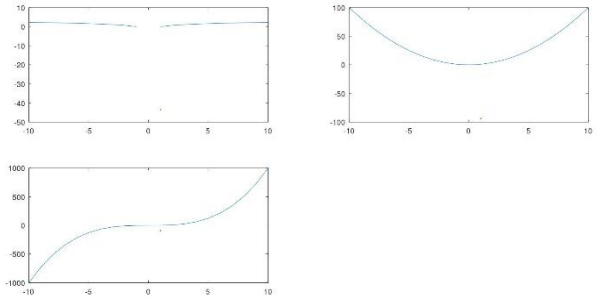

Figura 43 – Gráficos do Exercício 25.

- **Cor e formato dos gráficos:** É possível alterar características como qual tipo de linha será usada para desenhar a função e qual cor ela irá ter, para isso, essas cada um desses aspectos possui um código que deverá ser inserido entre ‘ ’, após definida a variável e a função, do jeito *plot(x,f,’característica’)*, todas os sinais devem ser acrescentados juntamente, por exemplo, se você quiser exibir uma função *f(x)*, com linhas tracejadas e amarelas, basta escrever *plot(x,f,’-.y’).* Mais comandos podem ser vistos na tabela abaixo.

Tabela 9 – Personalização de gráfico.

Código	Descrição	Código	Descrição
‘+’	Os pontos são demarcados por ‘+’	‘h’	Os pontos são demarcados por hexagramas
‘o’	Os pontos são demarcados por círculos	‘-‘	As linhas da função são contínuas(padrão)
‘*’	Os pontos são demarcados por sinais de estrelas	‘--'	As linhas da função são tracejadas
‘.’	Os pontos são demarcados por ‘.’	‘:’	As linhas da função são pontilhadas
‘x’	Os pontos são demarcados por ‘x’	‘-.’	As linhas da função alternam entre pontos e traços
‘|’	Os pontos são demarcados por linhas verticais	‘k’	Linhas na cor preta
‘_’	Os pontos são demarcados por linhas horizontais	‘r’	Linhas na cor vermelha
‘s’	Os pontos são demarcados por quadrados	‘g’	Linhas na cor verde
‘d’	Os pontos são demarcados por “diamantes”	‘b’	Linhas na cor azul

‘^’	Os pontos são demarcados por triângulos	‘y’	Linhas na cor amarela
‘v’	Os pontos são demarcados por triângulos invertidos	‘c’	Linhas na cor ciano
‘>’	Os pontos são demarcados por setas apontadas para a direita	‘m’	Linhas na cor magenta
‘<’	Os pontos são demarcados por setas apontadas para a esquerda	‘w’	Linhas na cor branca
‘p’	Os pontos são demarcados por pentagramas		

11. Entrada e saída de dados.

11.1. Comando Input.

Permite que o programador se comunique com o usuário, o *Input* pede que o usuário insira uma resposta, obrigatoriamente, embora, o programador insira a mensagem que irá aparecer na tela quando a linha do código for executada. Ao ler essa linha, o programa identifica que existe uma variável que necessita receber um valor, caso ela tenha sido declarada anteriormente, por exemplo, *x=input*, nesse caso o valor inserido será atribuído a x, do contrário, o valor irá para a variável temporária ‘ans’.

Ao inserir esse programa, é necessário escrever uma mensagem, o comando deve ser utilizado na forma *input(‘Mensagem’)*, nesse caso, o programa irá receber um número como resposta, caso a resposta seja uma string, ou seja, uma palavra, é preciso declarar da forma *input(‘Mensagem’,”s”)*.

11.2. Comando disp e fprintf

Permite que o usuário exiba na tela, valores de variáveis e frases, utilizando *disp(‘Frase ou nome da variável’)*, a mensagem exibida na tela será somente o valor da variável, sem mencionar qual o nome da mesma.

Já o comando *fprintf*, funciona de forma semelhante ao comando *printf* em linguagem C, isso significa que este permite que uma mensagem seja exibida juntamente com valores de variáveis. Como pode ser visto no exemplo a seguir:

Exemplo: Vamos supor que precisamos criar um programa que peça 3 informações para o usuário. São essas, seu nome, sua idade e sua altura, então, o programa criado ficaria assim.

```
Nome = input(‘Qual o seu nome?’,”s”);
Idade = input(‘Qual a sua idade?’);
Altura = input(‘Qual a sua altura?’);
fprintf(‘Nome: %s \n Idade: %d \n Altura: %f’, Nome, Idade, Altura)
```

Onde \n é um comando que permite que quando a mensagem for exibida, no ponto onde foi escrito \n, exista uma quebra de linha. As letras que seguem o símbolo de porcentagem fazem referência ao tipo de variável que são informadas após o sinal de

apóstrofo, como, respectivamente, Nome é uma string, Idade é um inteiro e Altura é um float, ou seja, assume valores com valores ‘após a vírgula’.

12. Estruturas e Laços Condicionais.

12.1. Comando if, else e elseif.

O comando *if*, do inglês ‘se’, estabelece uma ou mais condições definidas pelo programador, caso essas sejam atendidas, o programa irá executar uma série de comandos também escritos pelo programador, dessa forma criando o que chamamos de estrutura condicional. Essas condições são nada mais que valores de variáveis, após definir uma variável, colocamos na estrutura condicional algo como *variável==valor*, isso diz ao programa que ele deve checar se *variável* possui o *valor* que foi exibido, caso ela possua, a condição é verdadeira, e o comando é executado.

Ao inserir o comando *if(condição)*, o programa entende que logo em seguida, o usuário venha a escrever os comandos a serem executados caso a condição seja atendida, porém, caso essa não seja, existem algumas outras possibilidades. Primeiramente, o programador pode usar o comando *else* como forma de dizer ao programa, que na situação em que a condição não for atendida, todo o restante de possibilidades que não pertencem a essa condição caem automaticamente na situação do *else*, isso significa que os comandos que seguem o comando *else* serão executados, porém os que seguem o comando *if* não serão.

A segunda opção para o programador é utilizar o comando *elseif(condição)*, essa combinação determina que caso a primeira condição não seja atendida, o programa deve verificar se a essa condição é verdadeira.

Ao final da estrutura condicional, independente dela possuir *else* ou *elseif*, deve ser encerrada com o comando *end*. Tenha em mente que não é necessário utilizar *else* e *elseif* obrigatoriamente, sendo possível criar uma estrutura utilizando apenas um comando *if*.

Para utilizar mais de um parâmetro no mesmo comando, por exemplo, x<10 e x>5, utilize &&, para representar a condição ‘e’, e utilize || para representar a condição ‘ou’.

Tabela 10 – Diagrama da estrutura condicional

Comando	Avaliação	Resultado
Primeira condição (*if*)	Verdadeira	Comando executado
	Falsa	Pular comando
Segunda condição (*elseif*)	Verdadeira	Comando executado
	Falsa	Pular comando
Todo o restante (*else*)		Sempre será executado
		Sempre será executado

Exercício 26: Suponha que uma empresa de viagens a ônibus ofereça gratuidade a crianças menores de 10 anos, 50% do preço para adolescentes de 11 a 18 anos, 100% do preço para adultos até 59, e 70% de desconto para idosos a partir de 60. Crie um programa que pergunte a idade do usuário e consulte qual desconto ele tem direito, para então exibir uma mensagem na tela comunicando a ele.

Resolução:

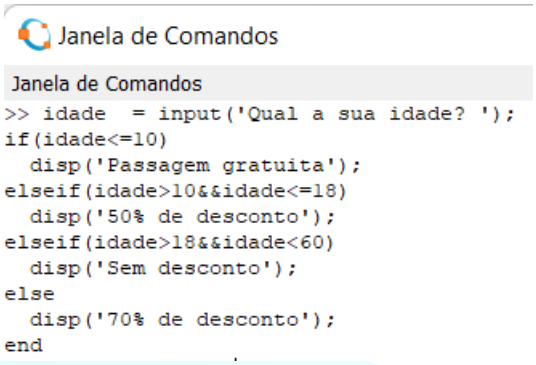

Figura 44 – Comandos do exercício 26.

12.2. Comando for.

Para esse comando criaremos um laço de repetição baseado num vetor, que por sua vez será criado por uma progressão aritmética. Dessa forma utilizaremos, *for v = 0:10*, nesse caso, será criado o vetor *v*, que possuirá valores começando em 0, esse será o nosso contador de repetições. Após declarado o nosso contador, devemos inserir comandos para serem executados em cada iteração do comando, e então toda vez que esses comandos forem executados o programa retornará para o comando *for* e o contador será incrementado.

Exercício 27: Crie um programa que monte uma matriz baseado em valores que o usuário determina.

Resolução:

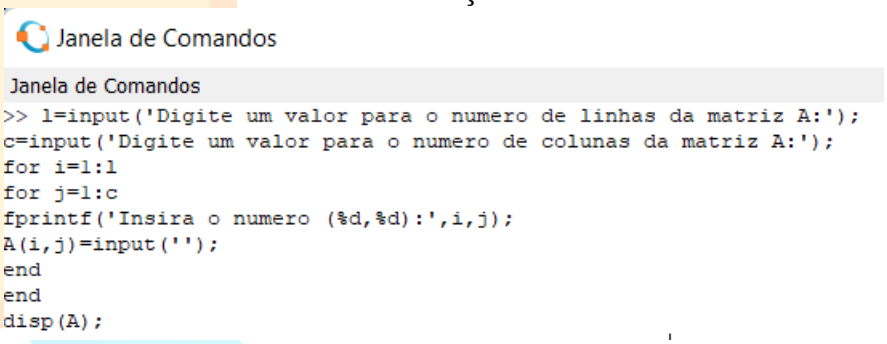

Figura 45 – Comandos do Exercício 27

12.3. Comando while.

Permite que o programa execute uma série de comandos repetidas vezes, enquanto uma condição não for cumprida. Essa condição pode ser algum tipo de comparação numérica, e o programa, por exemplo, *while a<5*, se *a* é uma variável de valor 0, e a cada iteração de *while*, *a* é incrementado em 1, então o programa continuará executando esses comandos até *a* atingir o valor 5, dessa forma tornando a condição *a<5* falsa. Também é possível criar um looping infinito utilizando, basta apenas manter o valor de *a* constante durante todas as interações.

12.4. Comando switch.

Esse comando é muito utilizado em situações em que é preciso que uma variável assuma valores específicos. Por exemplo, podemos criar um programa em que o usuário precise escolher qual valor atribuir a x, onde os valores podem ir de 1 a 3, nesse caso, o comando ficaria:

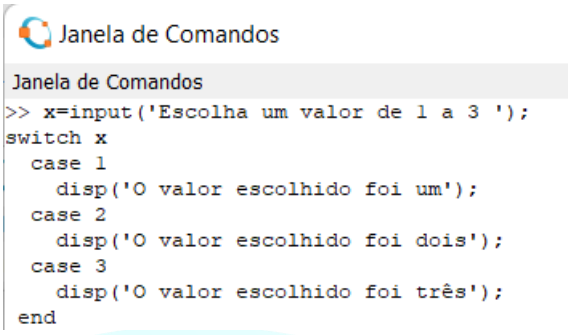

Figura – 45 Comando *While*.

O exemplo, embora simples, é apenas uma de muitas possibilidades do comando switch.