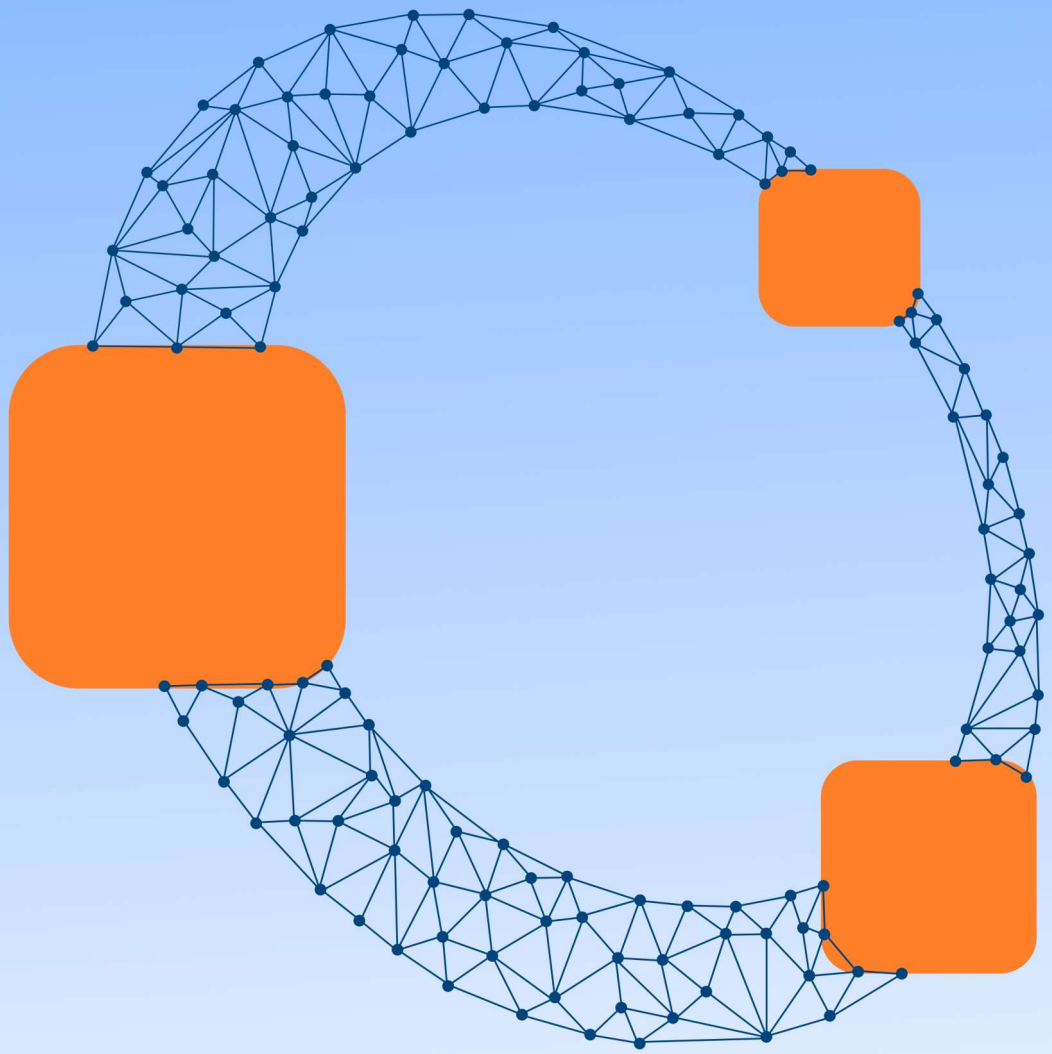
GNU Octave

Básico



Apostila do minicurso

PROGRAMA DE EDUCAÇÃO TUTORIAL ENGENHARIA MECÂNICA, UNIVERSIDADE FEDERAL DO ESPIRITO SANTO - CAMPUS GOIABEIRAS

Caro estudante,

Bem-vindos à mais uma apostila desenvolvida pelo PET Engenharia Mecânica para a SATE (Semama de Atualização e Treinamento em Engenharia), que visa aprofundar seus conhecimentos sobre o GNU Octave.

Para que seu estudo se torne proveitoso e prazeroso, esta apostila foi organizada em capítulos, com temas e subtemas que buscam apresentar o software e embasá-lo para que você possua o mínimo necessário para desenvolver seus trabalhos utilizando o software.

Vamos, então, iniciar nossas aulas? Bons estudos!

1ª Ed.

Conteúdo

1. O Software GNU Octave

Atualmente existem excelentes softwares de cálculo numérico, mas o GNU Octave se destaca por ser de livre distribuição e/ou modificação, o que faz este softwere possuir uma extensa gama de pessoas contribuindo com o melhoramento e adição de novas funcionalidades, atraindo cada vez mais aa atenção da comunidade científica.

As funções do Octave abrangem desde resolução de problemas lineares até integração numérica e tudo isso ainda pode ser expandido com a utilização com módulos carregáveis.

1.1 História do Octave

O GNU Octave foi originalmente escrito em 1988 para referenciar uma apostila sobre reatores químicos escrita por James B. Rawling (Universidade de Wisconsin-Madison) e John G. Ekerdt (Universidade do Texas). A primeira versão disponibilizada teve sua versão alfa liberada em 04 de janeiro de 1993, sendo a primeira versão final foi disponibilizada para utilização em 17 de fevereiro de 1994.

Desde a sua primeira versão até hoje, o software sofreu inúmeras melhoria e deixou de ser destinado apenas para o projeto de reatores químicos, atendendo às necessidades do meio acadêmico, científico e comercial.

Apesar de inicialmente ter sido pensado em Fortran, o Octave é completamente escrito em C++ e usa um interpretador dessa linguagem para a execução dos *scripts*, além de contar com a funcionalidade dos gráficos através de softwares como gnuplot e Grace.

Hoje o Octave, mesmo tendo o intuito da gratuidade de utilização, tem total compatibilidade com o MATLAB, seu principal rival, possuindo diversas funções semelhantes.

Mais informações e também o download do software pode encontrados do site do próprio GNU Octave: https://www.gnu.org/software/octave/.

1.2 Instalando o Octave

Como dito anteriormente, o software pode ser baixado no site supracitado simplesmente clicando em "Download"na barra superior. Em seguida, será solicitado qual o sistema operacional e a arquitetura do processador e do sistema operacional empregados (geralmente, 32 ou 64 bits).

O Octave é compatível atualmente com Mac, Linux, Windows e BSD (Um sistema operacional de código aberto distribuído pela Universidade da califórnia direcionado para o meio científico), então, certamente ele abrange a grande maioria das máquinas disponíveis no mercado. Finalmente, resta apenas esperar o fim do download e executar o instalador e seguir os passos de instalação geral de qualquer software.

1.3 Inicializando o Octave

A inicialização do Octave é relativamente simples e não requer nada que outro programa não necessite, porém, é importante ressaltar que, depois de instalado existiram dois arquivos executáveis: o "GNU Octave (CLI)"e o "GNU Octave (GUI)". O executável "(GUI)"abrirá a janela principal do Octave que conta com uma interface mais amigável aos mais diversos utilizadores, já o atalho "(CLI)"abrirá o prompt de comando que executará os comandos do Octave nas tradicionais linhas de comando, mas nesta apostila, tratar-se-á da utilização do atalho "Octave (GUI)"por ser mais agradável à maioria dos utilizadores.

1.4 Interface Gráfica

A janela principal do Octave apresenta três principais regiões fundamentais para o seu funcionamento.

O navegador de arquivos, na parte superior esquerda é onde, antes de iniciar a programação, o usuário deve escolher uma pasta em que o Octave salvará os arquivos gerados durante a utilização, permitindo uma melhor organização dos arquivos utilizados.

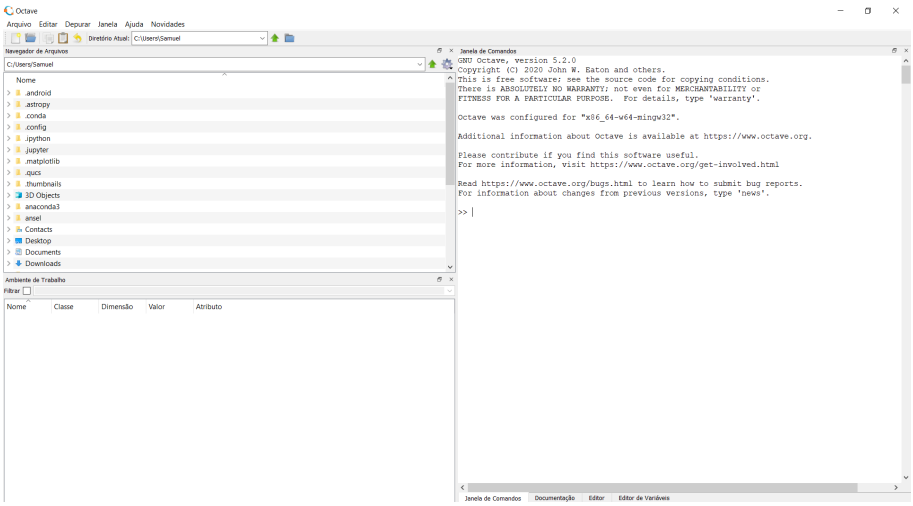


Figura 1.1: Tela inicial do GNU Octave.

O ambiente de trabalho, logo abaixo do navegador de arquivos fornecerá, após a inserção de variáveis, uma tabela que resumirá as variáveis empregadas no código, detalhando seu nome, classe, dimensão, valor e atributo. Todas essas informações sobre as variáveis serão tratados mais a diante.

A terceira parte da janela aberta por padrão é a janela de comandos. Nela, é possível inserir códigos e executá-los em tempo real, porém, nada do que for escrito nesta janela será salvo. Uma alternativa a isso é utilizar o editor.

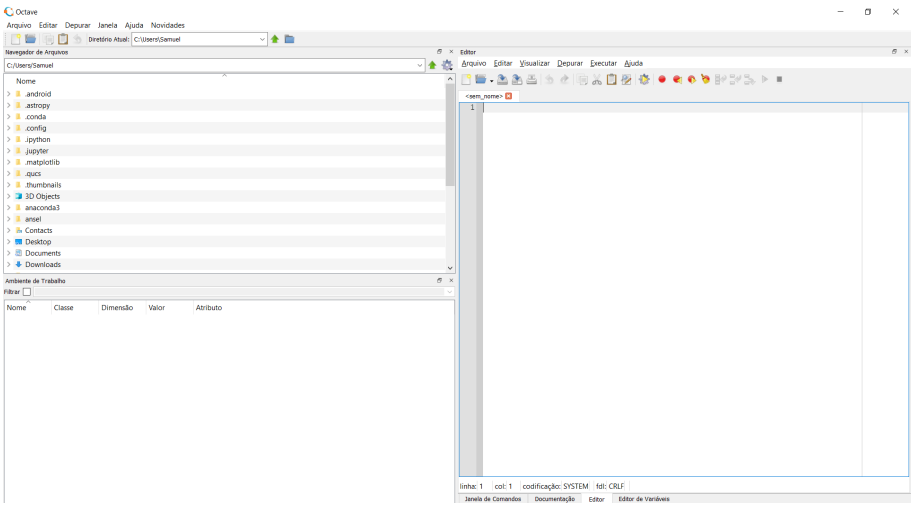


Figura 1.2: Aba editor aberta.

A janela editor pode ser aberta simplesmente clicando nessa opção no canto inferior direito, ao lado da opção "Documentação". Nesta janela, é possível inserir *scripts* que serão executados na janela de comandos, mas desta vez, com a vantagem de poder salvar o o código já escrito. A

cada vez que o código for executado, é necessário salvar as alterações no script, mas essa tarefa é automatizada pelo botão "Salvar Arquivo e Executá-lo", a engrenagem cinza com um triângulo amarelo.

2. Trabalhando com Variáveis

2.1 Declaração de Variáveis

A utilização de variáveis no Octave é razoavelmente simples e não requer, como em outras linguagens de programação, uma identificação prévia do tipo da variável. Aqui, utiliza-se a sintaxe para identificar o tipo de variável.

O nome da variável continua sendo essencial nos códigos e justamente por isso é que existem regras para criá-los:

- O nome deve iniciar com uma letra, seja maiúscula ou minúscula;
- Depois do primeiro caractere, pode-se usar números, ou outras letras, sendo que o interpretador distingue entre letras maiúsculas e minúsculas;
- O caractere de espaço não deve ser usado. Para separar palavras no nome da variável utiliza-se o "_"(*underline*).

A declaração de variáveis se dá escrevendo-se o nome, o símbolo de igual "="e o valor atribuído à variável como no exemplo a seguir.

■ **Exemplo 2.1** Exemplo de declaração de variável no Octave.

```
alfa1 = 30
alfa2 = 45
```

■

Algumas variáveis são reservadas para valores específicos pelo Octave, portanto, elas não podem ser utilizadas como nomes de outras variáveis. Algumas delas são:

ans	guarda resultados de operações que não foram armazenados em outras variáveis
pi	número Π
eps	menor número que somado a 1, é interpretado pelo computador um número que 1.
flops	número de operações em ponto flutuante. Ainda não foi implementado.
inf	Infinito.
NaN ou nan	Not a number, erro exibido para indeterminações matemáticas.
i e j	Usado para representar a parte imaginária de números complexos.
nargin	Número de argumentos de entrada de uma função
nargout	Número de argumentos que uma função devolve.
realmin	Menor número real que o computador consegue armazenar.
realmax	Maior número real que o computador consegue armazenar.
e	Constante de Euler, aproximadamente 2,718.

Um ponto importante a ser ressaltado é que mesmo que essas variáveis sejam ditas reservadas, elas ainda podem ser declaradas com outros valores, porém, neste caso, elas perderão seus valores iniciais. Por exemplo, se a variável "e" for declarada com o valor 5 (e=5), ela deixará de retornar o valor da constante de Euler, retornando 5 quando solicitada.

Existem ainda alguns comandos importantes para lidar com variáveis que estão listados abaixo.

who	lista as variáveis já declaradas pelo usuário.
clear + variável	O comando clear seguido do nome da variável apaga esta variável.
clear	Apaga todas as variáveis já declaradas pelo usuário.

2.2 Operações Básicas

O Octave, como tantos outros softwares de cálculo numérico, apresenta as operações matemáticas básicas. Na tabela abaixo, se encontram listadas as principais operações matemáticas entre duas variáveis a e b.

Adição	a + b
Subtração	a - b
Multiplicação	a * b
Divisão direta	a / b
Divisão indireta	a \ b
Potenciação	a ^ b
Raíz quadrada	sqrt(a)
Fatorial	factorial(a)

2.3 Tipos de valores guardados

2.3.1 Vetores e matrizes

Como dito anteriormente, é possível guardar tipos diferentes de valores em variáveis no Octave. Um tipo de variável frequentemente utilizada são os vetores e matrizes. Para declarar uma matriz, utiliza-se a vírgula para mudar de coluna (ou um simples espaço) e o ";" para mudar de linha e tudo isso deve ser inserido dentro de colchetes que delimitará o início e o fim da matriz. A declaração de vetores não é diferente, porém, pode-se utilizar de vetores verticais, utilizado-se o ponto e vírgula, ";", entre os componentes do vetor, ou vetores horizontais, separando os componentes por vírgula.

■ **Exemplo 2.2** Exemplo de declaração de variável vetorial ou matricial. ■

```
>>  x=[2,4,6;8,10,12]
x =

    2    4    6
    8   10   12
```

```
>> vet_a=[1,2,3]
vet_a =

   1   2   3
```

```
>> vet_b=[1;3;9]
vet_b =

   1
   3
   9
```

Figura 2.1: Exemplo de matriz (x), vetor horizontal (vet_a) e vetor vertical (vet_b)

2.3.2 Guardando palavras

O último tipo de variável a ser abordada nesta apostila são as *strings*. As *strings* nada mais são do que sequencias se caracteres que são guardadas na forma de um vetor, portanto podem ser utilizadas como tal. As *strings* são declaradas com de forma semelhante às outras variáveis com seu nome, o símbolo de igual e o seu valor entre aspas simples: nome = 'PET Mecanica', sendo que os espaços contam como um caractere.

Vale lembrar que quando armazena-se um algarismo arábico, um número, em uma *strings*, ele será interpretado como um caractere e não com o número que ele é, sendo utilizado a tabale de conversão ASCII para efetuar os cálculos e por isso podem resultar em valores estranhos à primeira vista.

As operações e manipulações com os tipos de variáveis já citados serão tratados no decorrer desta apostila, mas antes é interessante embasar alguns comandos que são frequentemente usados nessas operações.

3. Comando de fluxo, lógico e relacional

3.1 Operadores lógicos

O uso de comparações é recorrente no calculo numérico e uma das ferramentas que facilitam essas comparações são os operadores lógicos, que são usados quando se quer usar duas ou mais comparações. Os operadores lógicos no Octave são:

&&	E
||	OU
~ ou !	NÃO

- E

  Quando ocorrer a associação de duas expressões com &&, o resultado será verdadeiro (1) se ambas as expressões resultarem em verdadeiro individualmente. Caso uma delas seja falsa, ou ambas sejam falsas, o resultado será falso (0).

- OU

  A combinação de duas expressões com || resultará em verdadeiro sempre que pelo menos uma das expressões for verdadeira.

- NÃO

  O caractere til (~) ou a exclamação (!) têm o papel de inverter o resultado de uma expressão, mudando de verdadeiro para falso e vice e versa.

3.2 Operadores Relacionais

Assim como nas linguagens de programação, os operadores relacionais também estão presentes no Octave e seus símbolos seguem na tabela a seguir:

<	Menor que
<=	Menos ou igual
>=	Maior ou igual
>	Maior que
==	Igual
~= ou !=	diferente

O resultado dos operadores relacionais é sempre verdadeiro (1) ou falso (0). Para o caso de vetores e matrizes, o resultado será sempre um vetor ou uma matriz de zeros e uns conforme o resultado da comparação termo a termo.

3.3 Comandos de Fluxo

Os comandos de fluxo são usados para executar repetições de partes do código ou determinar se uma parte do código vai ou não ser executada. Os comandos de fluxo mais utilizados são o **for**, **while** e o operador **if-elseif-else**. Todos esses três comandos devem ser encerrados por end+nome ou simplesmente **end** , por exemplo, o comando **if** é encerrado por **endif** ou por **end**.

3.3.1 For

O for é o comando que executa um loop por uma quantidades de vezes predefinida e fixa. Sua sintaxe é dada no exemplo a seguir.

```
for condição
    Código a ser repetido.
endfor
```

A condição pode ser escrita de forma simples declarando-se uma variável que poderá ser exclusivamente usada no loop escrito da forma **i=a:b**, em que i é o nome da variável de controle do loop, e o comando a:b cria um vetor que começa no valor de a e vai com passo 1 até o valor de

b, por exemplo um loop for usando como condição i=1:5 executa o código 5 vezes. O comando for pode ser usado de forma aninhada, sempre tendo-se o cuidado de usar variáveis de controle diferentes para cada vez que o for é escrito. O exemplo a seguir exibe a sintaxe de dois comandos for aninhados, em que as palavras "PET Mecanica"é exibida na tela 25 vezes.

■ **Exemplo 3.1** Utilização do loop FOR.

```
for i=1:5
  for j=1:5
     printf("PET Mecanica ")
  endfor
endfor
```

■

3.3.2 While

O comando **while** funciona de forma análoga ao **for**, porém desta vez, é possível alterar a variável de controle durante a repetição, ou seja, a condição que vai controlar o loop vai depender do próprio loop. A variável de controle desta vez, deve ser uma constante determinada anteriormente do código e não recomenda-se que ela seja restrita ao loop, como ocorre no for. O a repetição while também pode ser aninhada da mesma forma que o for. Um exemplo de sintaxe do comando while é dada a seguir.

■ **Exemplo 3.2** Utilização do loop WHILE.

```
x=0
while x<5
   printf("PET Mecanica ")
   x=x+1;
endwhile
```

■

Este exemplo exibe as palavras "PET Mecanica"por 5 vezes.

3.3.3 If-elseif-else

Os comandos if e else, definem se o código em seu interior será executado, e em caso negativo, o que será feito.

A sintaxe desde comando possibilita a utilização de várias condições alternativas como o exemplo a seguir.

■ **Exemplo 3.3** If-elseif-else

```
x=0;
if x==1
  printf("Este e um exemplo")
  elseif x>=3
    printf("Em que isto nao e executado")
  else
    printf("Esta frase e exibida na saida.")
end
```

■

Neste exemplo, a variável x guarda o valor 0. Quando a primeira condição é testada, é retornado falso e o comando não é executado. Na segunda condição, o retorno também é falto e nada é feito. Para este caso em que todas as condições retornaram falso, é executado o comando depois de *else* e apenas a frase "Esta frase é exibida na saída."

4. Matrizes e Vetores

Neste capítulo serão apresentadas alguma ferramenta para facilitar a manipulação de vetores e matrizes no Octave.

4.1 Vetores e matrizes gerados por comando

4.1.1 Vetor por incremento

Este método cria um vetor a partir da informação do primeiro termo, do limite final e do passo (ou incremento) a ser utilizado. Caso o incremento seja unitário ele pode ser omitido. A estrutura básica é escrita separando os três parâmetros por ":", sendo o parâmetro central, o incremento. O exemplo a seguir cria um vetor composto pelos números 1, 5, 9, 13 e 17. Note que, caso o pode acontecer de o limite final não fazer parte do vetor, sendo ele necessário como parâmetro de parada.

O comando:

vet=1:4:20;

Tem como resultado o seguinte vetor.

vet=

1 5 9 13 17

4.1.2 Vetores linearmente espaçados

O Octave também permite a criação de um vetor em que todos os termos que o compões estão linearmente espaçados entre dois limites. Neste comando chamado *linspace* deve-se informar o valor do primeiro elemento, o valor do último elemento e o numero de elementos que o vetor terá, todos separados por vírgula.

O comando:

vet=linspace(0,15,5)

Tem como resultado o seguinte vetor.

vet=

0 3.75 7.5 11.25 15

4.1.3 Valores aleatórios

Para criar um vetor ou matriz com valores aleatórios entre 0 e 1, basta usar o comando *rand* informando o número de linha e o de colunas separados por vírgulas.

mat=rand(3,4)

Resultado: Uma matriz de 3 linhas e 4 colunas com números aleatórios entre 0 e 1.

4.1.4 Matriz mágica

O comando *magic* cria uma matriz quadrada de números inteiros sem repetição de forma que as somas dos números das linha, das das colunas e das somas diagonais são todas iguais ao mesmo valor. Como os números utilizados estão entre 1 e o quadrado do tamanho da matriz, quando deseja-se uma matriz 2x2, não se obtém essa propriedade das somas.

```
magic(3)
ans =
8    1    6
3    5    7
4    9    2
```

4.1.5 Vetor ou matriz nulos

A função **zeros** cria um vetor ou uma matriz de zeros a partir da informação do número de linhas e colunas separados por vírgulas.

```
zeros(1,5)
ans =
0 0 0 0 0
```

- Também existe o comando **ones** que, funcionando da mesma forma que o **zeros**, cria uma matriz de 1's.

4.1.6 Matriz identidade

O comando **eye** cria uma matriz identidade do tamanho desejado, apenas informando o tamanho.

4.1.7 Triângulo de Pascal

O comando **pascal** cria uma matriz composta pelo triângulo de Pascal do tamanho desejado.

```
pascal(4)
ans =
1   1   1   1
1   2   3   4
1   3   6   10
1   4   10   20
```

4.2 Alterando um elemento de um vetor ou matriz

É possível alterar elementos específicos de uma matriz ou vetor usando a localização do elemento entre parênteses. Para matrizes, a localização é dada primeiro pelo número da linha e depois pelo número da coluna separados por vírgulas. Em vetores, não há distinção entre linhas e colunas.

Vale lembrar que, diferentes de outras linguagens como a linguagem C que começa em 0, as posições de um elemento em um vetor ou matriz começam a ser contadas do 1 para a primeira linha, 2 para a segunda linha e assim por diante e funcionando da mesma forma para as colunas.

Caso a posição escolhida ainda não exista, ela será inserida. No caso das matrizes, se apenas um único elemento for inserido em uma linha ou coluna, os outros elementos necessários serão preenchidos por zeros enquanto não forem alterados

Normalmente, quando se quer alterar todos os elementos, utiliza-se um loop para automatizar o processo. No exemplo de sintaxe a seguir, uma matriz de uns tem seus elementos substituídos pela soma do número da linha e da coluna do elemento.

```
mat=ones(3)
for i=1:4
   for j=1:4
      mat(i,j)=i+j;
   endfor
endfor
mat
```

- Resultados exibidos:

```
mat =
1   1   1
1   1   1
1   1   1
mat =
2   3   4   5
3   4   5   6
4   5   6   7
5   6   7   8
```

Caso seja utilizado ":" no lugar do número de linhas, será possível alterar todos os elementos da coluna informada utilizando-se um único número ou um vetor de mesmo tamanho da coluna escolhida para quando os elementos são diferentes. Da mesma forma acontece se os ":" são colocados no lugar do número da coluna.

mat=eye(3)

mat(:,1)=[9 9 9]

- Resultado exibido:

mat =

Diagonal Matrix

1 0 0

0 1 0

0 0 1

mat =

9 0 0

9 1 0

9 0 1

4.2.1 Apagando elementos

Para deletar algum elemento de um vetor ou linha ou coluna de uma matriz, deve-se utilizar o seguinte escrever o nome da variável com o números da linha ou coluna que o elemento a ser deletado pertence separados por vírgula, usar o "=" e por fim inserir "[]".

4.3 Operações

As operações de soma, subtração e produtopor escalar funcionam como operações termo a termo para vetores e matrizes. O produto entre dois vetores ou duas matrizes ocorre de forma habitual com o símbolo "*". O produto elemento a elemento é executado ao se inserir um ponto antes do "∗". Ainda é possível executar a potenciação de vetores e matrizes termo a termo utilizando o nome da variável com os símbolos ".^ " e o expoente desejado. Caso o "." não seja inserido, a potenciação será resolvida utilizando o produto habitual de matrizes. A transposição de uma matriz é feita de forma automática com a utilização do símbolo "'" (apóstrofo) ou a função **ctranspose**. Por fim, a divisão termo a termo é executada pelo comando "./"

4.4 Funções com vetores matrizes

Algumas funções importantes que podem simplificar o trabalho com matrizes são apresentadas a seguir com a sua sintaxe.

det(A)	Determinante de A
inv(A)	Inversa de A
[P,D]=eig(A)	Autovetores na matriz P e autovalores em D
poly(A)	Retorna um vetor com os coeficiente do polinômio característico da matriz A quadrada.
norm(vet)	Retorna a norma do vetor vet.
cross(r,f)	Retorna o produto vetorial de dois vetores r e f tridimensionais.
dot(r,f)	Retorna o produto escalar de dois vetores r e f quaisquer.
sum(vet)	Retorna a soma dos elementos do vetor vet.
max(vet)	Retorna o maior elemento do vetor vet.
min(vet)	Retorna o menor elemento do vetor vet.
sort(vet)	Reorganiza os elementos em ordem crescente.
-sort(-vet)	Reorganiza os elementos em ordem decrescente.
size(vet)	Retorna a dimensão de um vetor ou matriz.
length(vet)	Retorna a dimensão de um vetor ou o número de colunas de uma matriz.

4.5 Divisão

O resultado da linha de comando "A/B", em que A e B são matrizes, é o produto de A pela inversa de B à direita de A. Já utilizando a barra "\" indicaria que B seria multiplicado pela inversa de A à esquerda de B. Este comando é útil para resolver sistemas lineares conhecendo a matriz dos coeficientes do sistema e o vetor de termos independentes. Considerando um sistema $Ax = b$, a divisão à esquerda ($x = A\backslash b$) é o vetor solução do sistema.

- Dica:

Se a barra está tombada para a direita o comando multiplicará pela inversa da matriz a direita, se for tombada à esquerda, será realizada a multiplicação pela matriz inversa à esquerda.

5. Criando Funções e Rotinas

5.1 Funções

O GNU Octave oferece uma ampla gama de funções prontas que facilitam a vida de quem está criando seus códigos nele. na tabela a seguir, estão listadas as principais funções pré-definidas, outras estão sendo apresentadas ao longo desta apostila e muitas outras estão disponíveis e podem ser encontradas com uma simples pesquisa na internet.

abs(x)	valor absoluto de x
acos(x)	arco cosseno de x
asin(x)	arco seno de x
atan(x)	arco tangente de x
cos(x)	cosseno de x
sin(x)	seno de x
tan(x)	tangente de x
exp(x)	exponencial de x
log(x)	logaritmo natural de x
log10(x)	logaritmo de x na base 10
ged(x,y)	máximo divisor comum de x e y
lem(x,y)	mínimo múltiplo comum de x e y
rem(x,y)	resto da divisão de x por y

Vale ressaltar que todas as funções trigonométricas utilizam os ângulos em radianos.

Porém, mesmo com essa grande gama de funções, pode ser necessário criar funções para casos específicos. Para a criação de funções, o Octave oferece duas possibilidades: criá-las no corpo do código ou salvá-las separadamente. Esta última possibilidade será tratada com maiores detalhes no final deste capítulo.

A criação de funções no corpo do código se dá de forma simples: basta obedecer a sintaxe a seguir:

```
function F=nome(x)
    Código que a função executará
endfunction
```

A letra F é o nome da variável que será utilizada apenas dentro do código da função para armazenar o valor que a função retornará. A palavra "nome" pode ser substituída pelo nome for desejado e será esta palavra que será utilizada para usar a função no código. Por fim, o "x" entre parênteses representa um parâmetro da função, ou seja, um valor que será utilizado pelo código da função. Caso sejam necessários mais de um parâmetro, eles podem ser inseridos separados por vírgula.

Um exemplo pode ser dado para a melhor compreensão: Criar-se-á uma função que retorna o valor da função z=4y-3x+5. Note que para esta função, são necessários dois parâmetros, x e y.

■ **Exemplo 5.1** Criação de uma função:

```
function z=FdeXY(x,y)
  z=4*y-3*x+5;
endfunction
a=1;
b=4;
valor=FdeXY(a,b)
```

■

Note que depois de criada a função, segue-se o código normalmente e quando se utiliza a função, passa-se os parâmetros dentro dos parênteses, sendo que neste caso, o valor de "a" é copiado para a variável "x" da função e o valor de "b" é copiado para a variável "y" da função. No final da função, o valor de z é guardado na variável "valor".

5.2 Funções anônimas

As funções são muito úteis no decorrer da programação, porém, quando uma função vai ser utilizada uma única vez, não há necessidade de empregar todo o formalismo já apresentado. As funções anônimas são escritas em uma única linha e são indicadas para funções simples que

serão utilizadas dentro de outras funções como evidenciará nos capítulos a seguir. A sintaxe de uma função anônima inicia-se com uma "@", seguida dos parâmetros utilizados na função entre parênteses e a expressão da função também entre parênteses. A seguir tem-se um exemplo de uma função que somará os senos de duas variáveis, x e y:

■ **Exemplo 5.2** Criação de uma função anônima:

```
@(x,y)(sin(x)+sin(y))
```

■

5.3 Rotinas

Os comandos do Octave são executados na janela de comandos, porém, como comentado anteriormente, os códigos escritos neste ambiente não são salvos, ou seja, ao fechar o Octave, tudo é perdido. Entretanto, para executar uma sequência de comandos em série, e armazenar estes códigos para posterior execução e/ou edição, é conveniente salvá-los.

Ao codificar na janela editor, o código é armazenado em um arquivo do tipo "m" e toda vez que ele é executado, sendo necessário, na primeira execução, escolher o nome e o local em que ele será armazenado. A partir da primeira execução, o Octave salvará todas as alterações neste mesmo arquivo a cada execução, sem a necessidade de salvar e depois executar o código.

O nome do arquivo deve seguir algumas regras para que o Octave consiga executá-lo: O mome não deve ter acentos, hifens ou outros caracteres que não sejam letras, números e *underlines*.

Os arquivos criados para armazenar o código são denominados rotinas, M-files ou script files e podem ser abertos sem a necessidade de procurá-lo dentro do Octave, como acontece com um arquivo do "Word", por exemplo.

Outra utilidade para os M-files é a possibilidade de criar funções que ficam armazenadas em M-files diferentes do que aquele em que foi salvo o restante do código. Essa funcionalidade promove uma maior organização do código e a possibilidade de usar uma função em vários códigos diferentes sem a necessidade de reescrevê-las.

Para utilizar esse recurso, o arquivo que contêm a função deve estar salvo na mesma pasta em que o código foi salvo e o nome do arquivo ".m" da função deve ser o mesmo utilizado para usar a função no código.

6. Gráficos

6.1 Gráficos 2D

Os gráficos mais comuns são os bidimensionais, normalmente apresentados em coordenadas cartesianas ou polares.

O comando mais utilizado para plotar gráficos 2D em coordenadas cartesianas é o **plot**. Como parâmetros, essa função pode aceitar apenas as variáveis a serem colocadas nos gráficos separadas por vírgulas, ou informações mais complexas que vão tratar do tipo e cor, por exemplo.

Outros comandos úteis são o **semilogx** , que plota gráficos em que a coordenada x está em escala logarítmica; **semilogy**, que faz o mesmo, mas com a coordenada y; **loglog**, que gera gráficos em escala logarítmica nas duas coordenadas; **polar**, gera gráficos em coordenadas polares; **stairs** gera gráficos em degrau; **bar** que plota histogramas; e **scatter**, que gera gráficos de dispersão.

Apesar de todas essas funções, para trabalhos acadêmicos e outras utilizações, é essencial que um gráfico possua algumas informações a mais. Tais informações podem ser inseridas direto no código e estão listadas a seguir:

title('Título')	Insere a palavra "Título" como título do gráfico
xlabel('legenda')	Insere a palavra "legenda" como legenda do eixo x
ylabel('texto')	insere a palavra "texto"como legenda do eixo y
text(x, y, 'texto')	Insere a palavra "texto" na coordenada de (x,y)
gtext('texto')	Adiciona a palavra "texto"na posição do mouse
legend('texto1')	Insere legendas nos gráficos
grid on	Insere grade que pode ser retirada mudando **on** para **off**
hold on	Plotas várias curvas no mesmo gráfico até que o código **hold off** seja executado
axis(v)	Plotar em uma faixa de valores (v é o vetor formado por xmin xmax ymin ymax)

Como já comentado, é possível modificar a aparência do gráfico. Abaixo estão listadas as opções de cores, tipos de linhas, espessura e marcadores possíveis no Octave.

Cor	Código
Azul	b
Branco	w
Ciano	c
Preto	k
Roxo	m
Verde	g
Vermelho	r
Amarelo	y

Marcador	Código
Mira	+
Círculo	o
Estrela	*
Ponto	.
Cruz	x
Quadrado	s
Diamante	d
Pentagrama	p

Linha	Código
Sólida	" -
Tracejada	"- -
Pontilhada	":
Traço-ponto	"-.

Por fim, tem-se a seguir um exemplo de utilização desses comandos.

■ **Exemplo 6.1** Criação de um gráfico 2D:

```
x = 0:pi/10:2*pi;
y1=sin(x);
y2=cos(x);
plot(x, y1, "-+r")
hold on
plot(x, y2, "bo--")
legend('Seno','Cosseno')
title('Seno e Cosseno de x')
xlabel('Angulo')
ylabel('Seno ou Cosseno')
grid on
```

■

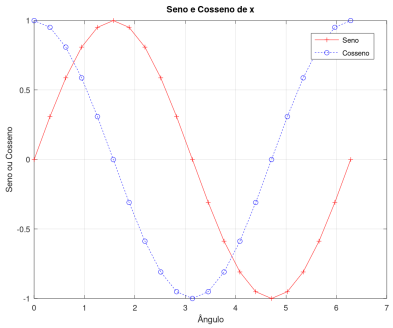


Figura 6.1: Resultado do Código.

6.2 Gráficos 3D

Os gráficos tridimensionais são de fácil execução no Octave, requerendo apenas um pouco mais de atenção do que os bidimensionais. A principal função para a criação de gráficos 3D é o **plot3** que funciona de forma semelhante ao **plot** e é indicado para funções que gerem uma linha como o exemplo a seguir.

■ **Exemplo 6.2** Gráfico para curvas tridimensionais:

```
t = 0: pi/10:10* pi ;
x=cos(t);
y=sin(t);
z=t
plot3(x, y,z, "-r")
legend('(cos(t),sin(t),t)')
title('Expiral')
xlabel('x(t)')
ylabel('y(t)')
zlabel('z(t)')
grid on
```

■

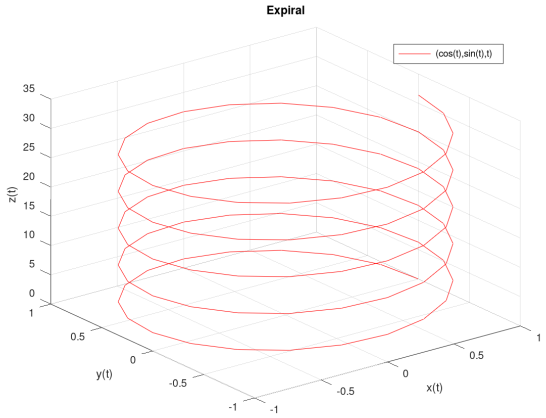


Figura 6.2: Resultado do Código.

A criação de gráficos de superfícies funciona com a criação de uma malha que representará a superfície. A função que executa tal ação é a função **mesh**, porém, antes de executá-la, deve-se definir uma malha de pontos em que o gráfico será formado com a função **meshgrid**. A seguir é apresentado um exemplo de utilização dessas funções.

■ **Exemplo 6.3** Gráfico de uma superfície:

```
[x,y]=meshgrid(-8:0.2:8,-8:0.2:8);
r=sqrt(x.^2+y.^2);
z=sin(r)./r;
mesh(x,y,z)
```

■

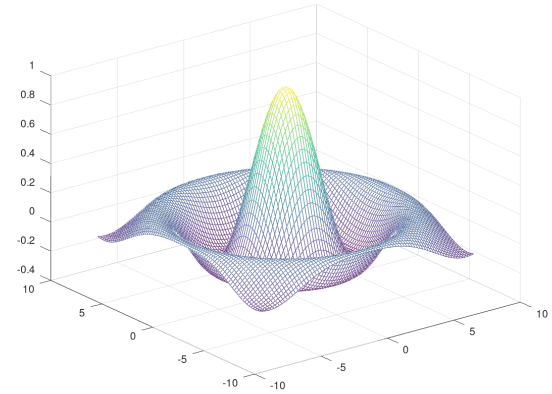


Figura 6.3: Resultado do Código.

Caso seja necessário utilizar uma aparência de superfície no lugar de uma malha vazada,

é possível utilizar a função **surf** que funciona de forma análoga à **mesh**. Outro recurso muito utilizado é a criação de curvas de nível a partir de superfícies tridimensionais. Essa função é executada pelo código **contour** que, dada a função e o número de curvas, ele forma as curvas de nível bidimensionais.

■ **Exemplo 6.4** Curvas de nível:

```
[x,y]=meshgrid(-8:0.2:8,-8:0.2:8);
r=sqrt(x.^2+y.^2);
z=sin(r)./r;
contour(z,5)
```

■

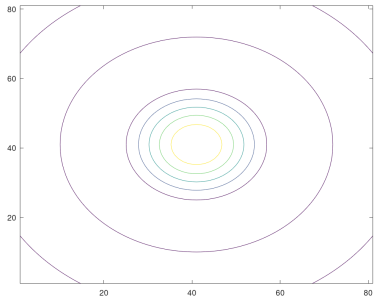


Figura 6.4: Resultado do Código.

Outras funções úteis como as funções de título, legenda e grid continuam a funcionar para gráficos 3D como funcionavam para os 2D.

6.3 Salvar Gráficos

Para salvar os gráficos gerados para posterior utilização em textos ou outros fins, o Octave dispõe de duas opções: gerar o gráfico e salvá-lo a partir da própria tela, como mostra a imagem a seguir.

Clicando em *Save As* é possível escolher o local, nome e formato de salvamento.

Porém, é possível automatizar este processo via código. Com a linha de código a seguir, é possível salvar o gráfico com o nome desejado na mesma pasta em que está armazenado o script. As palavras **formato** podem ser substituídas por "jpg", "png"ou "pdf"de acordo com o formato desejado. E **nome** pode ser alterada para o nome desejado.

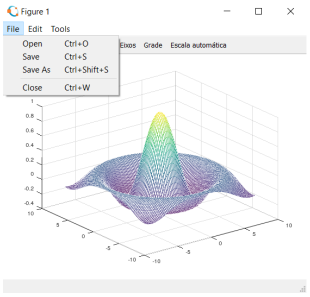


Figura 6.5: Resultado do Código.

print -d**formato nome.formato**

7. Polinômios

Os polinômios são muito utilizados para aproximar funções muito complexas ou completamente desconhecidas, por isso, o tratamento matemático de polinômios se torna muito importante. Desta forma, apresentar-se-a a seguir, um conjuntos de formas de utilizar polinômios no Octave.

7.1 Raízes

As raízes reais de um polinômio são determinadas com a função **roots(v)** em que **v** é um vetor que contém os coeficientes do polinômio. Por exemplo: para determinar as raízes do polinômio $-9x^2+8x+10$ declara-se **v=[-9 8 10]** e cria-se uma nova variável para receber o vetor coluna com as raízes encontradas pela função.

O Octave também permite a operação inversa, ou seja, determinar o polinômio a partir das suas raízes. A função **poly(r)** retorna os coeficientes do polinômio que apresenta as raízes do polinômio que apresenta as raízes contidas no vetor **r**.

7.2 Produto e Divisão

O produto e a divisão de dois polinômios são realizados pelas funções **conv(p1,p2)** e **deconv(p1,p2)** respectivamente, sendo **p1** e **p2** são os vetores que contêm os coeficientes dos vetores. Para a divisão, a ordem de inserção dos polinômios importa, já que o primeiro polinômio será o dividendo e o segundo, o divisor. A função deconv retorna dois vetores de coeficientes, sendo o primeiro o quociente e o segundo o resto.

No exemplo a seguir, tem-se o produto de **p1** $(4x^2-x+2)$ por **p2** $(x+1)$ e a divisão de **p1** por **p2**, sendo o quociente armazenado no vetor **quoc** e o resto em **resto**

```
p1 = [4 -1 2];
p2 = [1 1];
produto = conv(p1,p2)
[quoc resto] = deconv(p1,p2)
```

Resultado:

```
produto =
   4   3   1   2
quoc =
   4  -5
resto =
   0   0   7
```

7.3 Avaliação do Polinômio

Uma função polinomial pode ser avaliada em um ponto de diversas formas no Octave. Uma forma simples de fazê-lo é criar uma função que, dado o valor dos pontos desejados, retorne um vetor com os valores do polinômio nesse(s) pontos, respeitando, caso sejam tratados mais de um ponto, os critérios de operação com vetores apresentados na seção de Operações com Vetores. Essa forma de avaliação pode ser utilizada para qualquer outra função matemática que possa ser avaliada em um ponto.

A segunda forma de avaliar o polinômio é utilizando a função **polyval(p,x)**, em que **p** é um vetor contendo os coeficientes do polinômio e **x** é um vetor contendo os valores de x nos quais o polinômio será avaliado. A função **polyval** retorna um vetor com todas as avaliações para os valores de x informados.

7.4 Interpolação Polinomial

A interpolação polinomial pode ser muito útil para simplificar funções muito complexas em um conjunto de pontos ou construir funções a partir de um conjunto de dados discretizados seja para integral, derivar ou verificar valores intermediários posteriormente. A melhor opção no Octave para a construção do polinômio interpolador é a função **polyfit(x,y,n)**, que retorna os coeficientes do melhor polinômio que passa pelos pontos determinados pelos vetores **x** e **y** de grau **n**.

Usando a função **polyfit**, é possível ajustar curvas não polinomiais, apenas linearizando a curva por meio de manipulação matemática prévia. A seguir, tens alguns exemplos de curvas comuns e a forma do seu ajuste com a função **polyfit**.

$y = bx^m$	polyfit(log(x),log(y),n)
$y = be^{mx}$ ou $y = b10^{mx}$	polyfit(x,log(y),n) ou polyfit(x,log10(y),n)
$y = mlnx + b$ ou $y = mlogx + b$	polyfit(log(x),y,n) ou polyfit(log10(x),y,n)
$y = \frac{1}{mx+b}$	polyfit(x,1./y,n)

8. Técnicas de Cálculo

8.1 Máximos e Mínimos de uma Função

O Octave, como o Matlab disponibiliza duas funções para automatizar a procura de máximos e mínimos de uma função qualquer. No Octave, essa função é realizada por **fminbnd**(), que encontra o mínimo local da função. É importante resaltar que o máximo da função é encontrado, simplesmente multiplicando toda a função por -1, já que ainda não foi implementada uma função pronta para isso. A função fminbnd recebe como parâmetros o nome da função a ser analisada entre aspas simples e os valores de início e fim do intervalo a ser estudado e retorna o valor de x e y do mínimo da função.

A seguir, tem-se um exemplo de aplicação em que se determina o máximo e mínimo da função $y = x^2 + 2x$ entre os pontos -2 e 2.

■ **Exemplo 8.1** Máximos e mínimos de uma função:

```
function y=poly1(x)
  y=x^2+2*x;
endfunction
[xmin ymin] = fminbnd('poly1',-2,2);
function y=poly2(x)
 y=-x^2-2*x;
endfunction
[xmax ymax] = fminbnd('poly2',-2,2);
```

■

Note que os valores dos pontos (x,y) foram armazenados nas variáveis xmin e ymin para o ponto mínimo e xmax e y max para o ponto máximo. É importante ainda ressaltar que o valor de ymax está multiplicado por -1, já que todo o polinômio foi multiplicado por -1.

8.2 Derivação

A derivação de expressões no Octave é feita de forma simples para polinômios apenas utilizando a função **polyder()**, em que o seu único parâmetro necessário é o polinômio escrivo na forma de um vetor, como já foi discutido no capítulo anterior. É importante ressaltar que as derivadas de ordem superior de um polinômio é encontrada aplicando várias vezes a função **polyder**. Por outro lado, a derivação de outros tipos de funções não é tão simples. Tendo em vista que a derivada é aproximada pela inclinação da reta que passa por dois pontos próximos (x_1, y_1) e (x_2, y_2) da forma:

$$\frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1}$$

Pode-se criar um vetor que contenha os valores de x em um intervalo de interesse, verificar os valores da função para esses valores de x e encontrar os valores da derivada usando aproximação.

■ **Exemplo 8.2** Derivação de uma função qualquer:

```
function y=funcao(t)
 y=sin(cos(t));
endfunction


x=1:0.01:2;
y=funcao(x);


dx=diff(x);
dy=diff(y);


dfdx=dy./dx
```

■

Neste exemplo, cria-se um vetor x com valores de 1 a 2 com passo de 0,01. Em seguida, cria-se o vetor y que contem a avaliação da função para cada valor de x. Por fim, utiliza-se a função **diff**() para executar as subtrações necessárias em ambos os vetores e armazená-las nas variáveis dx e dy. Finalmente, realiza-se divisão termo a termo dos vetores dy e dx.

8.3 Integração

O calculo de integrais também apresenta funções definidas no Octave. Para integrais indefinidas em polinômios, pode-se utilizar a função **polyint**(), que recebe o vetor com os coeficientes do polinômio a ser integrado e retorna outro vetor com os coeficientes da integral indefinida. Para integrais definidas, tem-se algumas possibilidades, principalmente utilizando o método numérico da quadratura de Simpson, que é um método numérico de integração. O comando que calcula a integral utilizando a quadratura de Simpson é **quad**(). Os parâmetros necessários a essa função são a função a ser integrada (entre aspas simples se ela for definida anteriormente ou na forma de função anônima, mas sem aspas) e os limites de integração.

■ **Exemplo 8.3** Integral simples:

```
function y=funcao(x)
y=log(x);
endfunction


quad('funcao', 0,2)
```

■

Para integrais duplas, utiliza-se a função **dblquad**() da mesma forma que a função mostrada anteriormente, apenas com atenção aos limites de integração nos parâmetros da função: devem escritos primeiro os limites de x e depois os de y, ambos separados por vírgula. Da mesma forma, integrais triplas são calculadas com a função **triplequad**(), seguindo a mesma regra quanto aos limites de integração: primeiro os de x, depois os de y e finalmente os de z.

■ **Exemplo 8.4** Integral tripla:

```
triplequad(@(x,y,z)(x+2*y-e^z), 0,1, 0,2, 0,3)
```

■

Finalmente pode-se deixar o Octave escolher o melhor método de integração numérica utilizando as funções **integral**, **integral2** e **integral3**, que utilizam parâmetros idênticos às funções do método da quadratura e calculam integral simples, duplas e triplas, respectivamente, porém, o método de integração numérica é escolhido de pelo Octave.

9. Recursos Úteis

Neste último capítulo serão tratados alguns recursos que podem facilitar o processo de programação com o Octave, seja simplificando, inserindo informações dentre outros recursos.

9.1 Ponto-e-Vírgula

Durante esta apostila, foram usados exemplos de códigos em que eram utilizados ponto-e-vírgula no fim das linhas de comando. O recurso do ";" é usado quando não se deseja que o Octave exiba na janela de comandos o resultado do comando executado. Desta forma, os resultados exibidos tornam-se mais limpos e de fácil compreensão.

9.2 Comentários

Os comentários no decorrer do código são trechos de código que serão ignorados durante a execução dos comandos, porém eles são úteis para facilitar a compreensão do código ou para testar o efeito de parte do código sem precisar apagá-lo.

Os comentários são inseridos sempre antecedidos por um % ou . Para uma única linha de comentário, apenas o símbolo é necessário, mas para um comentário de várias linhas é necessário colocar o comentário entre chaves e inserir um dos símbolos já citados antes de cada chave.

■ **Exemplo 9.1** Comentários

```
%Este    um exemplo de coment rio linha.


#{
Este e um comentario
em bloco que
ocupa mais de uma
linha do codigo
#}
```

■

É recomendável sempre inserir comentários em todo o código, pois pode facilitar futuras melhorias no código.

9.3 A Função Help

Sempre que for necessário utilizar uma função e surgir alguma dúvida sobre seu funcionamento, pode ser usada a função **help** seguida do nome da função. Ao executar, o Octave dará uma descrição do funcionamento da função em inglês, que pode sanar algumas dúvidas.

9.4 Limpando a Janela de Comandos

A execução de diversos trechos de código, de *scripts* e testes de comandos pode fazer a janela de comandos tornar-se cheia e dificultar a visualização dos resultados desejados. Para solucionar esse problema, é aconselhável a utilização do comando **clc** antes de iniciar a escrita do *script*, que faz a limpeza da janela de comandos sem apagar as variáveis que já estejam na memória.

9.5 Leitura e Escrita de Dados

O Octave permite que o utilizador insira dados durante a compilação de um *script*. A inserção de dados é feita pela função **input()**. Esta função exibe na janela de comandos a mensagem que for inserida entre aspas em seus parênteses e armazena em uma variável o que o utilizador digitar.

■ **Exemplo 9.2** Entrada de valores pelo usuário durante a execução do código.

```
x=input("Esta mensagem e exibida para o utilizador")
```

■

As saídas de dados, ou seja, a exibição de uma mensagem ou resultado para o utilizador é feita por diversas funções. Entre elas, já foi mostrada a utilização da função **print** para salvar gráficos. A função **printf()** é capaz de exibir textos formatados na tela, ou seja, é capaz de exibir textos e valores de variáveis na tela. A exibição de mensagens na tela é interessante para verificar até onde o código está funcional se inserido vária mensagens no decorrer do código. Para exibir valores de variáveis, deve-se utilizar conversores de saída que são posicionados no lugar em que o valor será mostrado. Os principais conversores de saída são mostrados a seguir mostrando o que eles são capazes de imprimir.

%d	Um número inteiro
%f	Um número real
%e	Um número real com notação exponencial
%g	Um número real no formato mais conveniente
%c	Um caractere
%s	Uma string
%%	O símbolo de %

Depois de escrita a mensagem com os conversores posicionados, deve-se listas os nomes das variáveis referentes aos conversores na ordem em que aparecem na mensagem.

■ **Exemplo 9.3** Saída de dados formatados.

```
clc
clear
nome="PET Mec nica";
y=13;
printf("O %s e um dos %d Pet's da UFES",nome, y)
```

■

9.6 Formatos de Exibição

O Octave permite alterar o formato em que os números são exibidos, sem alterar as operações realizadas, tornando o resultado exibido mais adequado para cada situação. Por padrão, o formato

utilizado é o **short** que exibe apenas cinco algarismos, porém, pode-se alterar esse formato a qualquer momento, tendo em vista que depois de alterado, todos os resultados assumirão o mesmo formato até que seja alterado novamente.

A alteração de formato de exibição se dá pela função **format** seguida do formato desejado. Os formatos mais utilizados são exibidos a seguir.

short	5 algarismos
long	16 algarismos
bank	2 casas decimais
short e	5 algarismos e potencia de base 10
long e	16 algarismos e pontencia de base 10
short g ou **long g**	Escolhe entre decimal e notação conforme a necessidade
short eng ou **long eng**	Escolhe entre decimal ou notação com expoente múltiplo de 3
hex	Base hexadecimal
+	Exibe apenas o sinal do número

Referências

MARQUES, Nuno Cavalheiro; MORGADO, Carmen. **Octave: Guia de Estudo**. 2010. Disponível em: http://www.uft.edu.br/engambiental/prof/catalunha/arquivos/octave/octave_Nuno.pdf. Acesso em: 22 abr. 2021.

TEIXEIRA, Sergio Roberto; SODRE, Ulysses; OLIVEIRA, Andrielber da Silva; TAFFOLI, Sônia Ferreira Lopes. **OCTAVE - Uma Introdução**: primeiros contatos com o ambiente de programação numérica octave. Primeiros contatos com o ambiente de programação numérica Octave. 2010. Disponível em: http://www.uel.br/projetos/matessencial/superior/pdfs/octave-final.pdf. Acesso em: 20 maio 2021.

AMBRÓSIO, Camili. **Tutorial do Octave**: gnu-octave versão 2.1.42. GNU-Octave versão 2.1.42. 2005. Disponível em: https://cachoeiro.ifes.edu.br/images/stories/2020/ifes_em_casa/ifes_em_casa_manual_octave.pdf. Acesso em: 15 maio 2021.

PET - ENGENHARIA DE COMPUTAÇÃO (Brasil). Universidade Federal do Espírito Santo. **Mini-curso de MATLAB e Octave para Cálculo Numérico**. Disponível em: http://www.inf.ufes.br/~luciac/cn/MatlabOctave.pdf. Acesso em: 10 maio 2021.

FRANZINI, Guilherme Rosa et al. **Introdução à programação em ambientes MATLAB R e Octave.** 2017. Disponível em: https://edisciplinas.usp.br/pluginfile.php/3482396/mod_resource/content/1/IntroducaoMatlabOctave_v1_2017.pdf. Acesso em: 26 abr. 2021.

ALMEIDA NETO, Fernando Gonçalves de; NASCIMENTO, Vítor Heloiz. **Apostila Introdutória de Octave/Matlab**. 2016. Disponível em: http://www.eletronica.uacsa.ufrpe.br/sites/eletronica.uacsa.ufrpe.br/files/apostila_matlab_uacsa_2016.pdf. Acesso em: 17 abr. 2021.

PET - ENGENHARIA MECÂNICA (Brasil). Universidade Federal do Espírito Santo. **Ma-**

tlab: Básico. Disponível em: https://petengenhariamecanica.ufes.br/sites/petengenhariamecanica.ufes.br/files/field/anexo/apostila.pdf. Acesso em: 03 abr. 2021.