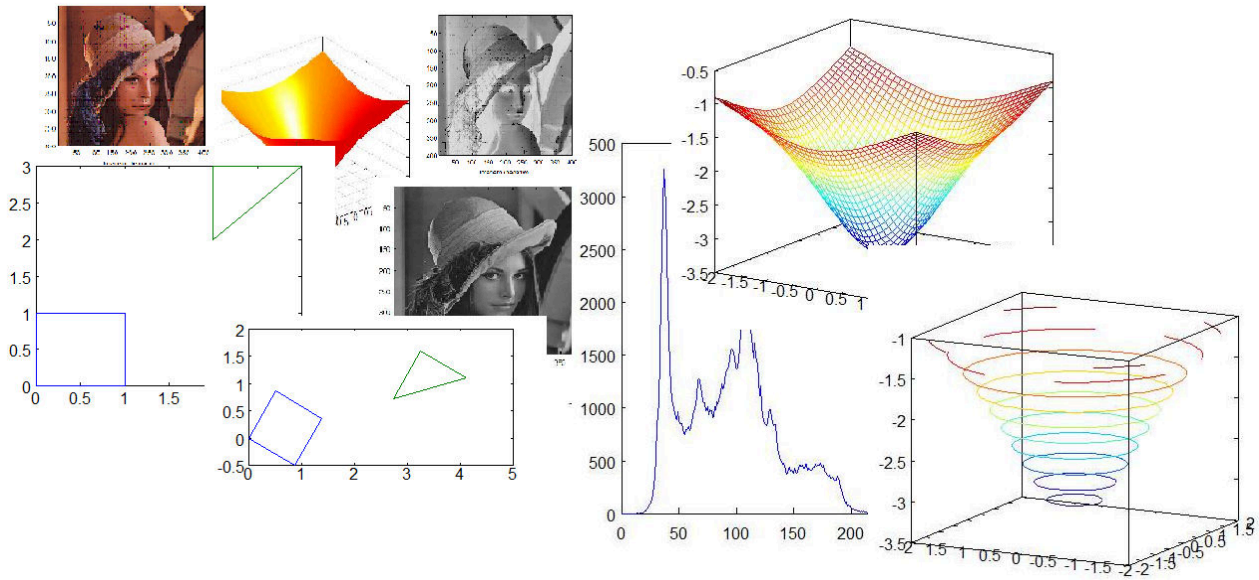
Universidade Estadual Paulista
FCT – Faculdade de Ciências e Tecnologia
Departamento de Cartografia

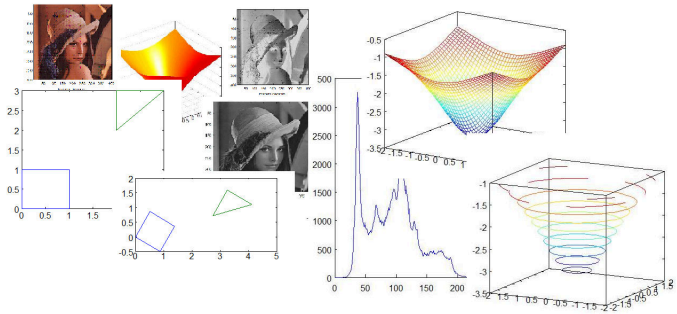
Tutorial do *GNU Octave* / *GNU Octave* Tutorial



Autores:
Mauricio Galo
Paulo de Oliveira Camargo

Presidente Prudente - SP, 2021

Tutorial do *GNU Octave* / *GNU Octave* Tutorial



Autores:
Mauricio Galo
Paulo de Oliveira Camargo
Departamento de Cartografia / FCT - UNESP

2021

Tutorial do *GNU Octave* / *GNU Octave* Tutorial

Este material foi preparado a partir das seguintes referências (em ordem alfabética):

CORAL, A. M.; SANTOS, M. P.; BASTOS, T. D. A.; BORBA, M. *Curso de Matlab*. Universidade Federal de Santa Catarina, Dep. de Eng. de Produção, Programa Especial de Treinamento – PET. Florianópolis – SC, 1999. 29p.

EATON, J. W.; BATEMAN, D.; HAUBERG, S. *GNU Octave* – Edition 3 for Octave version 3.2.3, July, 2007. 672p.

EATON, J. W.; BATEMAN, D.; HAUBERG, S.; WEHBRING, R. *GNU Octave* – Edition 4 for Octave version 4.0.0 – Free Your Numbers, March, 2015. 966p.

HANSELMAN, D.; LITTLEFIELD, B. *Matlab - Versão do Estudante:* guia do usuário **(**Tradução). São Paulo: Makron Books, 1997. 305p.

McANDREW, A. *Introduction to Digital Image Processing with MATLAB®*. Thomson Course Technology, 2004. 509p. ISBN: 0-534-40011-6

PAGAMISSE, A.; SOUZA, L. H. G. *Introdução ao Software Octave*. Semana de Cursos de Matemática, Estatística e Computação, 25-29 de agosto de 2003, FCT/UNESP, Presidente Prudente, 2003. 54p.

SIGMON, K. *MATLAB Primer* – Third Edition. Department of Mathematics, University of Florida, Gainesville, 1993. 35p. Disponível em http://www.math.toronto.edu/mpugh/primer.pdf. Acesso em Março/2021.

ZERI, L. M. M. *Apostila de Matlab*. Instituto Nacional de Pesquisas Espaciais – INPE, 2001. 19p.

1. INTRODUÇÃO

1.1. O que é *GNU Octave*

O *GNU Octave* é um aplicativo que foi originalmente desenvolvido com o propósito didático, mais especificamente para o projeto de reatores químicos e surgiu a partir da intenção de criar um aplicativo no qual a programação fosse mais rápida do que nas demais linguagens.

Segundo os autores, os alunos poderiam dedicar mais tempo na análise e solução dos problemas de química, do que especificamente na implementação.

O seu desenvolvimento começou por volta de 1988 e um de seus idealizadores foi **John W. Eaton**, além de **James B. Rawlings** da Universidade de Wisconsin-Madison **e John G. Ekerdt** da Universidade do Texas. Atualmente o desenvolvimento é feito por vários profissionais, de diferentes países, sendo a sua distribuição feita de acordo com a licença GLP (*GNU General Public License*).

Onde encontrar (**páginas oficiais**):

http://www.gnu.org/software/octave/

http://octave.sourceforge.io/

Características básicas:

- Domínio público;
- Possui vários comandos que são semelhantes ao MATLAB® e SciLab®;
- É um software multiplataforma uma vez são disponíveis versões para diferentes sistemas operacionais: Linux, Unix, Windows, Mac, etc.

Últimas versões (**estáveis**):

4.4.0 (Lançada em 30/Abril/2018)

4.4.1 (Lançada em 9/Agosto/2018)

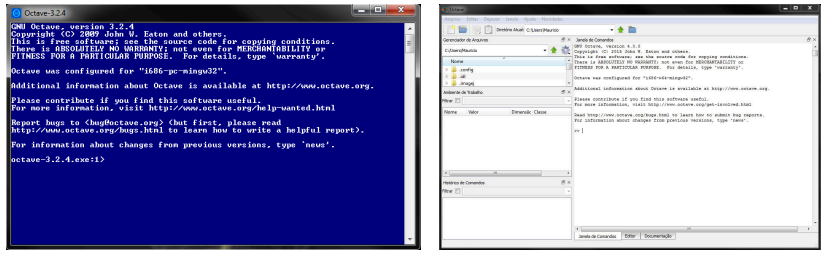
5.1.0 (Lançada em 1/Março/2019)

5.2.0 (Lançada em 31/Janeiro/2020)

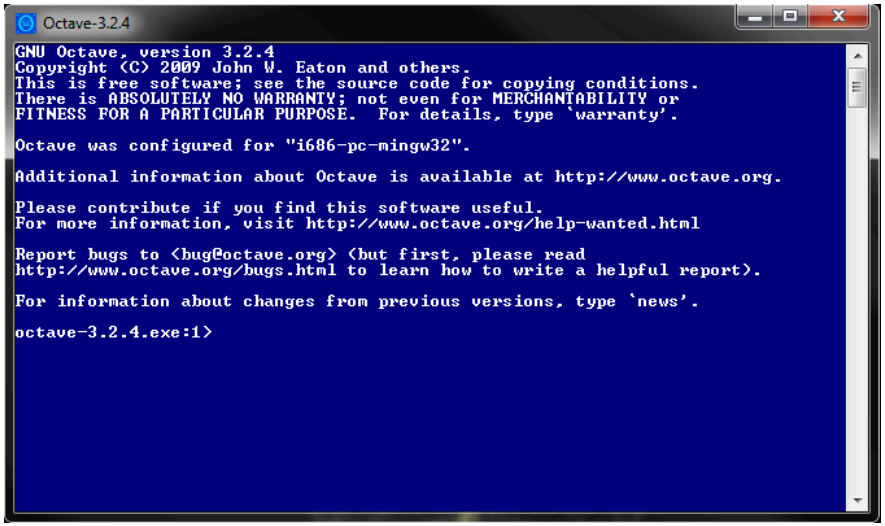
6.2.0 (Lançada em 20/Fevereiro/2021)

1.2. Ambiente de trabalho e comandos básicos do sistema

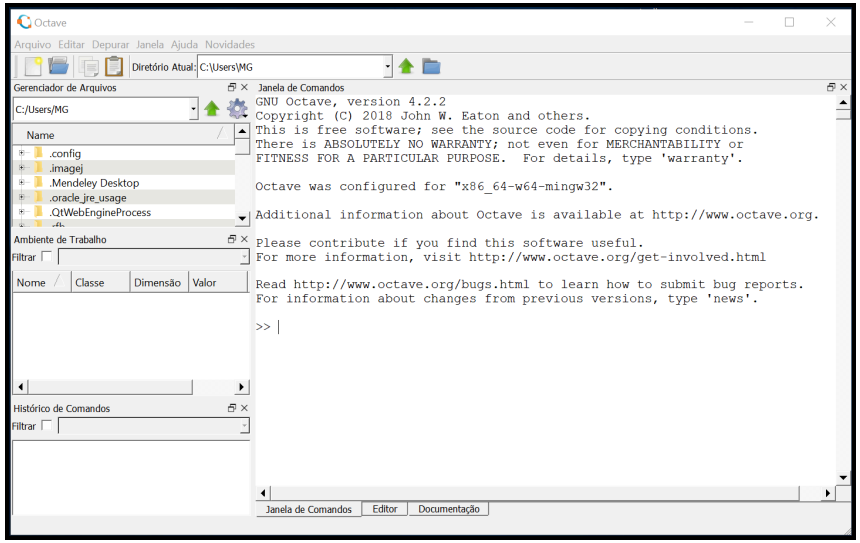
Desde o inicio do desenvolvimento do *GNU Octave*, em 1988, foram lançadas inúmeras versões, para as diferentes plataformas[1]. Na maior parte das versões a única possibilidade de trabalho era via uma interface não gráfica. Há poucos anos a opção de utilização de uma interface gráfica foi possível, tornando mais prática e amigável sua atualização. Na figura abaixo são mostradas duas interfaces (versões 3.2.4 e 4.0.0, resp.).



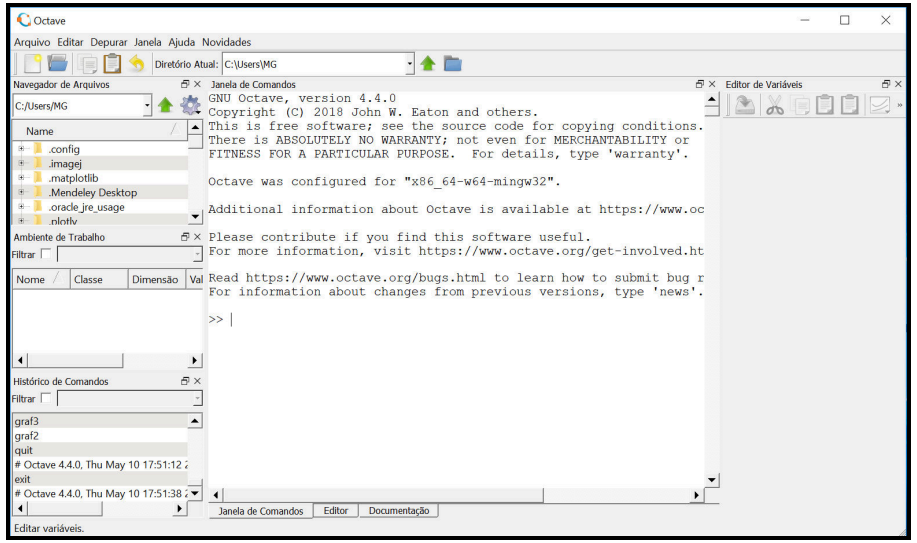
[1] Consulte a página "ftp://ftp.gnu.org/gnu/octave/" para ver algumas destas versões.



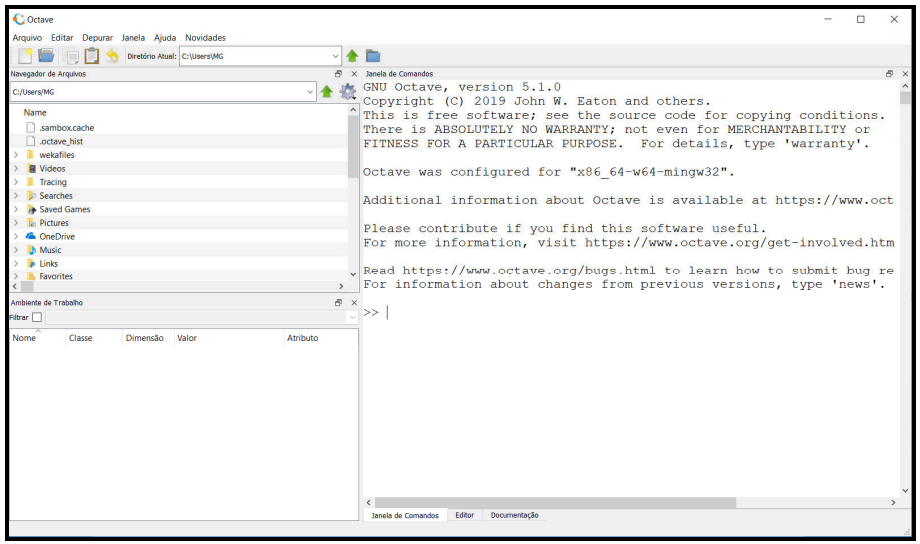
Interface não gráfica – *GNU Octave* versão 3.2.4.



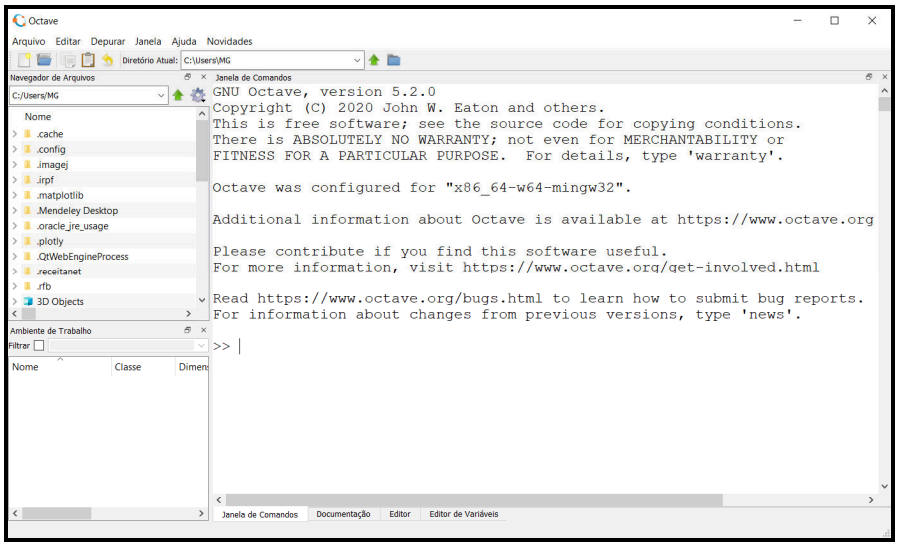
Interface gráfica do *GNU Octave* versão 4.2.2, lançada em março de 2018.



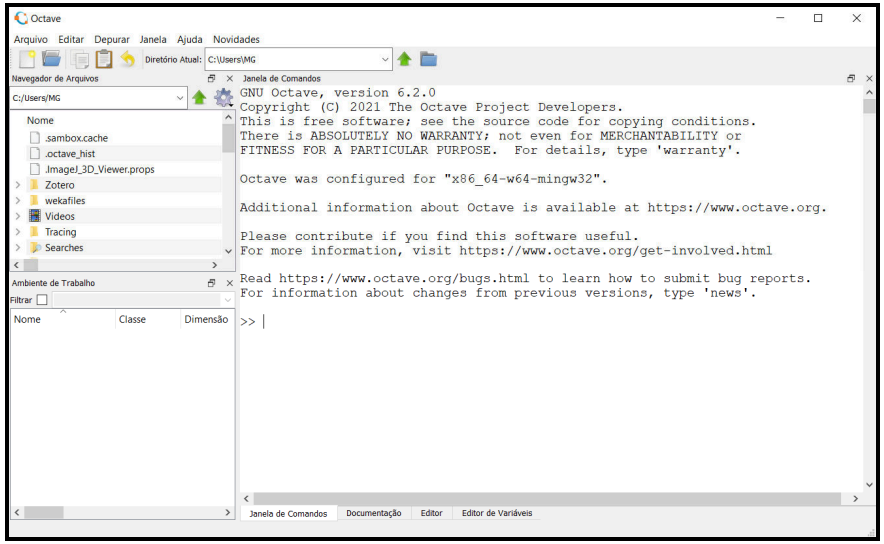
Interface gráfica do *GNU Octave* versão 4.4.0, lançada em abril de 2018.



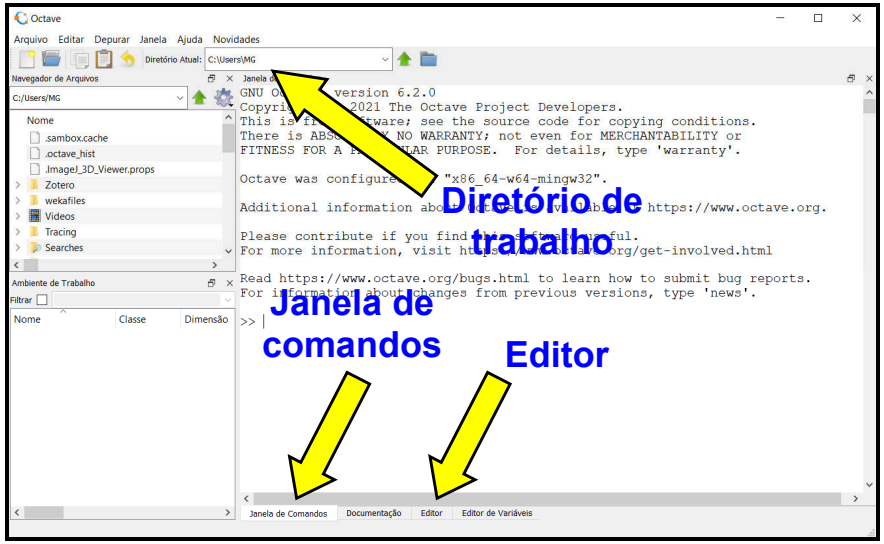
Interface gráfica do *GNU Octave* versão 5.1.0, lançada em março de 2019.



Interface gráfica do *GNU Octave* versão 5.2.0, lançada em janeiro de 2020.

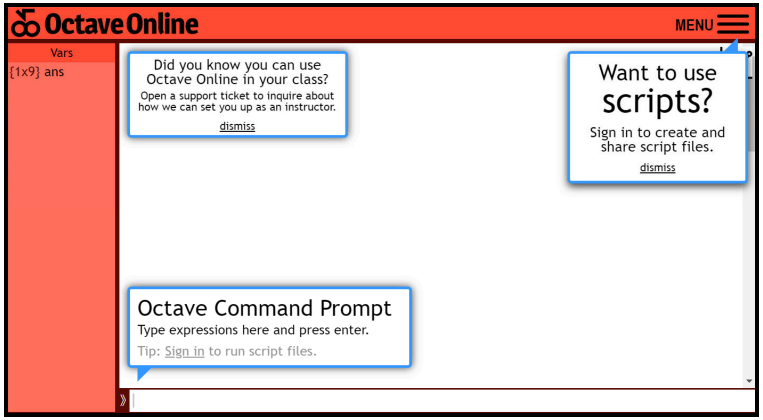


Interface gráfica do *GNU Octave* versão 6.2.0, lançada em fevereiro de 2021.



Interface gráfica do *GNU Octave* versão 6.2.0.

Além dessas versões, que podem ser instaladas e executadas no modo *off-line*, também é possível executar o *GNU Octave* de modo *online* (ver *link* abaixo).



Interface gráfica do *GNU Octave* executado no modo *online*.
Fonte: https://octave-online.net

Alguns comandos do *GNU Octave* são básicos e semelhantes a outros usados em sistemas operacionais como DOS, Unix, Linux, dentre outros. A tabela seguinte mostra alguns destes comandos, que podem se acionados a partir do *prompt* do *GNU Octave*.

cd	Troca o diretório de trabalho atual
dir	Lista o conteúdo do diretório atual
pwd	Mostra o *path* do diretório atual de trabalho
delete	Exclui arquivo Exemplo de uso: delete *arquivo.ext*
type	Mostra o conteúdo de arquivos texto Exemplo de uso: type *arquivo.ext*
what	Lista arquivos específicos do *GNU Octave* no diretório corrente
help	Ajuda do *GNU Octave* Exemplo de uso: help *comando*
quit ou exit	Fechar o aplicativo e sair do sistema

Alguns outros comandos: mkdir, rmdir, etc.

1.3. Pacotes / Bibliotecas adicionais do *GNU Octave*

O *GNU Octave* é um software em constante desenvolvimento e após a instalação do aplicativo, outros pacotes adicionais podem ser instalados. Para ter uma lista dos pacotes disponíveis para o *GNU Octave* sugere-se a consulta à seguinte página http://octave.sourceforge.net/.

Antes de instalar qualquer pacote adicional, sugere-se ativar o comando "pkg list" no *prompt* do *GNU Octave* para ver todos os pacotes que são pré-instalados junto com a instalação do aplicativo.

Lista de pacotes/bibliotecas disponíveis para o usuário ao instalar o aplicativo *GNU Octave* versão 6.2.0.

audio	instrument-control	queueing
communications	interval	signal
control	io	sockets
data-smoothing	linear-algebra	sparsersb
database	lssa	specfun
dataframe	ltfat	splines
dicom	mapping	statistics
financial	miscellaneous	stk
fits	nan	strings
fuzzy-logic-toolkit	netcdf	struct
ga	nurbs	symbolic
general	ocs	tisean
generate_html	odepkg	tsa
geometry	optim	video
gsl	optiminterp	windows
image	quaternion	zeromq

Para a instalação de algum pacote adicional, de nome ***package_file_name.tar.gz***, basta ativar o seguinte comando no *prompt* do *GNU Octave*:

pkg install ***package_file_name.tar.gz***

2. INICIANDO O *GNU Octave*

2.1. Variáveis

O *GNU Octave* trabalha essencialmente com um tipo de variável: matriz, que pode conter números (complexos ou não) e textos. Em alguns casos, um tratamento especial é dado a uma matriz 1 x 1 (escalar) ou a matrizes 1 x n ou n x 1 (vetores).

2.1.1. Entrando com valores

No *GNU Octave* não é necessário declarar as variáveis e os respectivos tipos (inteiro, char, double, etc,) para iniciá-las, como é feito em outras linguagens de programação (C/C++, dentre outras). Ao atribuir valores numéricos (ou alfanuméricos) a uma variável, o programa aloca a memória automaticamente.

A maneira mais fácil de entrar com uma pequena quantidade de valores é digitando diretamente os dados:

- envolva os elementos com colchetes, [ ]; (Para matrizes e vetor)
- separe cada elemento com espaços ou vírgulas;
- use ponto-e-vírgula (;) para indicar fim da linha.

Por exemplo, para entrar com a matriz abaixo na memória do computador, e guardá-la na variável A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Basta digitar:

```
» A=[1 2 3; 4 5 6; 7 8 9]
```


ou

```
» A=[1, 2, 3; 4, 5, 6; 7, 8, 9]
```

Resultado: A =

```
1   2   3
4   5   6
7   8   9
```

OBS: Para que o computador realize a operação e não mostre a saída, basta terminar a expressão com ponto-e-vírgula (**;**). Isto é muito útil para evitar que o computador mostre resultados de cálculos intermediários desnecessários, e para acelerar as operações.

2.1.2. Variáveis reservadas

Existem algumas variáveis que são intrínsecas ao *GNU Octave*, como por exemplo:

help	Ajuda do *GNU Octave*
ans	Nome de variável padrão usado para resultados
eps	Precisão do número real (ponto flutuante)
nan	*Not a Number* (indeterminação: 0/0)
realmax	Maior número real positivo utilizável
realmin	Menor real positivo utilizável
inf	Infinito: 1/0
computer	Tipo de computador
pi	3,14159265358979
i, j	Unidade imaginária (i^2=-1)
version	Versão do *GNU Octave*

2.2. Expressões e comandos básicos

O *GNU Octave*, assim como o MATLAB, são linguagens de alto nível e interpretadas. Nestas linguagens o sistema avalia as expressões digitadas, que são geralmente na forma:

variável = expressão

Os números reais são escritos em notação decimal e para criar números complexos basta escrever *i* (ou *j*) depois da parte imaginária. Alguns exemplos de números permitidos são mostrados abaixo:

1/3	-99	.0001
9.63973	1.602E-20	6.025E23
3 + 2i	-3.1459i	3E5j

Todo cálculo é realizado com todas as casas decimais (eps), embora os números mostrados ocultem algumas dessas casas. Para mudar o formato de saída dos números, pode-se usar o comando format.

O formato "*default*", chamado de formato short, mostra aproximadamente 5 dígitos significativos ou usam notação científica. Exemplo:

```
>> x = [4/3 1.2345e-6]
```

é mostrada, para cada formato usado, da seguinte maneira:

format short	1.3333e+000 1.2345e-006
format long	1.333333333333333e+000 1.234500000000000e-006
format hex	3ff5555555555555 3eb4b6231abfd271
format rat	4/3 9/7290401 (Aproximação racional)
format bank	1.33 0.00
format +	++

O formato + é uma maneira compacta de mostrar matrizes de grandes dimensões. Os símbolos "+", "-", e "espaço em branco" são mostrados, respectivamente para elementos positivos, elementos negativos e zeros.

Podem-se construir expressões com os operadores aritméticos usuais:

+	adição	/ e \	Divisão
-	subtração	^	Potenciação
*	multiplicação	'	matriz transposta

O *GNU Octave* possui um conjunto de funções matemáticas elementares, com seno (sin), tangente (tan), logaritmo (log10), etc. Por exemplo, para calcular o seno de 30 e guardar na variável x pode-se escrever:

```
» x=sin(30)
x=-0.98803                                    (para format short)
```


Funções Elementares (colocadas em ordem alfabética)

abs(x)	Valor absoluto ou módulo de um número
acos(x)	Arco co-seno
angle(x)	Calcula o ângulo de fase (em radianos) para números complexos
asin(x)	Arco seno
atan(x)	Arco tangente
cos(x)	Cosseno
cross(a,b)	Produto vetorial dos vetores a e b
exp(x)	Exponencial (e^x)
inv(x)	Matriz inversa da matriz x
log(x)	Logaritmo natural ($\log_e x$)
log10(x)	Logaritmo na base 10
max(x)	Maior elemento em x

Funções Elementares (continuação)

mean(x)	Média de x
min(x)	Menor elemento em x
sin(x)	Função seno
sqrt(x)	Raiz quadrada
std(x)	Desvio padrão
sum(x)	Soma dos elementos de x
tan(x)	Tangente

Algumas dessas operações podem ser aplicadas a vetores ou matrizes. Para a função std(x), por exemplo, se o argumento x for um vetor o resultado será o desvio padrão desse vetor. Se x for uma matriz, será calculado o desvio padrão para **cada coluna** dessa matriz. O mesmo ocorre com as funções sum, max e mean, entre outras.

Usando os comandos vistos, determinar a média, desvio padrão, a soma dos elementos, o valor máximo e mínimo do vetor V abaixo:

```
» V=[1 2 3 4 5 6 7 8 9]
```


2.2.1. Comandos

Ao sair do *GNU Octave* (através do comando quit ou exit) todas as variáveis do *workspace* são perdidas, a menos que sejam salvas usando o comando save.

Como usar o comando save, bem como outros relacionados:

save *nome_de_arquivo nome_da(s)_variável(is)*

Salva somente as variáveis especificadas.

load *nome_de_arquivo*

Carrega as informações salvas.

clear *nome_da_variável*

Apaga a variável especificada do *workspace*.

O clear (sem variável) apaga todas as variáveis.

Um dos comandos mais úteis no *GNU Octave*, bem como em diversos aplicativos, é o help, que fornece ajuda *on-line* sobre qualquer outro comando. Por exemplo, para obter ajuda sobre o comando who:

```
» help who
```

WHO – Lista as variáveis no *workspace* atual.

WHOS – Lista as variáveis e informações adicionais sobre elas.

Outro comando útil: lookfor. Este comando procura e lista todas as funções do *GNU Octave* que contém a palavra-chave especificada.

```
» lookfor mean
```

Resultado: mean, meansq e mean2.

2.2.2. Exercícios

1) Armazene no *workspace* os seguintes valores:

```
a = 3
b = -2.123
c = 4*3
d = [1 2 3 4]
e = [1; 2; 3; 4]
```

Utilize os comandos who e whos e observe as diferenças.

Ative o comando clear, e digite novamente o comando whos.

2) Armazene no *workspace* os seguintes valores:

```
a = 3.452
b = -25.123
c = 4*pi
```

3) Verifique o resultado das seguintes das seguintes operações:

a) (a + b) c
b) (c – b) a / b
c) tan(c) – asin(b)
d) sin(cos(tan(c)))

Efetuar os cálculos (formato padrão), depois mudar para o formato long

Comandos: save, quit (ou exit), load, clear, whos.

4) Atribua as seguintes expressões às variáveis:

a) 4.12 a – pi/c para x
b) log(a + 40.1353) para y
c) log(a) para z

5) Salve as variáveis x, y, z em um arquivo chamado exemplo.mat.

6) Saia do *GNU Octave*, entre novamente e carregue as variáveis salvas anteriormente.

7) Apague a variável z e verifique se ela foi realmente apagada.

3. OPERAÇÕES COM MATRIZES E VETORES

O *GNU Octave* permite a manipulação de linhas, colunas, elementos individuais e partes de matrizes.

3.1. Geração de vetores

A geração de uma seqüência de números pode ser feita utilizando:

```
> x = 1:5    ou          > x = 1:1:5
```

gera um vetor linha contendo os números de 1 a 5 com incremento unitário.

Outros exemplos com incrementos diferentes de um:

```
> y = 0:pi/4:pi
```

que resulta em:

```
y =
  0.0000   0.7854   1.5708   2.3562   3.1416
```

Incrementos negativos também são possíveis.

```
> z = 6 : -1 : 1
z =
  6  5  4  3  2  1
```

Pode-se também gerar vetores usando a função linspace.

Sintaxe: linspace (*inicio*, *fim*, *número de elementos*)

Exemplo de uso:

```
> k = linspace (0, 1, 6)
```

```
k =
   0  0.2000  0.4000  0.6000  0.8000  1.0000
```


3.2. Elementos das matrizes

Um elemento específico da matriz pode ser acessado especificando a linha e a coluna do elemento desejado, fazendo A(*linha, coluna*). Por exemplo, dada a matriz A:

```
A =
    1 2  3
    4 5  6
    7 8  9
```

Qual o resultado da seguinte operação?

```
>> A(3,3) = A(1,3) + A(3,1)
```


Resulta em:

```
A =
    1 2  3
    4 5  6
    7 8  10
```


"Extração" de submatrizes de uma dada matriz

```
A =
   92  99  11  18
   98  80  17  14
   14  81  88  20
```

```
>> B = A(2:3,4)
```

Armazena em B os elementos das linhas 2 e 3 da coluna 4.

```
B =
   14
   20
```

```
>> C = A(2:3,2:4)
C =
   80  17  14
   81  88  20
```

C é uma submatriz 2x4, formada pelas linhas 2 e 3 e colunas 2 a 4 da matriz A.

```
>> D = triu(A)
D =
      92  99  11  18
      00  88  17  14
      00  00  88  20
```

<< Extrai a triangular superior.

Geração de algumas matrizes especiais:

Exemplo 1: B uma matriz 10x10 unitária.

```
>> B = ones (10)
```

Exemplo 2: C uma matriz de dimensão 8x8 com elementos nulos.

```
>> C = zeros (8)
```

Matriz identidade: eye

Matriz diagonal: diag << Extrai a diagonal de uma dada matriz.

Matriz de números randômicos: rand

3.3. Operações com matrizes

Operações matemáticas simples (adição, subtração, divisão e multiplicação) envolvendo matrizes são semelhantes às operações com escalares.

```
» m=[8 1 6; 3 5 7; 4 9 2];

» 3*m

ans =
   24    3   18
    9   15   21
   12   27    6
```



```
» m+100
ans =
   108  101  106
   103  105  107
   104  109  102
```

Nas operações entre matrizes devem ser respeitadas as regras usuais da matemática quanto ao número de linhas e colunas que duas matrizes devem ter para serem somadas, multiplicadas, etc. No entanto existem operações especiais. Sendo A=[a_1 a_2 ... a_n] e B=[b_1 b_2 ... b_n] duas matrizes, então:

- A./B = [a_1/b_1 a_2/b_2 ... a_n/b_n];
- A.*B = [$a_1 \cdot b_1$ $a_2 \cdot b_2$ ... $a_n \cdot b_n$];
- A.^B = [a_1^b_1 a_2^b_2 ... a_n^b_n];

Alguns comandos adicionais para a operação com matrizes:

inv	Matriz inversa de uma dada matriz Sintaxe: inv(*matriz*)
det	Calcula o determinante (D) de uma matriz bem como o recíproco do número de condição (RCOND) Sintaxe: [D, RCOND]=det(*matriz*)
trace	Traço de uma matriz Sintaxe: trace(*matriz*)
cond	Número de condição de uma matriz. Sintaxe: cond(*matriz*)
rank	*Rank* de uma matriz Sintaxe: rank(*matriz*)

3.4. Exercícios

1) Dadas as seguintes matrizes e vetores:

```
» A = [1 2 3; 4 10 6; 7 8 19]
» B = [4 5 6; 1 2 3; 8 7 6]
» C = [4 5 6]
» L = [4; 5; 6]
```

2) Calcule:

a) D = A+B

b) E = A*B

c) F = A-B

d) G = B^T-A

e) H = A^TA

f) I = A C^T

g) Determinar a inversa de A e salvar em D

h) Produto D vezes A

i) Autovalor e Autovetor de A

j) Calcule o traço e o determinante de A

k) Resolver o sistema AX = L

4. GRÁFICOS

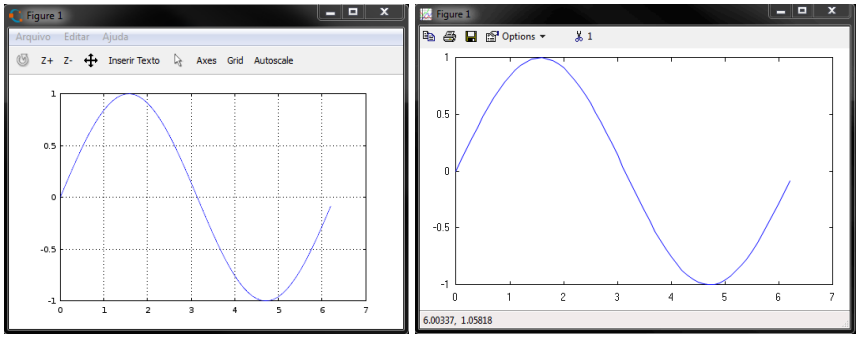
A construção de gráficos no *GNU Octave* é bem simples e a maior parte dos comandos é muito parecida com os usados pelo MATLAB, sendo possível a geração de gráficos bidimensionais ou tridimensionais.

Existe no *GNU Octave* um grande conjunto de comandos que permite a construção de gráficos.

Na versão atual a saída gráfica é, por *default*, criada usando o OpenGL e Qt. Caso as bibliotecas do OpenGL não estejam disponíveis o GnuPlot é utilizado.

Caso o usuário queira saber qual a interface em uso basta digitar *graphics_toolkit* no *prompt* do *GNU Octave*. Para modificar a interface de saída o comando *graphics_toolkit* (*opção*) pode ser usado.

As imagens abaixo mostram o mesmo gráfico gerado com as opções "fltk" e "gnuplot", respectivamente.



4.1. Comandos gráficos básicos

Na tabela seguinte são mostrados alguns comandos básicos que permitem a geração de gráficos bidimensionais no *GNU Octave*.

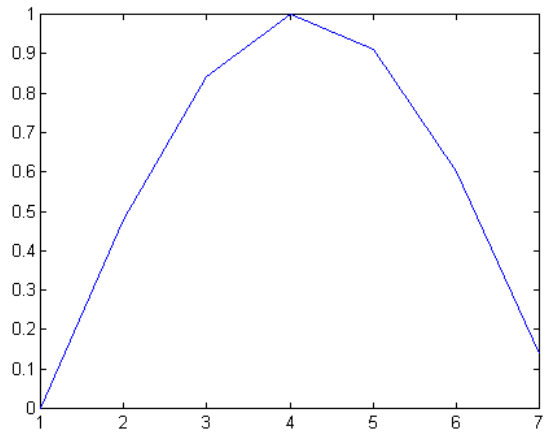
plot	Visualização de gráficos lineares no plano
bar	Gráfico de barras
stem	Sequência discreta
stairs	Plotar em degraus
errorbar	Plotar barra de erros
hist	Plotar histograma
comet	Plotar com trajetória de cometa

Se **y** é um vetor, plot(y) produz um gráfico linear com o índice dos elementos de **y** na abscissa e os elementos de **y** na ordenada.

Por exemplo, para plotar os números [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0.14], entre com o vetor **y** e execute o comando plot, como mostrado:

```
>> y = [0.0  0.48  0.84  1.0  0.91  0.6  0.14];
>> plot(y)
```

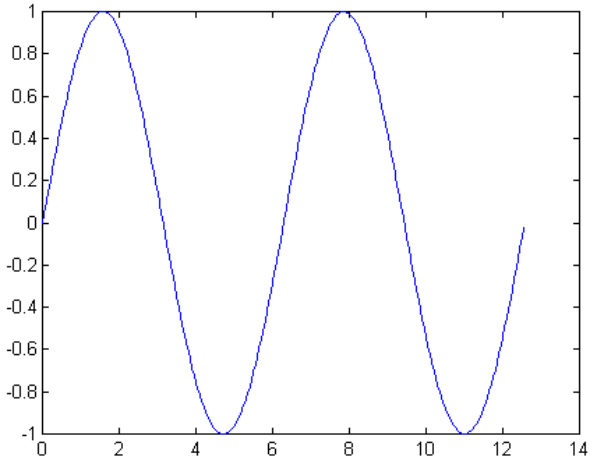

Resultado:



Se t e y são vetores com dimensões iguais, o comando plot(t,y) produz um gráfico bidimensional dos elementos de t versos os elementos de y, por exemplo

```
>> t = 0:0.05:4*pi;
>> y = sin(t);
>> plot(t,y)
```

Resultado:



O *GNU Octave* pode também plotar múltiplas linhas em apenas um gráfico.

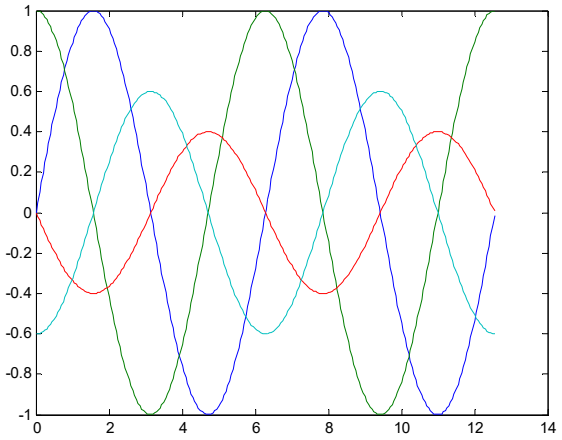
Exemplo de uso do comando plot com múltiplos argumentos.

Exemplo:

```
>> plot(t,sin(t),t,cos(t),t,0.4*sin(t+pi),t,0.6*cos(t+pi))
```

Obs : Observe que as informações são fornecidas aos pares, sendo um desles o que será representado na abscissa e o seguinte na ordenada, e assim sucessivamente.

Resultado:



4.2. Estilos de Linha e Símbolo

Os tipos de linhas, símbolos e cores usadas para mostrar os gráficos podem ser controlados. Seguem alguns exemplos, seguidos dos respectivos resultados (próxima página):

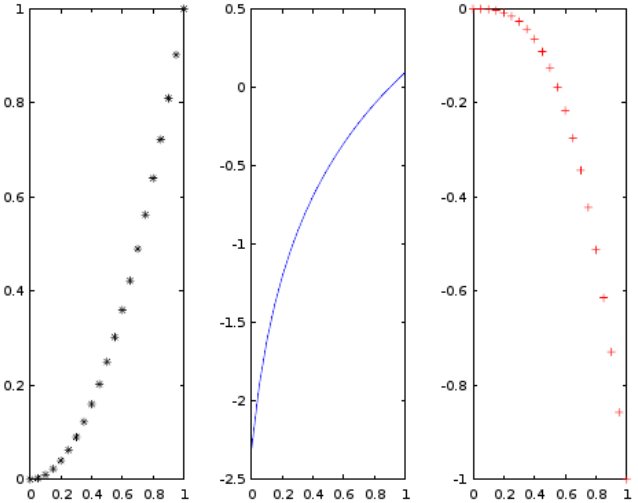
```
>> x = 0:0.05:1;
>> subplot(1,3,1);
>> plot(x,x.^2,'*k')
>> subplot(1,3,2);
>> plot(x,log(x+0.1),'-b')
>> subplot(1,3,3);
>> plot(x,-x.^3,'+r')
```

Símbolo: * Cor: **Preto**

Símbolo: - Cor: **Azul**

Símbolo: + Cor: **Vermelha**

Resultado:



Outros tipos de linhas, pontos e cores também podem ser usados:

TIPO DE LINHA	
-	_____

Mais opções: '--', ':', '-.'

TIPO DE PONTO	
.	. . . . .
*	**********
o	oooooooooooo
+	++++++++++
X	x x x x x x x
^	^ ^ ^ ^ ^ ^ ^

Mais opções: 's', 'd', 'p', 'h'.

CORES	
y	amarelo
m	magenta
c	cian
r	vermelho
g	verde
b	azul
w	branco
k	preto

4.3. Exercícios

```
>> t = 0:0.5:4*pi;
>> y = sin(t);
```

Criar os seguintes gráficos:

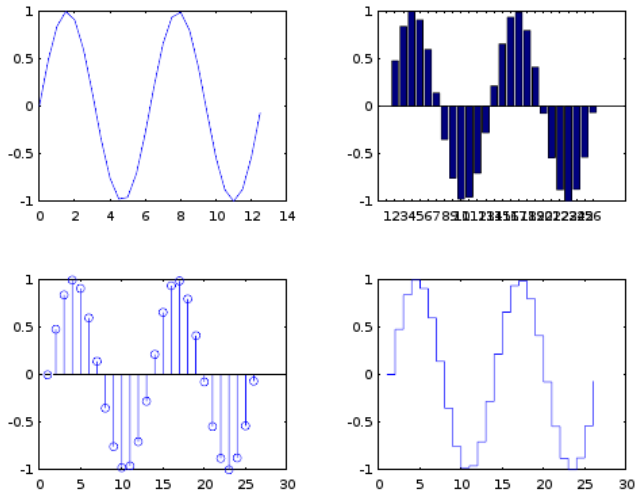
Gráfico simples (plot)

Gráfico de barras (bar)

Sequência discreta (stem)

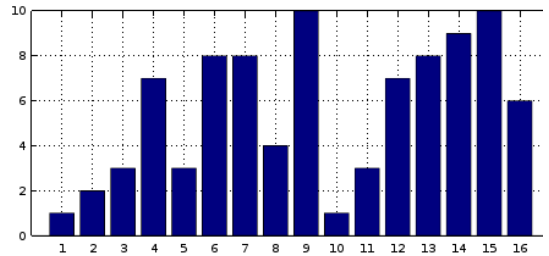
Escada (stairs)

Resultado:

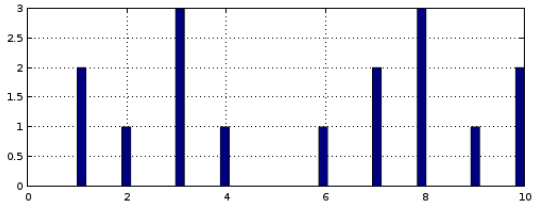


Dado o vetor: y= [1 2 3 7 3 8 8 4 10 1 3 7 8 9 10 6] construa o gráfico de barras bem como o histograma (hist) de frequências.

Resultados: Gráfico de barras e histograma de frequências, respectivamente.



```
>> bar(y);
>> grid('on');
```

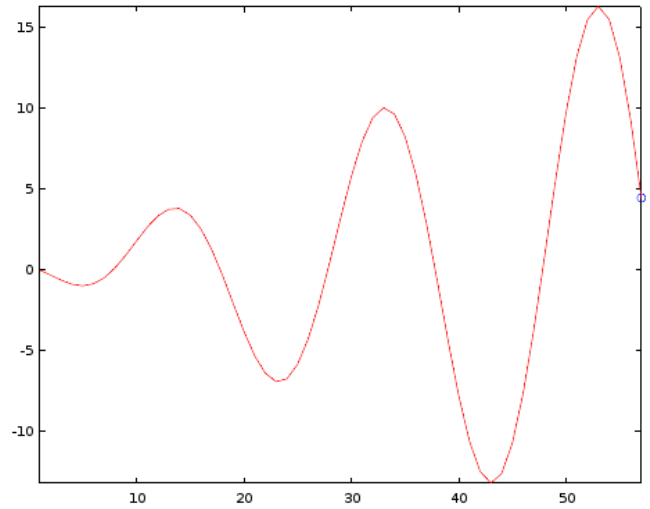


```
>> hist(y,50);
>> grid('on');
```


Verificar o resultado do uso dos seguintes comandos:

```
>> t=1:pi/10:6*pi;
>> y=(1-t).*sin(t);
>> comet(y)
```


Resultado final:



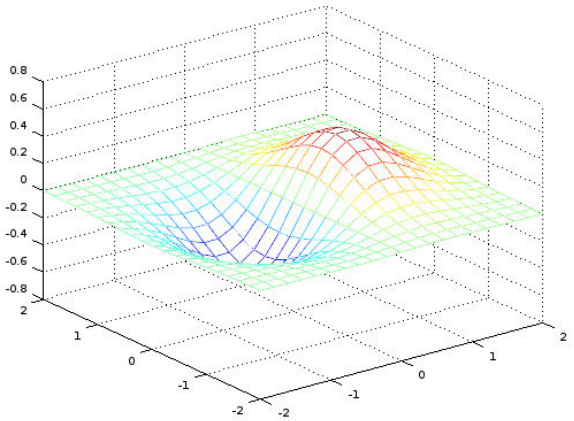
4.4. Geração de Gráficos Tridimensionais e Isolinhas

Estes são alguns comandos para gerar gráficos tridimensionais e contornos.

plot3	Mostrar gráfico no espaço 3D
contour	Permite visualizar curvas de nível no plano
contour3	Permite visualizar curvas de nível no espaço 3D
meshgrid	Permite a criação de estruturas (2D e 3D) para representação 3D
mesh	Permite plotar malhas 3D
meshc	Combinação mesh/contour
surf	Plotar superfície 3D
surfc	Combinação surf/contour
slice	Plot visualização volumétrica
cylinder	Gerar cilindro
sphere	Gerar esfera

O comando mesh(x,y,z) cria uma perspectiva tridimensional plotando os elementos da matriz z em relação ao plano definindo pelas matrizes x e y. Exemplo de uso:

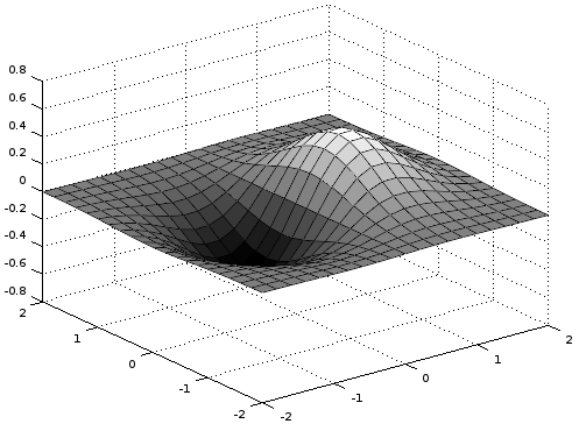
```
>> t = -2:0.2:2;
>> [x, y] = meshgrid(t, t);
>> z = x.* exp(-x.^2 - y.^2);
>> mesh(x,y,z)
```

Verifique as seguintes opções:

```
>> surf(x, y, z)
>> colormap(gray);
```


Resultado:

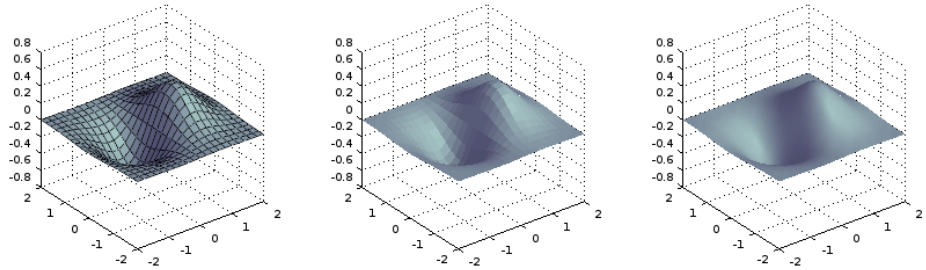


Alguns outros paletes: autumn, bone, cool, copper, cubehelix, flag, gray, hot, hsv, jet, lines, ocean, pink, prism, rainbow, spring, summer, white, winter, …

Outros exemplos de superfícies (ainda usando a mesma função) no qual se considera que a superfície é iluminada por uma fonte de luz, dando o efeito de "sombreamento". Seguem algumas das alternativas:

```
>> colormap(bone)
>> surfl(x, y, z), shading faceted

>> colormap(bone)
>> surfl(x, y, z), shading flat

>> colormap(bone)
>> surfl(x, y, z), shading interp
```


Resultados:



(a)... ..(b) (c)

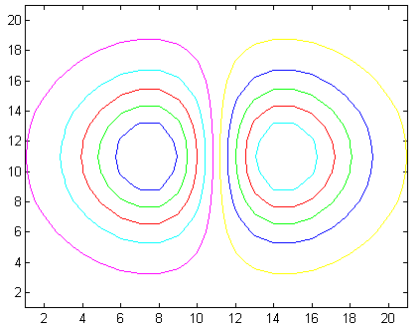
(a) Sombreamento interpolado, com as linhas de transição visíveis (shading faceted)

(b) Sombreamento interpolado mostrando as faces planas (shading flat)

(c) Sombreamento interpolado (shading interp)

Curvas de nível

O comando contour(z,10) mostra a projeção da superfície definida (pela função z), no plano cartesiano xy, com 10 iso-linhas. A figura seguinte mostra o resultado para a superfície anterior.



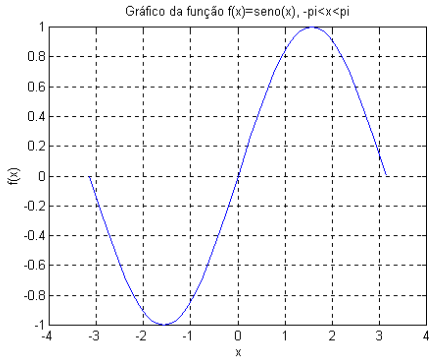
4.5. Anotações no Gráfico

O *GNU Octave* possui alguns comandos que permitem adicionar informações em um gráfico, como por exemplo:

title	Título do gráfico
xlabel	Título do eixo X
ylabel	Título do eixo Y
zlabel	Título do eixo Z
text	Inserir anotação no gráfico
gtext	Inserir anotação com o “mouse”, de modo interativo
grid	Linhas de grade

Exemplos:

```
>> fplot('sin', [-pi pi]);
>> title 'Gráfico da função f(x)=seno(x), -pi<x<pi';
>> xlabel 'x';
>> ylabel 'f(x)';
>> grid ('on');
>> text(1,.2,'Curva sin(x)');
>> gtext 'Sin(x)';
```

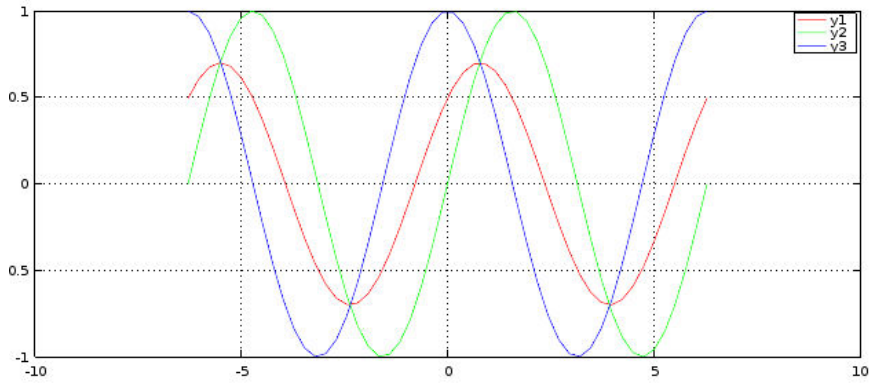
Exemplo do comando fplot para múltiplas funções

```
fplot('[tan(x),sin(x),cos(x)]',2*pi*[-1 1 -1 1])
```


Uma outra possibilidade para a visualização de múltiplas funções:

```
>> figure(2)
>> t=linspace(-2*pi,2*pi,50);
>> y1=tan(t);
>> y2=sin(t);
>> y3=cos(t);
>>
>> plot(t,y1,"r");
>> hold on
>> plot(t,y2,"g");
>> plot(t,y3,"b");
>> legend('y1','y2','y3');
```

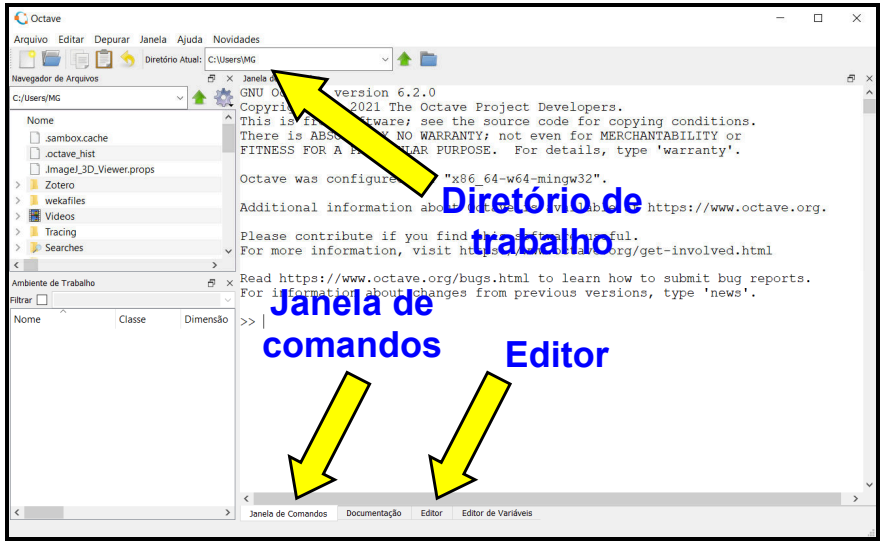

Resultado:



5. ARQUIVOS *SCRIPT* (.m)

Os comandos do *GNU Octave* são normalmente digitados na área de trabalho, onde cada linha de comando é introduzida e imediatamente processada. O *GNU Octave* é também capaz de executar sequências de comandos armazenadas em arquivos ASCII com extensão m. Estes arquivos podem ser criados por alguns aplicativos como *Notepad++*, *Notepad*, *Textpad*, *Sublime Text*, etc.

Uma outra possibilidade para criar e editar estes arquivos é por meio do editor do próprio ambiente de trabalho do *GNU Octave*. Como pode-se notar na próxima figura, tem-se na parte inferior da tela a aba Editor. É sugerido que antes de criar o arquivo, com extensão "m", que seja escolhido e definido o diretório de trabalho, como indicado na próxima figura.



GUI do *GNU Octave* versão 6.2.0, onde são destacados alguns elementos.

Na sequência são apresentados alguns comandos e declarações especiais para serem usados nos arquivos *script*, por exemplo:

Comando	Função
% ou #	Inserir um comentário no arquivo ".m"
clear	Apagar todos os dados da memória
input	Usado para a entrada de dados a partir da janela de comando
pause	Provoca uma pausa na execução do arquivo até que qualquer tecla seja digitada
clc	Limpar a janela de comando
figure(n)	Abrir e mostrar a janela gráfica de número n
disp('...')	Mostrar no ambiente de trabalho o texto colocado entre '...'
close	Fechar todas as janelas gráficas

Exemplo de arquivo *script*

```
#Mostrar a funcao y=ax^2 + bx + c no intervalo -5<x<5
  clear;
  aux='s';
  while aux=='s',
     clc
     disp(' ');
     disp(' Dada a equação y=ax^2 + bx + c ...'); disp(' ');
     a=input('Digite o valor de a = ');
     b=input('Digite o valor de b = ');
     c=input('Digite o valor de c = ');
     x=-5:0.1:5;
     y=a*x.^2+b*x+c;
     figure(1)
     plot(x,y)
      grid on
     clc
     disp(' ');
     aux=input(' Deseja ver outra curva ? (s/n) ==> ','s');
  endwhile
disp('Programa finalizado.');
close;
```

6. CONTROLE DE FLUXO

Os comandos que controlam o fluxo especificam a ordem em que as operações são realizadas. No *GNU Octave* estes comandos são semelhantes aos usados na linguagem C bem como em outras linguagens.

6.1. Laço *for*

O laço for é o controlador de fluxo mais simples e usado na programação *GNU Octave*. Exemplo de uso do comando for:

```
>> for i = 1:n;
>>    for j = 1:n;
>>       ...
>>    endfor
>> endfor
```


É muito comum construções em que conjuntos de laços for são usados, principalmente em operações envolvendo vetores e matrizes:

```
for i=1:10;
  for j=1:10;
     A(i,j)=i+j;
     B(i,j)=i-j;
  endfor
endfor
A
B
C=A+B
```


6.2. Comando *while*

No exemplo abaixo o laço while é executado se a condição testada for verdadeira.

```
a = 1; b = 15;
while a<b
   a = a+1
   b = b-1
endwhile
disp('fim do loop')
```

- Inicialmente são atribuídos valores para **a** e **b**.
- A condição **a<b** é testada.
- Se ela for verdadeira o corpo do laço será executado e o procedimento é repetido.
- Quando o teste se tornar falso o laço terminará, e a execução continuará no comando que segue o laço, após o endwhile.

Operadores relacionais:

<	Menor do que
>	Maior do que
<=	Menor ou igual a
>=	Maior ou igual a
==	Igual
~=	Diferente

Operadores lógicos

&	e
|	ou
~	não

6.3. Declarações if, else

A seguir, é apresentado um exemplo do uso da declaração if no *GNU Octave*.

```
for i = 1:5;
   for j = 1:5;
      if i == j
         A(i,j) = 2;
      else
         if abs(i-j) == 1
            A(i,j) = 1;
         else
            A(i,j) = 0;
         endif
      endif
   endfor
endfor
A
```


7. ABERTURA E VISUALIZAÇÃO DE IMAGENS

São vários os comandos destinados à visualização e processamento de imagens. Boa parte destes comandos é incorporada em bibliotecas (ou pacotes) específicas como a denominada "image", disponível na página http://octave.sourceforge.net/.

O propósito central desta seção é apresentar apenas alguns comandos básicos, em função das diversas possibilidades de processamento. Dada a diversidade de comandos um, ou mais cursos poderiam ser ministrados com este propósito.

Pode-se notar que boa parte dos comandos são similares ao da biblioteca MATLAB. Para detalhes adicionais sobres os diversos comandos sugere-se MºANDREW (2004) e EATON et al (2015).

A tabela seguinte mostra alguns comandos básicos, que permitem a operação com imagens no *GNU Octave*:

imread	Faz a leitura de imagens
image	Permite a visualização de uma matriz como uma imagem
hist	Calcula o histograma de um conjunto de dados
rgb2gray	Converte imagem colorida (RGB) para tons de cinza
axis	Controla a aparência e a escala dos eixos
disp	Visualização de um vetor no modo texto
figure	Abre uma nova janela gráfica
colormap	Permite definir um mapa de cores Exemplo: colormap(gray(256))
imfinfo	Função que retorna uma estrutura que contem diversas informações sobre determinada imagem Exemplo de uso: imfinfo("arquivo_imagem")

Abertura de imagem / Visualização / Conversão RBG para Tom de Cinza

Crie um arquivo (.m), digite, salve e execute os comandos abaixo:

```
warning('off');
pkg load image;
nome='lenna.jpg';
DATA=imread(nome);

figure(1);
image(DATA);
xlabel([ 'Imagem: ' nome ]);
disp([ 'Imagem entrada: ' nome ]);

figure(2);
DATAg=rgb2gray(DATA);
image(DATAg);
colormap(gray(256));
xlabel([ 'Imagem (tom de cinza) ' ]);
disp([ 'Imagem (tom de cinza) : ok' ]);
```

Abertura de imagem / Visualização / Conversão RBG para Tom de Cinza (Cont.)

No mesmo arquivo criado antes, escreva um código que permita criar a imagem negativa em tom de cinza. Mostre esta imagem:

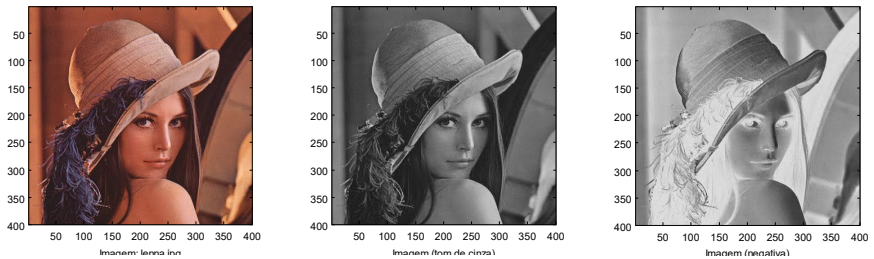
```
...

figure(3);
```


Abertura de imagem / Visualização / Conversão RBG para Tom de Cinza (Cont.)

No mesmo arquivo criado antes, escreve um código que permita criar a imagem negativa em tom de cinza. Mostre esta imagem:

```
...

figure(3);
DATAn=255.-DATAg;
image(DATAn);
colormap(gray(256));
xlabel([ 'Imagem (negativa)' ]);
disp([ 'Imagem (negativa)     : ok' ]);
```

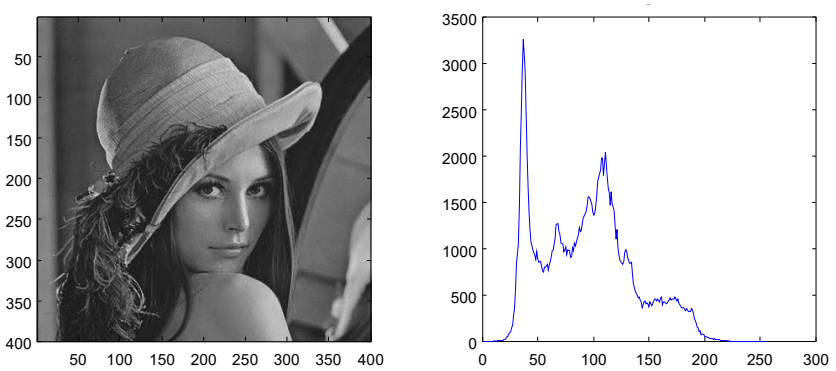

Resultado do processamento anterior, onde é mostrada a imagem original (colorida), a convertida para tons de cinza e a negativa.



Leitura / Cálculo do Histograma de Frequência / Visualização do Histograma

```
warning('off');
pkg load image;
nome='lenna.jpg';
DATA=imread(nome);
DATAg=rgb2gray(DATA);

figure(1);
image(DATAg);
colormap(gray(256));
xlabel([ 'Imagem: ' nome ]);
disp([ 'Imagem (original): ' nome ]);

[freq,tom]=hist(DATAg(:),0:255);
figure(2);
plot(tom,freq);
title([ 'Histograma' ]);
```

Resultado do processamento anterior, onde é mostrada a imagem lida e respectivo histograma.



Alguns comandos adicionais relacionado ao processamento de imagens:

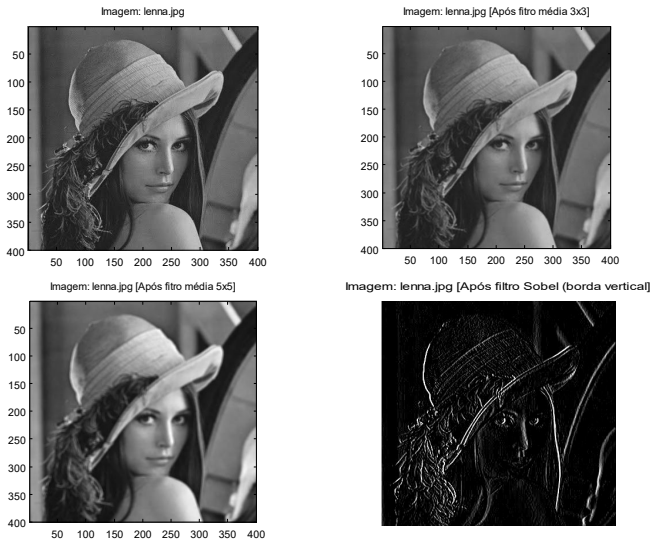
imshow	Permite a visualização de uma imagem
imagesc	Permite a visualização de uma imagem após a aplicação de uma escala, de modo que todo o mapa de cores seja utilizado
imwrite	Permite salvar uma imagem em arquivo (imwrite(imagem, "nome.ext")) Formatos aceitos: jpg, tif, gif, pgm, bmp, png, etc
filter2(B,X)	Realiza a correlação do dado em X usando o filtro 2D armazenado em B
conv2 (A,B)	Realiza a convolução da matriz A sobre a matriz B
uint8	Converte os elementos para inteiros sem sinal (8 bits)
int8	Converte os elementos para inteiros de 8 bits

Leitura / Aplicação de um filtro a uma imagem / Visualização

```
pkg load image;
nome='lenna.jpg';
DATA=imread(nome);
DATA=rgb2gray(DATA);

figure(1);
image(DATA);
colormap(gray(256));
title([ 'Imagem: ' nome ]);

f33=(1/9).*[ 1  1  1;
             1  1  1;
             1  1  1 ];
PARCIALa=filter2(f33,DATA,'same');
figure(2);
image(PARCIALa);
colormap(gray(256))
title([ 'Imagem: ' nome ' [Após fitro media 3x3]' ]);
```


Leitura / Aplicação de um filtro a uma imagem / Visualização (Continuação)

```
...

f55=(1/25).*[ 1  1  1  1  1;       1  1  1  1  1;
              1  1  1  1  1;       1  1  1  1  1;
              1  1  1  1  1 ];
PARCIALb=conv2(DATA,f55,'same');
figure(3);
imagesc(PARCIALb);
colormap(gray(256));
title([ 'Imagem: ' nome ' [Após fitro média 5x5]' ]);

f2da=[   -1   0   1;  -2   0   2;  -1   0   1 ];
PARCIAL=filter2(f2da,DATA);
figure(4);
imshow(uint8(PARCIAL));
title([ 'Imagem: ' nome ' [Apos filtro Sobel (borda
vertical]' ]);
```



Abertura de imagem / Armazenamento em outros formatos

No mesmo arquivo *script* anteriormente escrito, incorpore no código o armazenamento das imagens após aplicação dos filtros de suavização 3x3 e 5x5, em arquivos com diferentes formatos:

. . .

Abertura de imagem / Armazenamento em outros formatos

No mesmo arquivo *script* anteriormente escrito, incorpore no código o armazenamento das imagens após aplicação dos filtros de suavização 3x3 e 5x5, em arquivos com diferentes formatos:

. . .

```
imwrite(uint8(PARCIALa),"filtro3x3.jpg");
imwrite(uint8(PARCIALb),"filtro5x5.png");
disp(['Imagens filtro3x3.jpg e filtro5x5.png salvas.']);
```


8. IMPORTAÇÃO / EXPORTAÇÃO DE DADOS

Neste tópico são apresentados alguns comandos adicionais relacionados à manipulação de arquivos e ao uso de funções criadas pelo usuário.

8.1. Importação e Exportação de Dados

Os dados (matrizes, vetores, escalares, cadeia de caracteres, etc) disponíveis no Ambiente de Trabalho do *GNU Octave*, como foi visto, podem ser armazenados em arquivos no modo texto, binário (e outros formatos), utilizando o comando save.

Existem diversas maneiras de utilizar este comando, como pode-se ver nos exemplos mostrados na sequência:

save arql.sai X Y Z –binary	salva as variáveis X, Y e Z no arquivo binário "arq1.sai"
save arq2.sai X Y Z -ascii	salva as variáveis X., Y e Z no arquivo texto "arq2.sai" com 8 casas decimais
save arq3.sai X Y Z -ascii -double	salva as matrizes X., Y e Z no arquivo texto "arq3.sai" com 16 casas decimais

Os dados obtidos por outros programas podem ser importados pelo *GNU Octave*, desde que estes dados sejam gravados em formato apropriado. Se os dados são armazenados no formato ASCII, e no caso de matrizes, com colunas separadas por espaços e cada linha da matriz em uma linha do texto, o comando load pode ser usado.

Por exemplo suponha que um programa qualquer criou arquivo "polig1.dat" que contém a seguinte matriz.

```
1 1
3 1
3 3
1 3
1 1
```

Ao executar o comando:

```
>> load polig1.dat
```

o *GNU Octave* importa a matriz, que é armazenada com o nome polig1.


```
» polig1
polig1 =
     1     1
     3     1
     3     3
     1     3
     1     1
```


O *GNU Octave* pode também importar (através do comando load) os dados que foram anteriormente exportados por ele. Por exemplo, para importar as variáveis X, Y e Z, anteriormente exportadas usando o comando save, pode-se fazer:

load arq1	save arq1 X Y Z
load arq2.sai	save arq2.sai X Y Z -ascii
load arq3.sai	save arq3.sai X Y Z -ascii -double

Deve-se ressaltar que o comando save, quando usado para exportar os dados do *GNU Octave* em formato texto, exporta apenas um bloco contendo todas as variáveis. Quando os dados são importados através do comando load, apenas uma variável com nome do arquivo é criada.

Exemplo:

```
>> x = rand(3,3)
x =
   0.2190 0.6793 0.5194
   0.0470 0.9347 0.8310
   0.6789 0.3835 0.0346

>> y = rand(3,3)
y =
   0.0535  0.0077 0.4175
   0.5297  0.3835 0.6868
   0.6711  0.0668 0.5890
```



```
>> save arq2.sai X Y -ascii
>> clear
>> whos
>> load arq2.sai
>> arq2
   arq2 =

   0.2190      0.6793   0.5194
   0.0470      0.9347   0.8310
   0.6789      0.3835   0.0346
   0.0535      0.0077   0.4175
   0.5297      0.3834   0.6868
   0.6711      0.0668   0.5890
```


9. CRIAÇÃO DE FUNÇÕES

9.1. Criação de funções

O uso de funções é muito útil tanto na execução de algumas tarefas repetitivas quanto no aproveitamento do código em diferentes aplicações. Pode-se, deste modo, fazer a criação bibliotecas de funções destinadas a solução de alguns problemas específicos.

A estrutura geral de uma função do *GNU Octave* é a seguinte:

```
function [ retorno ] = teste (variáveis)
...
endfunction
```


Na sequência são mostrados dois exemplos de funções. Uma delas foi criada com o objetivo de, dado um ângulo (alfa) em radianos, montar a matriz de rotação M:

$$M(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix}.$$

A segunda função tem o propósito de, dada uma matriz de rotação M e um ponto de coordenas (x,y), aplicar a matriz de rotação de modo que se tenha:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M(\alpha) \begin{bmatrix} x \\ y \end{bmatrix}$$

Função "f_matriz_rotacao" armazenada no arquivo f_matriz_rotacao.m

```
function Mrot=f_matriz_rotacao(alfa)

# F_MATRIZ_ROTACAO Calcula a matriz de rotação no plano
#                  Entrada: Ângulo em graus
#                  Retorno: Matriz de rotação 2x2
#
# Autores: Mauricio Galo e Paulo Camargo
# UNESP, 2015

   alfa=alfa*pi/180;
   Mrot=[  cos(alfa)  sin(alfa);
          -sin(alfa)  cos(alfa)];

endfunction
```


Função "f_aplica_rotacao" armazenada no arquivo f_aplica_rotacao.m

```
function [xrot,yrot]=f_aplica_rotacao(M,x,y)

# F_APLICA_ROTACAO Faz a rotação de um ponto (x,y) usando
#                  a matriz de rotação M
#                  Retorno: Coordenadas rotacionadas do
#                           ponto (x,y)
#
# Autores: Mauricio Galo e Paulo Camargo
# UNESP, 2015

   xrot=x*M(1,1) + y*M(1,2);
   yrot=x*M(2,1) + y*M(2,2);

endfunction
```


Como exemplo de aplicação das funções criadas para o *GNU Octave*, inicialmente escreva um *script* que permite ler nas variáveis polig1.dat e polig2.dat, bem como permita a visualização destes vetores, usando o comando plot:

Conteúdo do arquivo *polig1.dat*	Conteúdo do arquivo *polig2.dat*
0 0	*2 2*
1 0	*3 3*
1 1	*2 3*
0 1	*2 2*
0 0	

Leitura dos dados / Visualização dos polígonos

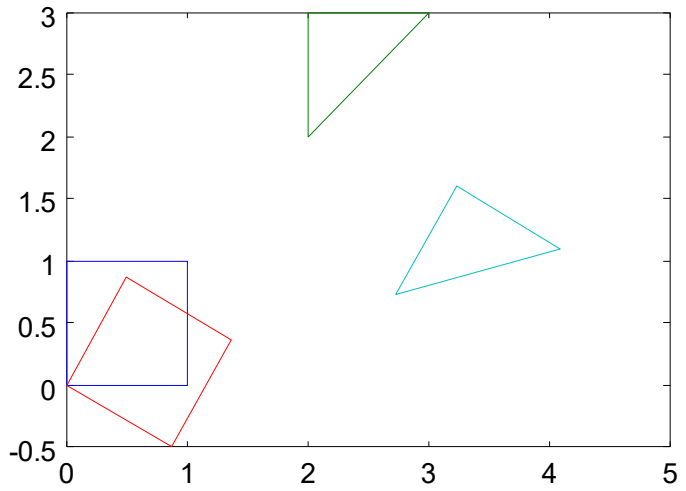
```
load 'polig1.dat'
load 'polig2.dat'

[lin1,col1]=size(polig1);
[lin2,col2]=size(polig2);

x1=polig1(1:lin1,1);
y1=polig1(1:lin1,2);
x2=polig2(1:lin2,1);
y2=polig2(1:lin2,2);

figure(1);
plot(x1,y1,x2,y2);
```

Cálculo da Matriz de Rotação / Rotação / Visualização dos polígonos após rotação

No mesmo *script* anterior, calcule e matriz de rotação, aplique-a a todos os pontos dos dois polígonos e faça a visualização.

```
...
ROT = f_matriz_rotacao(30);

[x1r,y1r]=f_aplica_rotacao(ROT,x1,y1);
[x2r,y2r]=f_aplica_rotacao(ROT,x2,y2);

figure(2);
plot(x1r,y1r,x2r,y2r);

figure(3);
plot(x1,y1,x2,y2,x1r,y1r,x2r,y2r);
```


Visualização dos polígonos antes e após rotação



REFERÊNCIAS

CORAL, A. M.; SANTOS, M. P.; BASTOS, T. D. A.; BORBA, M. *Curso de Matlab*. Universidade Federal de Santa Catarina, Dep. de Eng. de Produção, Programa Especial de Treinamento – PET. Florianópolis – SC, 1999. 29p.

EATON, J. W.; BATEMAN, D.; HAUBERG, S. *GNU Octave – Edition 3 for Octave version 3.2.3*, July, 2007. 672p.

EATON, J. W.; BATEMAN, D.; HAUBERG, S.; WEHBRING, R. *GNU Octave* – Edition 4 for Octave version 4.0.0 – Free Your Numbers, March, 2015. 966p.

HANSELMAN, D.; LITTLEFIELD, B. *Matlab - Versão do Estudante: guia do usuário* (Tradução). São Paulo: Makron Books, 1997. 305p.

M^{c}ANDREW, A. *Introduction to Digital Image Processing with MATLAB*®. Thomson Course Technology, 2004. 509p. ISBN: 0-534-40011-6

QUARTERONI, A.; SALERI, F.; GERVASIO, P. *Scientific Computing with MATLAB and Octave*. Fourth Edition. Text in Computational Science and Engineering. BARTH, T. J.; GRIEBEL, M.; KEYES, D. E.; NIEMINEN, R. M.; ROOSE, D.; SCHLICK. T. (Eds.). Lausanne: Springer, 2014.

PAGAMISSE, A.; SOUZA, L. H. G. *Introdução ao Software Octave*. Semana de Cursos de Matemática, Estatística e Computação, 25-29 de agosto de 2003, FCT/UNESP, Presidente Prudente, 2003. 54p.

SIGMON, K. *Matlab Primer* – Third Edition. Department of Mathematics, University of Florida, Gainesville, 1993. 35p. Disponível em http://www.math.toronto.edu/mpugh/primer.pdf. Acesso em Março/2021.

SIQUEIRA, A. F. *Octave – Seus primeiros passos na programação científica*. São Paulo: Casa do Código, 2015. 202p.

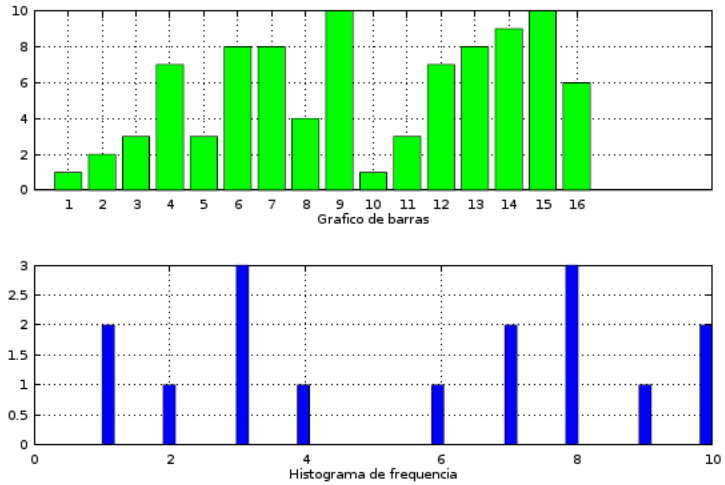
ZERI, L. M. M. *Apostila de Matlab*. Instituto Nacional de Pesquisas Espaciais – INPE, 2001. 19p.

ANEXOS

Nesta seção serão apresentados alguns exemplos adicionais de comandos e respectivos resultados, que podem ser úteis em algumas aplicações na visualização de dados em algumas áreas das ciências exatas e engenharias, usando o aplicativo *GNU Octave*.

Anexo I

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): subplot, bar, hist.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear
y = [1 2 3 7 3 8 8 4 10 1 3 7 8 9 10 6];
figure;
subplot(2,1,1);
bar(y,'g');
xlabel('Grafico de barras');
grid
subplot(2,1,2);
hist(y,50,'b')
xlabel('Histograma de frequencia');
grid
```

Resultado gráfico do *script* anterior.

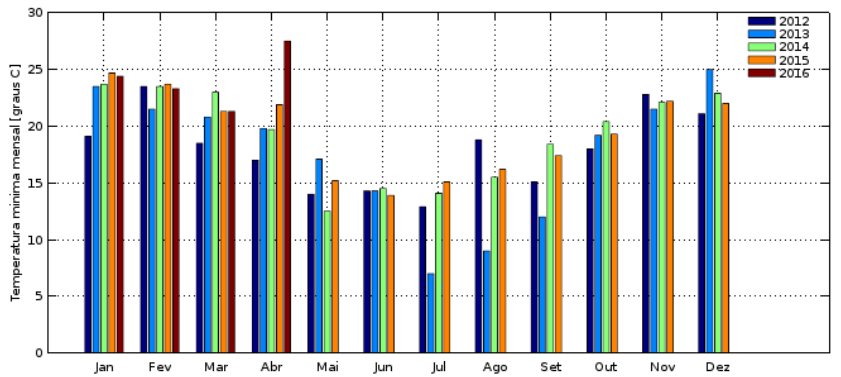
Anexo II

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): bar, legend.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear
figure;
periodo = 2012:1:2016;
dados = [ 19.1  23.5  23.7  24.7  24.4;
          23.5  21.5  23.5  23.7  23.3;
          18.5  20.8  23.0  21.3  21.3;
          17.0  19.8  19.7  21.9  27.5;
          14.0  17.1  12.5  15.2  0;
          14.3  14.3  14.5  13.9  0;
```

(continua)

```
          12.9   7.0  14.1  15.1  0;
          18.8   9.0  15.5  16.2  0;
          15.1  12.0  18.4  17.4  0;
          18.0  19.2  20.4  19.3  0;
          22.8  21.5  22.1  22.2  0;
          21.1  25.0  22.9  22.0  0  ];
B=bar(dados);
legend(B,'2012','2013','2014','2015','2016');
legend('Boxoff');
meses=({'Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set
','Out','Nov','Dez'});
set(gca,'XTickLabel',meses);
ylabel('Temperatura minima mensal [graus C]');
```

Obs.: Fonte dos dados: www.ciiagro.sp.gov.br. Os dados referentes ao mês de abril/2016 correspondem à coleta de dados dos primeiros 15 dias de abril, uma vez que os dados foram obtidos neste site em 16/Abril/2016.



Resultado gráfico do *script* anterior.

Fonte dos dados: CIIAGRO – Centro Integrado de Informações Agrometeorológicas (site: www.ciiagro.sp.gov.br)

Anexo III

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): pie.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

PAIS = {'CHINA 49%' 'JAPAO 12%' 'ALEMANHA 6,7%' 'COREIA DO
SUL 3,8%' 'FRANCA 2,6%' 'HOLANDA 2,2%' 'ITALIA 2,6%' 'OUTROS
21,1%'};
DATA = [ 49 12 6.7 3.8 2.6 2.2 2.6 21.1 ];
figure;
pie(DATA,PAIS);
S=['PRINCIPAIS COMPRADORES DE MINERIO DE FERRO BRASILEIRO EM
2010. Fonte: IBRAM'];
title(S,'fontsize',11);
```

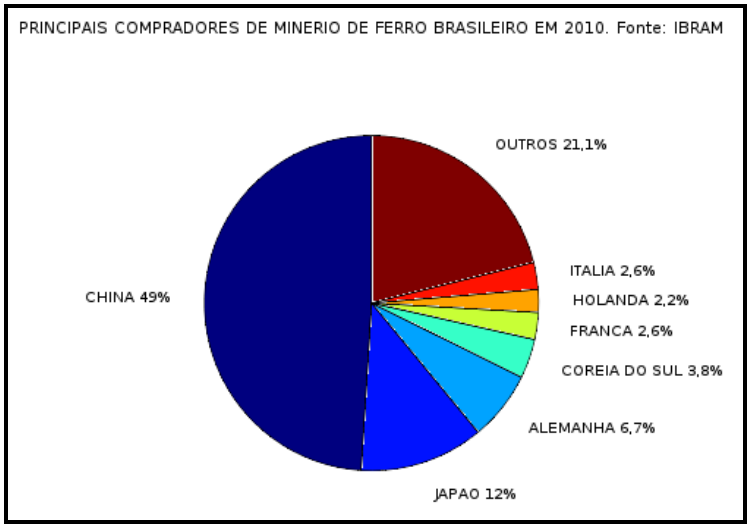



Gráfico resultante do *script* anterior.

Anexo IV

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): plot, legend, uso de letras gregas.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

alpha = 2;
beta = 2.4;
Delta = pi;
t=0:0.1:2*pi;
plot(t,alpha*sin(t),t,beta*sin(t/2),t,Delta*sin(t/2.5));
xlabel('Angulo t [rad], \alpha = 2, \beta = 2.4, \Delta =
\pi');
legend({'\alpha*sin(t)','\beta*sin(t/2)','\Delta*sin(t/2.5)'
},'location','southwest');
grid
```

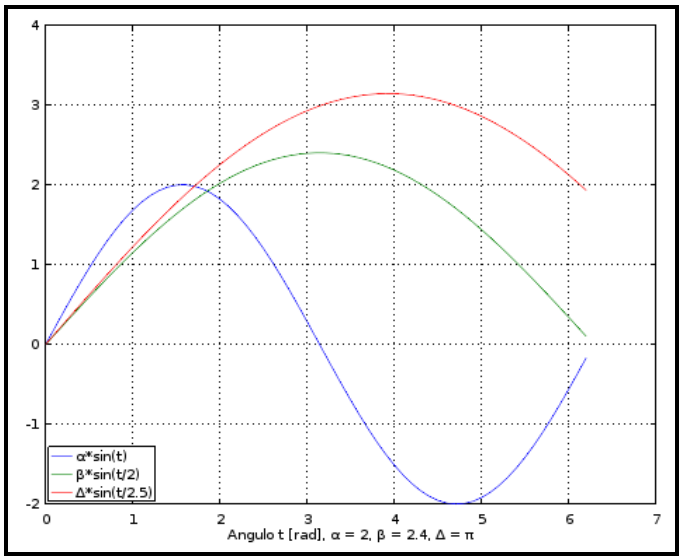



Gráfico resultante do *script* anterior.

Anexo V

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): num2str, linewidth.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

lambda=input('Entre com o fator de escala: ');
figure(1)
t=0:0.2:6*pi;
y=(1-t).*sin(t);
y1=lambda*(1-t).*sin(t);
plot(t,y,'linewidth',2,t,y1,'linewidth',2);
ylabel('f(t)');
legend('y(t)','\lambda.y(t)','location','northwest');
legend('boxoff');
```

(continua)


```
S = [ 'Angulo t [rad], Fator de escala: \lambda='
num2str(lambda) ];
xlabel(S);
```

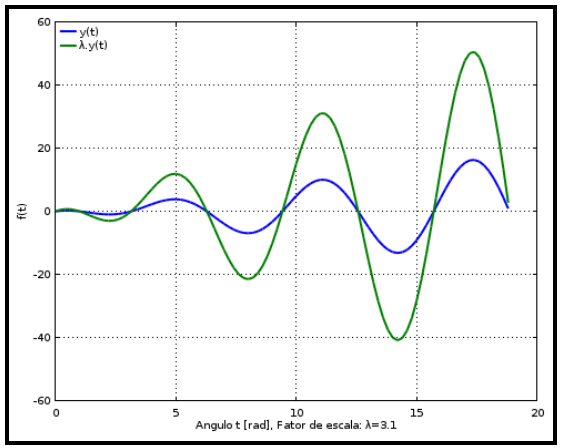


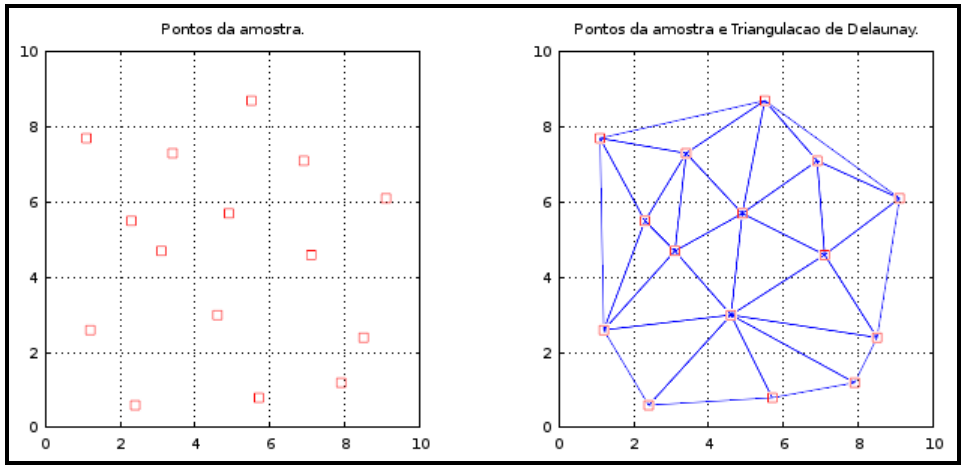
Gráfico resultante do *script* anterior.

Anexo VI

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): size, delaunay, triplot, axis.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

close all;
clear;
clc;

xy = [
6.9 4.9 1.1 1.2 7.9 5.7 2.4 4.6 3.1 3.4 5.5 9.1 7.1 8.5 2.3;
7.1  5.7  7.7  2.6  1.2  0.8  0.6  3.0  4.7  7.3  8.7  6.1  4.6  2.4
5.5];
xy=xy';
```

(continua)


```
[lin,col]=size(xy);
x=xy(1:lin,1);
y=xy(1:lin,2);
figure

subplot(1,2,1)
plot(x,y,'sr');
title('Pontos da amostra.');
axis('square');
grid;

subplot(1,2,2)
plot(x,y,'sr');
tri=delaunay(x',y');
hold on;
triplot(tri,x,y);
title('Pontos da amostra e Triangulacao de Delaunay.');
axis('square');
grid;
```

Gráficos resultantes do *script* anterior.

Anexo VII

```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): subplot, bar, hist.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear
load terreno.txt
[lin,col]=size(terreno);
x=terreno(1:lin,1);
y=terreno(1:lin,2);
z=terreno(1:lin,3);
xn = x - min(x);
yn = y - min(y);
zn = z;
```

(continua)

```
figure;
plot(xn,yn,'.r');
xlabel('X(m)');
ylabel('Y(m)');

figure;
plot3(xn,yn,zn,'.r');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');

limX = min(xn):1.:max(xn);
limY = min(yn):1.:max(yn);
[xf,yf] = meshgrid(limX, limY);
zf = griddata(xn, yn, zn, xf, yf);

figure;
plot(xn,yn,'.r');
hold on
```

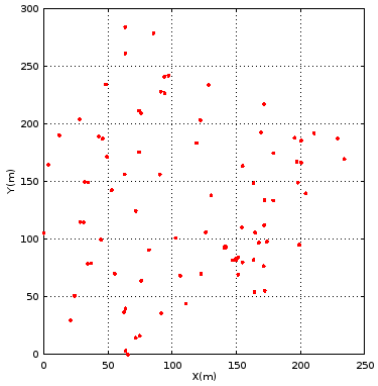
(continua)

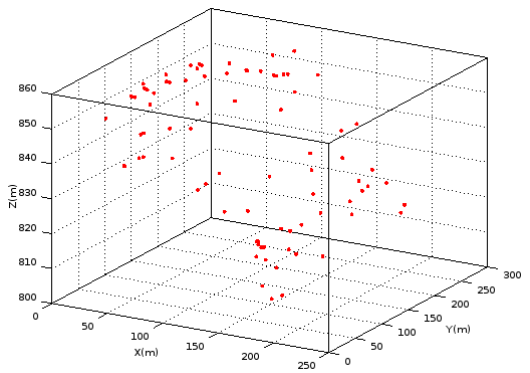

```
contour(zf,50,'-b');

xlabel('X(m)');
ylabel('Y(m)');

figure;
surfc(xf, yf, zf);
title('Superficie topografica');
xlabel('X(m)');
ylabel('Y(m)');
zlabel('Z(m)');
```

Na sequência são mostrados o arquivo *terreno.txt*, usado no *script* anterior, bem como os quatro gráficos resultantes da execução deste *script*.

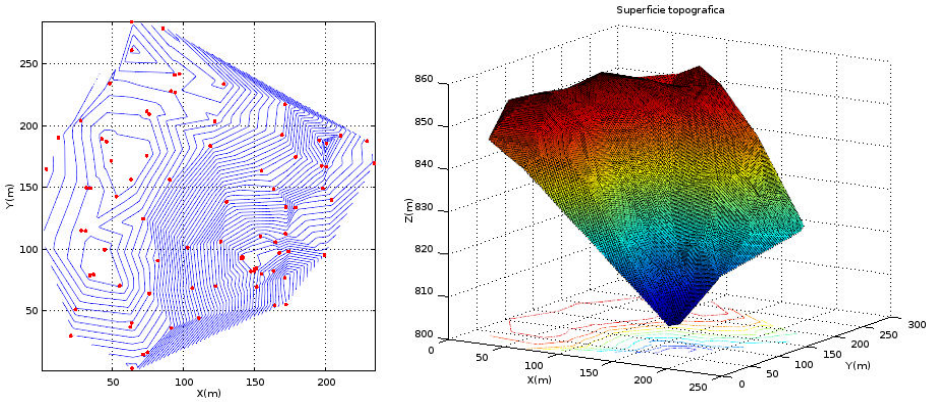
Conteúdo do arquivo *terreno.txt*.

10605.777 14253.405 847.374	10654.248 14300.966 832.379	10645.834 14366.175 850.873
10599.733 14226.811 849.188	10677.961 14273.010 821.125	10569.665 14349.903 855.487
10615.293 14198.717 843.856	10695.797 14296.792 830.147	10566.565 14352.056 855.569
10634.561 14206.868 835.203	10687.212 14311.491 835.304	10551.639 14366.817 853.152
10630.045 14231.201 834.698	10702.598 14296.371 825.103	10598.039 14338.327 855.792
10626.659 14263.897 834.841	10727.848 14302.539 825.620	10586.884 14319.117 855.908
10646.331 14232.702 827.488	10722.865 14257.941 816.945	10573.056 14334.192 856.122
10650.098 14268.751 824.985	10757.893 14332.396 825.310	10555.576 14312.625 853.111
10665.250 14256.447 818.154	10724.310 14329.433 829.970	10558.450 14312.015 853.120
10664.420 14255.853 818.118	10720.952 14330.090 832.536	10535.919 14352.970 850.769
10664.845 14255.081 818.081	10702.665 14337.463 837.446	10527.406 14327.464 849.723
10665.840 14255.707 818.115	10721.728 14311.994 828.226	10555.024 14277.472 853.647
10673.666 14244.882 817.800	10752.754 14350.186 825.874	10552.173 14277.784 853.858
10673.593 14246.394 817.840	10719.296 14350.783 829.237	10557.990 14241.363 854.902
10675.106 14247.072 817.834	10724.365 14348.249 834.469	10560.678 14241.972 854.754
10671.091 14244.626 818.551	10693.065 14355.301 843.320	10568.406 14262.295 854.681
10678.698 14242.686 814.706	10734.297 14354.692 830.744	10576.697 14305.307 855.266
10688.459 14268.374 821.455	10695.266 14380.014 843.335	10523.849 14268.045 851.384
10691.294 14259.813 816.981	10617.750 14403.847 851.262	10587.444 14202.788 849.155
10695.599 14275.018 823.062	10621.073 14404.658 851.418	10586.427 14199.660 848.979
10697.511 14260.744 815.977	10609.384 14441.500 854.433	10579.011 14232.853 854.422
10687.189 14244.902 812.722	10587.327 14446.788 851.514	10547.881 14213.814 850.228
10695.143 14239.535 805.741	10598.051 14374.378 853.819	10595.544 14177.230 844.544
10675.229 14232.082 816.265	10599.717 14372.089 853.823	10598.582 14179.058 844.836
10687.963 14217.071 809.833	10618.061 14389.550 852.020	10544.620 14192.465 846.335
10595.478 14287.307 851.135	10615.075 14390.768 851.944	10589.274 14162.826 842.976
10614.218 14318.957 850.357	10587.427 14423.912 848.785	10587.583 14166.114 842.864
10642.839 14346.278 847.213	10572.092 14397.026 852.387	10695.918 14217.893 806.531
10678.690 14326.350 838.391	10652.161 14396.553 853.625	
(continua)	(continua)	# Fim do arquivo





Gráficos 1 e 2 resultantes do *script* anterior.



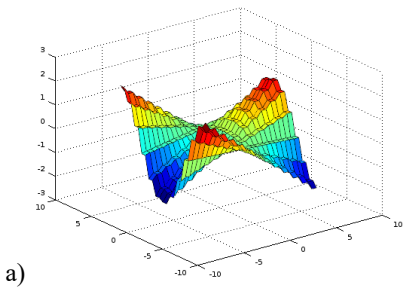
Gráficos 3 e 4 resultantes do *script* anterior.

Anexo VIII

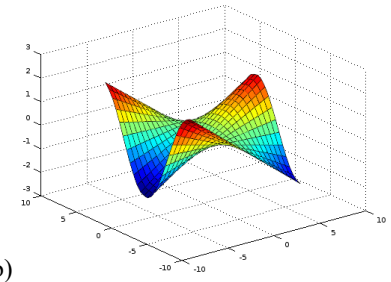
```
%{
  Exemplo de uso de alguns comandos do GNU Octave
  Comandos (principais): meshgrid, surf, interp2.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear
Lmin = -2*pi;
Lmax =  2*pi;
[x,y] = meshgrid(Lmin:Lmax);
z = (1/3)*x.*cos(y/2);

figure;
surf(x,y,z);
```

(continua)

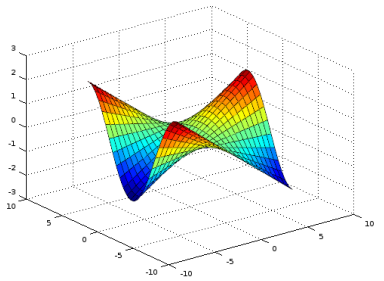

```
[xi,yi] = meshgrid(Lmin:0.5:Lmax);
zi = interp2(x, y, z, xi, yi);
figure;
surf(xi,yi,zi);

zi1 = interp2(x, y, z, xi, yi,'nearest');
figure;
surf(xi,yi,zi1);

zi2 = interp2(x, y, z, xi, yi,'cubic');
figure;
surf(xi,yi,zi2);

zi3 = interp2(x, y, z, xi, yi,'spline');
figure;
surf(xi,yi,zi3);
%
% Fim do script
%
```

a)



b)



c)

Superfícies 3 a 5 resultantes do *script* anterior, onde os interpoladores “nearest” (a), “cubic” (b) e “spline” (c) foram utilizados.

Anexo IX

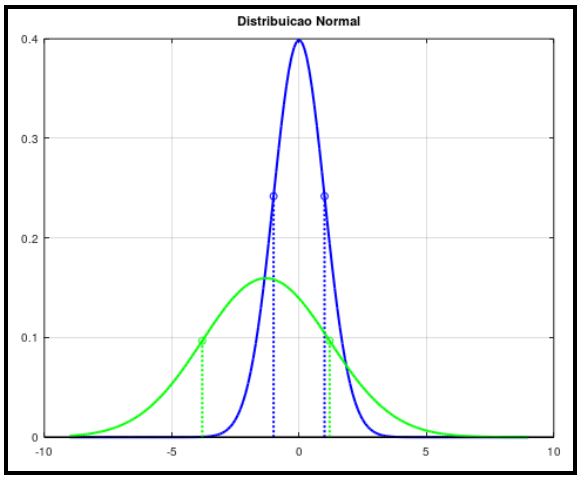
```
%{
  Visualização da curva normal
  Comando (principal): normpdf

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear all;
clc
pkg load statistics

disp(' ')
disp(' ########################################## ');
disp('  Visualizacao da curva normal no GNU Octave  ');
disp(' ########################################## ');
disp(' ')
```

(continua)


```
% Média e desvio da amostra 1
am1_med = 0;
am1_des = 1;

% Média e desvio da amostra 2
am2_med = -1.3;
am2_des = 2.5;

% Posição correspondente ao desvio-padrao dado
L1 = [ am1_med-am1_des; am1_med+am1_des ];
L2 = [ am2_med-am2_des; am2_med+am2_des ];

figure(1);
x = -9:0.1:9;

% Visualização da curva 1
```

(continua)


```
plot(x,normpdf(x,am1_med,am1_des),'-b','linewidth',2);
hold on
stem(L1,normpdf(L1,am1_med,am1_des),':b','linewidth',2);
hold on

% Visualização da curva 2

plot(x,normpdf(x,am2_med,am2_des),'-g','linewidth',2);
hold on
stem(L2,normpdf(L2,am2_med,am2_des),':g','linewidth',2);
grid
title('Distribuicao Normal');
hold off

%
% Fim do script "Curvas_normais.m"
%
```

Resultado do *script* mostrando as duas curvas normais.

Anexo X

```
%{
  Exemplo de uso de comandos do GNU Octave,
  que permitem o cálculo de distribuições
  estatísticas (Normal, t e Qui-quadrado)
  Comandos: norminv, tinv, chi2pdf.

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2019
%}

clear all; clc;
pkg load statistics

% NS - Nivel de Significância (%)
% GL - Graus de Liberdade
NS = 0.1;
GL = 25;
```

(continua)


```
% Distribuição Normal Padronizada (z)
dist_z = norminv(1-NS);

% Distribuição t
dist_t = tinv(1-NS, GL);

% Distribuição Qui-Quadrado
dist_QQ = chi2inv(1-NS, GL);

disp(' ')
disp(' ############################################ ');
disp('          Exemplo de uso de algumas*          ');
disp('      funcoes estatisticas no GNU Octave      ');
disp(' ############################################ ');
disp(' ')
S=sprintf(' Nivel de significancia [%%]: %5.3f',NS);
disp(S)
S=sprintf(' Graus de liberdade        : %d',GL);
disp(S)
```

(continua)


```
S=sprintf(' z teorico                 : %8.3f',dist_z);
disp(S)
S=sprintf(' t teorico                 : %8.3f',dist_t);
disp(S)
S=sprintf(' QQ teorico                : %8.3f',dist_QQ);
disp(S)
disp(' ')
disp(' * Compare os valores determinados com os');
disp('   obtidos por uma tabela estatistica.  ');
disp(' ')

%
% Fim do script 'Distribuicoes_exemplo.m'
%
```


Resultado do *script* anterior.

```
############################################
         Exemplo de uso de algumas*
     funcoes estatisticas no GNU Octave
############################################

Nivel de significancia [%]: 0.100
Graus de liberdade        : 25
z teorico                 :    1.282
t teorico                 :    1.316
QQ teorico                :   34.382

* Compare os valores determinados com os
  obtidos por uma tabela estatistica.
```

Anexo XI

```
%{
  Exemplo de uso de comandos do GNU Octave
  Comando (principal): fill

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2020
%}

clear all;
close all;
clc;

load poligonoAt1.dat
load poligonoAt2.dat

[linhas1,colunas]=size(poligonoAt1);
```

(continua)


```
for i=1:linhas1
   x1(i)=poligonoAt1(i,1);
   y1(i)=poligonoAt1(i,2);
endfor

[linhas2,colunas]=size(poligonoAt2);
for i=1:linhas2
   x2(i)=poligonoAt2(i,1);
   y2(i)=poligonoAt2(i,2);
endfor

figure(1)
plot(x1,y1,'r',x2,y2,'g');
axis('square');
title('Polígonos - Sem preenchimento');

figure;
h=fill(x1,y1,'r',x2,y2,'g');
```

(continua)


```
axis('square');
title('Polígonos - Com preenchimento');

%
% Fim do script 'Preenchimento.m'
%
```


Conteúdo dos arquivos *poligonoAt1.dat* e *poligonoAt2.dat*.

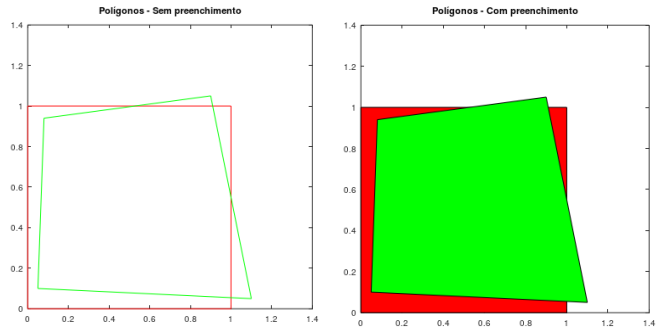
poligonoAt1.dat

0.0	0.0
1.0	0.0
1.0	1.0
0.0	1.0
0.0	0.0

poligonoAt2.dat

0.05	0.10
1.10	0.05
0.90	1.05
0.08	0.94
0.05	0.10

Resultados do *script preenchimento.m*.



Anexo XII

```
%{
  Exemplo de uso de comandos do GNU Octave
  Comandos (principais): fill, facealpha

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2020
%}

clear all;
close all;
clc;

load poligonoAt1.dat
load poligonoAt2.dat

[linhas1,colunas]=size(poligonoAt1);
```

(continua)

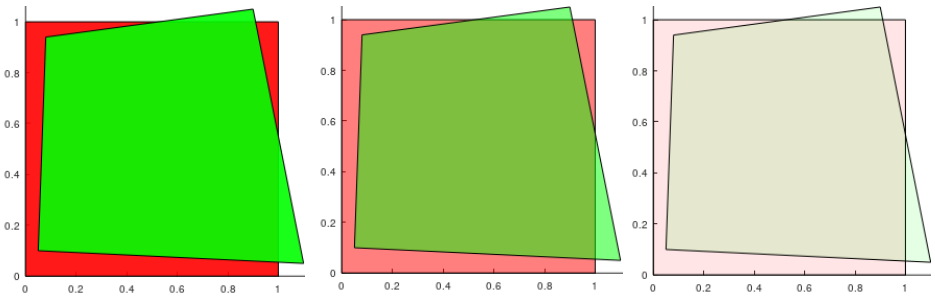

```
for i=1:linhas1
   x1(i)=poligonoAt1(i,1);
   y1(i)=poligonoAt1(i,2);
endfor

[linhas2,colunas]=size(poligonoAt2);
for i=1:linhas2
   x2(i)=poligonoAt2(i,1);
   y2(i)=poligonoAt2(i,2);
endfor

figure(1);
h=fill(x1,y1,'r',x2,y2,'g');
set(h,'facealpha',0.9);
axis('square');
title('Polígonos (Opacidade=0.9)');

figure;
```

(continua)


```
h=fill(x1,y1,'r',x2,y2,'g');
set(h,'facealpha',0.5);
axis('square');
title('Polígonos (Opacidade=0.5)');

figure;
h=fill(x1,y1,'r',x2,y2,'g');
set(h,'facealpha',0.1);
axis('square');
title('Polígonos (Opacidade=0.1)');

%
% Fim do script 'transparencia.m'
%
```


Resultados do *script transparencia.m* para diferentes valores do parâmetro `facealpha`.

`facealpha = 0` (Transparência máxima / Opacidade mínima)
`facealpha = 1` (Transparência mínima / Opacidade máxima)



Resultados para facealpha=0.9, facealpha= 0.5 e facealpha = 0.1, respectivamente.

Anexo XIII

```
%{
  Exemplo de uso de comandos do GNU Octave
  Comandos: quiver

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2020
%}

clear all;
close all;
clc;

% Leitura dos dados em arquivo
load residuos_calibracao.dat

% Fator de escala a ser aplicado nos vetores dos resíduos
esc = 1;
```

(continua)


```
% Determinação da dimensão do arquivo
[pontos,colunas]=size(residuos_calibracao);

% Colunas 1 e 2 - Coordenadas x e y
% Colunas 3 e 4 - Distorção em x e y, respectivamente
for i=1:pontos
   x(i)  = residuos_calibracao(i,1);
   y(i)  = residuos_calibracao(i,2);
   dx(i) = residuos_calibracao(i,3);
   dy(i) = residuos_calibracao(i,4);
end

figure(1);
plot(x,y,'+b');
title('Distribuição dos pontos');
axis([ -17 17 -13 13 ]);
axis('equal');
```

(continua)


```
figure;
quiver(x,y,dx,dy,esc,'b');
title('Vetores de distorção');
axis([ -17 17 -13 13 ]);
axis('equal');

%
% Fim do script 'Vetores_no_Plano.m'
%
```

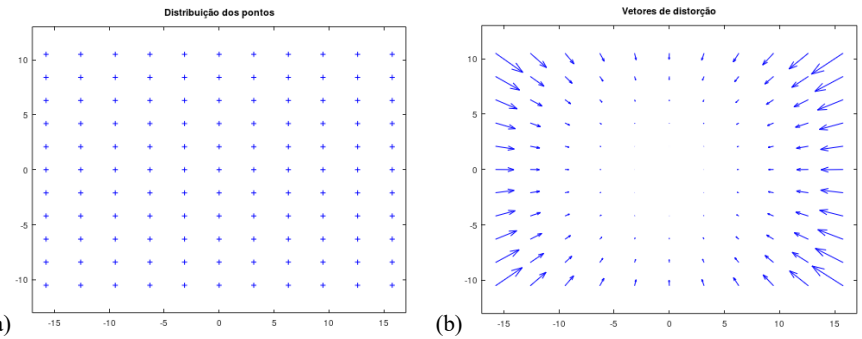
Na sequência é mostrado trecho do arquivo de dados usado no *script* *Vetores_no_Plano.m*.

Trecho do arquivo *residuos_calibracao.dat*.

```
% Coordenadas (x,y) e distorção*
% #1 #2 - Coordenadas x e y (mm)
% #3 #4 - Distorções em x e y (mm)
%
% *Distorção Radial Simétrica

 -15.750  -10.500    0.313    0.212
 -12.600  -10.500    0.185    0.158
  -9.450  -10.500    0.101    0.117
  -6.300  -10.500    0.049    0.088
  -3.150  -10.500    0.018    0.071
   0.000  -10.500   -0.003    0.067
   3.150  -10.500   -0.026    0.075
   6.300  -10.500   -0.063    0.095
   9.450  -10.500   -0.123    0.128
  12.600  -10.500   -0.220    0.174
  15.750  -10.500   -0.364    0.231
 -15.750   -8.400    0.277    0.149
 -12.600   -8.400    0.157    0.107
  -9.450   -8.400    0.080    0.074
  -6.300   -8.400    0.036    0.050
...
...
```

```
...
...
   3.150    8.400   -0.019   -0.045
   6.300    8.400   -0.049   -0.062
   9.450    8.400   -0.104   -0.089
  12.600    8.400   -0.194   -0.127
  15.750    8.400   -0.331   -0.175
 -15.750   10.500    0.320   -0.224
 -12.600   10.500    0.191   -0.168
  -9.450   10.500    0.105   -0.125
  -6.300   10.500    0.052   -0.095
  -3.150   10.500    0.019   -0.078
   0.000   10.500   -0.003   -0.074
   3.150   10.500   -0.028   -0.082
   6.300   10.500   -0.066   -0.103
   9.450   10.500   -0.128   -0.137
  12.600   10.500   -0.226   -0.184
  15.750   10.500   -0.371   -0.244

% Fim do arquivo
```

As imagens, na sequência, mostram a distribuição dos pontos (a) e os vetores das distorções para cada pontos (b), a partir dos dados lidos no arquivo *residuos_calibracao.dat*, resultantes do *script Vetores_no_Plano.m*.



(a) (b)

Distribuição dos pontos em uma imagem (a) e vetores das distorções em cada pontos (b).

Anexo XIV

```
%{
  Raios de Curvatura

  Autores: M. Galo e Paulo de O. Camargo
  Unesp, Pres. Prudente, 2021
%}

clear all; close all; clc;
Parametros_do_Elipsoide;

dFonte_Numeros = 15;
dFonte_Texto   = 18;
fTexto         = 'arial';

% Determinacao das constantes do elipsoide
a  = a/1000.;    % Conversão para [km]
b  = a*(1. - f);
```

(continua)


```
E2 = f*(2. - f);

ind=0;
for lat=0:1:90
  ind=ind+1;
  t(ind)=lat;
  faux = sqrt(1. - E2*(sin(lat*pi/180)**2));
  fN(ind) = a / faux;
  fM(ind) = a*(1. - E2) / ( faux**3 );
  fMN(ind) = a*((1. - E2)**0.5)/(faux**2);
endfor

figure(1);
h=plot(t,fN,t,fM,t,fMN);
set(h,"linewidth",1);
set(gca, 'fontsize', dFonte_Numeros);
set(gca, 'fontname', fTexto, 'xtick',0:15:90);
legend('N','M','Raio Medio');
legend('location','southeast');
```

(continua)


```
%legend('boxoff');
title('Raios de curvatura - Elipsoide', 'fontsize',
dFonte_Texto, 'fontname', fTexto);
xlabel('Latitude [graus]','fontsize', dFonte_Texto,
'fontname', fTexto);
ylabel('Raios de curvatura [km]', 'fontsize', dFonte_Texto,
'fontname', fTexto);
grid;

%
% Fim do script 'Raios_de_Curvatura.m'
%
```

Na sequência é mostrado o arquivo "`Parametros_do_Elipsoide.m`", que contém os parâmetros do elipsoide utilizado. Pode-se observar que o conteúdo desse arquivo é lido pelo *script* a partir da seguinte linha de comandos:

```
Parametros_do_Elipsoide;
```

Conteúdo do arquivo *Parametros_do_Elipsoide.m*.

```
%{
   Arquivo: Parametros_do_Elisoide.m
%}

0;

% Elipsoide SAD 69
% a: semi-eixo [m]
% f: achatamento

a = 6378160;
f = 1./298.25;

% Fim do arquivo
```


O resultado do *script* anterior é mostrado na figura abaixo.

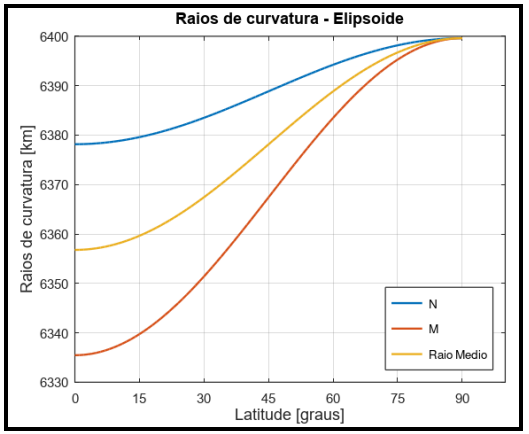


Gráfico mostrando os raios de curvatura do elipsoide de revolução.

Fim do arquivo