UNIVERSIDADE DO ESTADO DE MINAS GERAIS | UEMG UNIDADE ITUIUTABA	Universidade do Estado de Minas Gerais Unidade Ituiutaba Curso de Engenharia Elétrica

APOSTILA DE

PROGRAMAÇÃO MATLAB

Prof. Walteno Martins Parreira Júnior

www.waltenomartins.com.br

waltenomartins@yahoo.com

2015

SUMÁRIO

1 APRESENTAÇÃO

1.1 O que é o MATLAB?

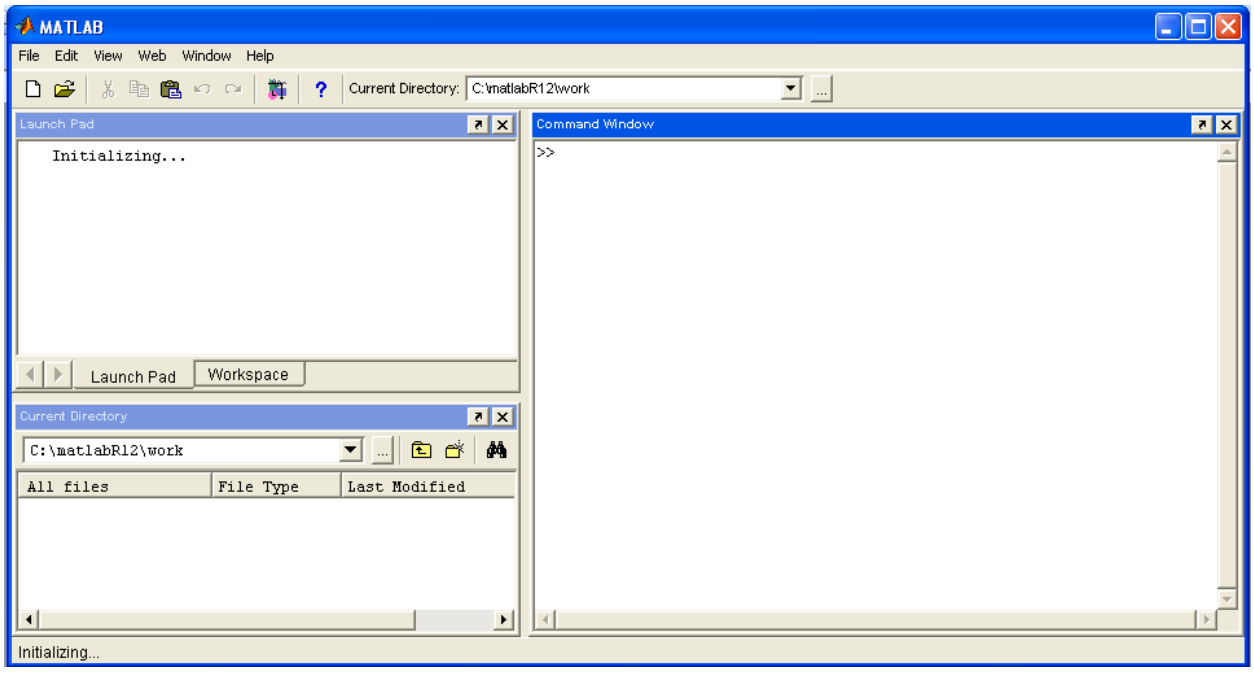
MATLAB é um "software" interativo de alto desempenho voltado para o cálculo numérico. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional.

O MATLAB é um sistema interativo cujo elemento básico de informação é uma matriz que não requer dimensionamento. Esse sistema permite a resolução de muitos problemas numéricos em apenas uma fração do tempo que se gastaria para escrever um programa semelhante em linguagem Fortran, Basic ou C. Além disso, as soluções dos problemas são expressas no MATLAB quase exatamente como elas são escritas matematicamente.

1.2 Carregando o MATLAB

No Gerenciador de Programas do Microsoft Windows deve-se abrir o grupo de programas do MATLAB for Windows, que contém o ícone do aplicativo MATLAB. Um duplo clique no ícone MATLAB carrega o aplicativo MATLAB.

Quando o MATLAB é carregado, duas janelas são exibidas: a Janela de Comando (Command Windows) e Janela Gráfica (Graphic Windows). A Janela de Comando é ativada quando se inicializa o MATLAB, e o "prompt" padrão (>>) é exibido na tela.



A partir desse ponto, o MATLAB espera as instruções do usuário.

O MATLAB faz cálculos simples e científicos como uma calculadora. Por exemplo, suponha que o aluno compra 3 objetos que custam 25 reais cada e 5 objetos que custam 12 reais cada. Quanto custou a compra?

No MATLAB você pode-se resolver este problema de pelo menos duas maneiras. A mais simples é digitando no prompt de comando:

```
>> 3*25 + 5*12
```

A resposta será:

```
ans =

   135
```

Observar que no MATLAB a multiplicação tem precedência sobre a adição. Note também que ele chamou o resultado de ans.

Alternativamente, você pode-se usar variáveis para armazenar informação. Deste modo, digitando no prompt de comando:

```
>> q1=3, p1=25, q2=5, p2=12
```

A resposta será:

```
q1 =

     3

p1 =

    25

q2 =

     5

p2 =

    12
```

E digitando no prompt de comando:

```
>> total=q1*p1+q2*p2
```

A resposta será:

```
total =

   135
```

Explicando a ação. Primeiro, foram criadas quatro variáveis, q1, p1, q2 e p2, atribuindo a elas os seus respectivos valores. Observar que no MATLAB o sinal de igual tem um sentido diferente daquele da Matemática. Aqui, igual significa atribuição. O que estiver à direita do sinal de igual é ``colocado'' na variável que estiver à esquerda. Finalmente, cria-se uma variável chamada total que recebeu o valor total da compra. Usa-se a vírgula para separar os comandos que são dados em uma mesma linha. Esta separação poderia ser feita com ponto e vírgula. Mas, neste caso o MATLAB não mostra os resultados dos comandos executados. Usando o exemplo anterior:

```
>> q1=3; p1=25; q2=5; p2=12;

>> total=q1*p1+q2*p2;
```

Em qualquer momento, pode-se ver o valor que está contido em uma variável, simplesmente digitando no prompt o nome dela.

```
>> total
```

total =

135

1.3 Área de Trabalho

Comandos que foram dados anteriormente podem ser obtidos novamente usando as teclas e ↓. Por exemplo, pressionando a tecla uma vez você obtém o último comando digitado no prompt. Pressionando repetidamente a tecla se obtém os comandos digitados anteriormente, um de cada vez na direção para trás. Analogamente, pressionando-se a tecla ↓, mas na direção para frente. Mais ainda, digitando no prompt os primeiros caracteres de um comando dado anteriormente e então pressionando-se a tecla , obtém-se o comando mais recente tendo aqueles caracteres iniciais. Em qualquer momento, as teclas ←, → podem ser usadas para se mover o cursor dentro de um comando, no prompt. Desta forma um comando pode ser corrigido, além das teclas **Delete** e **Backspace**.

O MATLAB oferece as seguintes operações aritméticas:

>> *a+b* soma a e b. Por exemplo, 5+6.

>> *a-b* subtrai a de b. Por exemplo, 15-12.

>> *a*b* multiplica a por b. Por exemplo, 3.14*0.15.

>> *a/b* divide a por b. Por exemplo, 32/4.

>> *a^b* calcula a elevado a b. Por exemplo, 5^(1/2).

Os operadores são:

- \+ Adição,
- \- Subtração,
- * Multiplicação,
- / Divisão,
- ^ Exponenciação.

A ordem com que são avaliadas as expressões é dada pela seguinte regra: expressões são avaliadas da esquerda para a direita, com a potência tendo a mais alta precedência, seguida pela multiplicação e divisão que têm igual precedência, seguidas pela adição e subtração que têm igual precedência. Parêntesis podem ser usados para alterar esta ordem. Sendo que neste caso, os parêntesis mais internos são avaliados antes dos mais externos.

Exemplo de operação matemática:

>>5 + 5

A resposta será:

ans =
10

Este resultado significa que a sua resposta foi armazenada na variável 'ans', pois todos os resultados devem ser armazenados em uma variável.

No MATLAB, as variáveis são declaradas automaticamente, portanto basta fazer uma atribuição. Por exemplo:

```
>>a = 5
a =
    5
```

Neste caso, a variável 'a' recebe o valor 5, assim o mesmo cálculo anterior pode ser:

```
>>b = a + a
b =
    10
```

Obs.: Colocando '**;** ' no final de cada sentença, o MATLAB não retorna a resposta.

Pode-se solicitar o MATLAB apresentar uma lista das variáveis que ele conhece, utilizando o comando who:

```
» who
Your variables are:
a       ans         b
```

Observe que o MATLAB não informa o valor das variáveis, mas somente lista seus nomes. Para descobrir seus valores, basta introduzir seus nomes após o prompt do MATLAB ou usar o comando **whos**.

```
>> whos
  Name      Size        Bytes  Class
  a         1x1             8  double array
  ans       1x1             8  double array
  b         1x1             8  double array
Grand total is 3 elements using 24 bytes
```

Para entrar com uma matriz pequena, por exemplo, usa-se:

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

colocando colchetes em volta dos dados e separando as linhas por ponto e vírgula.. Quando se pressiona a tecla <enter> o MATLAB responde com

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

Para inverter esta matriz usa-se

```
>> B = inv(A)
```

e o MATLAB responde com o resultado. Observar que neste caso em particular, como o determinante é muito próximo de zero (0), há uma mensagem de erro, pois o resultado pode não ter validade.

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.541976e-018.

B =
*1.0e+016 **

-0.4504	*0.9007*	*-0.4504*
0.9007	*-1.8014*	*0.9007*
-0.4504	*0.9007*	*-0.4504*

Quando a matriz possui um determinante não é apresentada a mensagem de advertência e é apresentada a matriz inversa:

```
>> A = [1 2 3; 4 5 6; 1 4 8]
A =
    1    2    3
    4    5    6
    1    4    8
>> B = inv(A)
B =
   -5.3333    1.3333    1.0000
    8.6667   -1.6667   -2.0000
   -3.6667    0.6667    1.0000
```

1.4 Editor de Linhas de Comando

As teclas com setas podem ser usadas para se encontrar comandos dados anteriormente, para execução novamente ou sua reedição. Por exemplo, suponha que você entre com

>> log (sqt(tan(pi/5)))

Como para calcular a raiz quadrada o comando certo é ***sqrt***, o MATLAB responde com uma mensagem de erro:

??? Undefined funcion or variable sqt.

Ao invés de reescrever a linha inteira, simplesmente pressione a tecla "seta para cima". O comando errado retorna, e você pode, então, mover o cursor para trás usando a tecla "seta para esquerda" ou o ponto de inserção com o "mouse" ao lugar apropriado para inserir a letra "**r**". Então, o comando retorna a resposta apropriada:

>> log (sqrt(tan(pi/5)))

ans =
-0.1597

Além das teclas com setas, pode-se usar outras teclas para reeditar a linha de comando. A seguir é dada uma breve descrição destas teclas:

↑	retorna a linha anterior
↓	retorna a linha posterior
←	move um espaço para a esquerda
→	move um espaço para a direita
Ctrl ←	move uma palavra para a esquerda
Ctrl →	move uma palavra para a direita
Home	move para o começo da linha
End	move para o final da linha
Del	apaga um caracter a direita
Backspace	apaga um caracter a esquerda

2 INTRODUÇÃO

O MATLAB trabalha essencialmente com um tipo de objeto, uma matriz numérica retangular podendo conter elementos complexos (deve-se lembrar que um escalar é uma matriz de dimensão l x l e que um vetor é uma matriz que possui somente uma linha ou uma coluna).

2.1 Entrando com Matrizes Simples

As matrizes podem ser introduzidas no MATLAB por diferentes caminhos:

- digitadas na Janela de Comando (lista explícita de elementos),
- geradas por comandos e funções,
- criadas em arquivos ".m",
- carregadas a partir de um arquivo de dados externo.

O método mais fácil de entrar com pequenas matrizes no MATLAB é usando uma lista explícita. Os elementos de cada linha da matriz são separados por espaços em branco ou vírgulas e as colunas separadas por ponto e vírgula, colocando-se colchetes em volta do grupo de elementos que formam a matriz. Formula é:

A(linha; coluna)

Por exemplo, entre com a expressão

>> A=[1 2 3;4 5 6;7 8 9]

Pressionando <enter> o MATLAB mostra o resultado

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

A matriz **A** é salva na memória RAM do computador, ficando armazenada para uso posterior.

As matrizes podem, também, ser introduzidas linha a linha, o que é indicado para matrizes de grande dimensão. Por exemplo:

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

Outra maneira para entrar com matrizes no MATLAB é através de um arquivo no formato texto com extensão "**.m**". Por exemplo, se um arquivo chamado "**gera.m**" contém estas três linhas de texto,

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

então a expressão "**gera**" lê o arquivo e introduz a matriz A.

```
>>gera
```

O comando **load** pode ler matrizes geradas pelo MATLAB e armazenadas em arquivos binários ou matrizes geradas por outros programas armazenadas em arquivos ASCII.

2.2 Elementos das Matrizes

Os elementos das matrizes podem ser qualquer expressão do MATLAB, por exemplo.

```
>> x = [-1.3 sqrt(2) ((1+2+3)*4/5)^2]
```

resulta em

```
x =
-l.3000          1.4142          23.0400
```

Um elemento individual da matriz pode ser reverenciado com índice entre parênteses. Continuando o exemplo com a inserção de um valor na posição 6 do vetor:

```
>> x(6) = abs(x(l))
```

produz:

```
x =
-1.3000    1.4142    23.0400    0    0    1.3000
```

Note que a dimensão do vetor x é aumentada automaticamente para acomodar o novo elemento e que os elementos do intervalo indefinido são estabelecidos com o valor zero.

Grandes matrizes podem ser construídas a partir de pequenas matrizes. Por exemplo, pode-se anexar outra linha na matriz **A** usando

```
>> r= [ l0 11 12];
>> A= [A;r]
```

que resulta em

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

10 11 13

Note que o vetor **r** não foi listado porque ao seu final foi acrescentado ";".

Pequenas, matrizes podem ser extraídas de grandes matrizes usando ";". Por exemplo,

>> *A = A(1:3,:);*

seleciona as três primeiras linhas e todas as colunas da matriz **A** atual, modificando-a para sua forma original. Observe que apenas as linhas foram especificadas.

>> *A = A(1:3);*

seleciona a primeira linha e todas as colunas da matriz **A** atual, modificando-a para uma matriz linha.

A =
1 4 7

Considerando a matriz A e executando o comando

>> B=A(2:3)

O resultado é:

B =
4 7

Pode-se observar que o comando copia parte da primeira coluna a partir da segunda linha, pois foram omitidos os valores para a coluna e identificado somente as linhas. Assim, o vetor B contem dois valores.

E o exemplo a seguir determina que as duas primeiras linhas sejam atribuídas para a nova matriz C. Observe que o caractere dois pontos indica todas as colunas da matriz.

>> C= A(1:2,:)

O resultado é:

C =
1 2 3
4 5 6

2.3 Declarações e Variáveis

O MATLAB é uma linguagem de expressões. As expressões usadas são interpretadas e avaliadas pelo sistema. As declarações no MATLAB são frequentemente da forma

>> variável = expressão

ou simplesmente

>> expressão

As expressões são compostas de operadores e outros caracteres especiais, de funções e dos nomes das variáveis. A avaliação das expressões produzem matrizes, que são então mostradas na tela e atribuídas às variáveis para uso futuro. Se o nome da variável e o sinal de igualdade "**=**" são omitidos, a variável com o nome **ans**, que representa a palavra "answer" (resposta), é automaticamente criada. Por exemplo, digite a expressão:

>> 1900/81

que produz

ans=
23.4568

Se o último caractere da declaração é um ponto e vírgula, ";", a impressão na tela é suprimida, mas a tarefa é realizada. Esse procedimento é usado em arquivos com extensão ".m" e em situações onde o resultado é uma matriz de grandes dimensões e temos interesse em apenas alguns dos seus elementos.

Se a expressão é tão grande que não cabe em apenas uma linha, pode-se continuar a expressão na próxima linha usando um espaço em branco e três pontos,"...", ao final das linhas incompletas. Por exemplo,

>> s = l - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...

>> - 1/8 + 1/9 - l/10 + 1/11 - 1/12 + 1/13;

calcula o resultado da série, atribuindo a somatória à variável **s**, mas não imprime o resultado na tela. Note que os espaços em branco entre os sinais "=", "+" e "-" são opcionais, mas o espaço em branco entre "1/7" e "..." é obrigatório.

As variáveis e funções podem ser formadas por um conjunto de letras, ou por um conjunto de letras e números, onde somente os primeiros 19 caracteres do conjunto são identificados. O MATLAB faz distinção entre letras maiúsculas e minúsculas, assim **a** e **A** não são as mesmas variáveis. Todas as funções devem ser escritas em letras minúsculas: **inv(A)** calcula a inversa de **A**, mas **INV(A)** é uma função indefinida.

2.4 Obtendo Informações da Área de Trabalho

Os exemplos de declarações mostrados nos itens acima criaram variáveis que são armazenadas na *Área de Trabalho* do MATLAB. Executando

>> who

obtêm-se uma lista das variáveis armazenadas na Área de Trabalho:

Your variables are:

A ans r s x

Que mostra as cinco variáveis geradas em nossos exemplos, incluindo **ans**.

Uma informação mais detalhada mostrando a dimensão de cada uma das variáveis correntes é obtido com **whos** que para nosso exemplo produz:

Name	*Size*	*Elements*	*Bytes*	*Density*	*Complex*
A	*3 by 3*	*9*	*72*	*Full*	*No*
ans	*1 by 1*	*1*	*8*	*Full*	*No*
R	*1 by 3*	*3*	*24*	*Full*	*No*
S	*1 by 1*	*1*	*8*	*Full*	*No*
X	*1 by 6*	*6*	*48*	*Full*	*No*

Grand total is 20 elements using 160 bytes

Cada elemento de uma matriz real requer 8 bytes de memória, assim nossa matriz **A** de dimensão 3x3 usa 72 bytes e todas variáveis utilizadas um total de 160 bytes.

2.5 Números e Expressões Aritméticas

A notação decimal convencional, com ponto decimal opcional e o sinal de menos, é usada para números. A potência de dez pode ser incluída como um sufixo. A seguir são mostrados alguns exemplos de números aceitos:

3	-99	0.00001
9.637458638	1.602E-20	6.06375e23

As expressões podem ser construídas usando os operadores aritméticos usuais e as regras de precedência:

1	^	exponenciação
2	/	divisão a direita
2	\	divisão a esquerda
3	*	multiplicação
4	+	adição

4	-	subtração

Deve-se notar que existem dois símbolos para divisão: as expressões 1/4 e 4\1 possuem o mesmo valor numérico, isto é, 0,25. Parênteses são usados em sua forma padrão para alterar a precedência usual dos operadores aritméticos.

2.6 Números e Matrizes Complexas

Números complexos são permitidos em todas as operações e funções no MATLAB. Os números complexos são introduzidos usando-se as funções especiais **i** e **j**. Por exemplo:

```
>> z= 3 + 4*i
```

ou

```
>> z= 3 +4*j
```

Outro exemplo é

```
>> w= r * exp(i*theta)
```

As seguintes declarações mostram dois caminhos convenientes para se introduzir matrizes complexas no MATLAB:

```
>> A= [1 2; 3 4]+i*[5 6;7 8]
```

e

```
>> A= [1+5*i 2+6*i; 3+7*i 4+8*i]
```

que produzem o mesmo resultado.

Se **i** ou **j** forem usados como variáveis, de forma que tenham seus valores originais modificados, uma nova unidade complexa deverá ser criada e utilizada de maneira usual:

```
>> ii = sqrt(-1);
```

```
>> z = 3 + 4*ii
```

2.7 Formato de Saída

O formato numérico exibido na tela pode ser modificado utilizando-se o comando **format**, que afeta somente o modo como as matrizes são mostradas, e não como elas são computadas ou salvas (o MATLAB efetua todas operações em dupla precisão).

Se todos os elementos das matrizes são inteiros exatos, a matrizes é mostrada em um formato sem qualquer ponto decimal. Por exemplo,

>> x = [-1 0 1]

sempre resulta em

x =
-1 0 1

Se pelo menos um dos elementos da matriz não é inteiro exato, existem várias possibilidades de formatar a saída. O formato "default", chamado de formato **short**, mostra aproximadamente 5 dígitos significativos ou usam notação científica. Por exemplo a expressão

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

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

format short	1.3333 0.0000
format short e	1.3333e+000 1.2345e-006
format long	1.33333333333333 0.000000123450000
format long e	1.333333333333333e+000 1.234500000000000e-006
format hex	3ff5555555555555 3eb4b6231abfd271
format rat	4/3 1/810045
format bank	1.33 0.00
format +	++

Com o formato **short** e **long**, se o maior elemento da matriz é maior que 1000 ou menor que 0.001, um fator de escala comum é aplicado para que a matriz completa seja mostrada. Por exemplo,

*>> x = 1.e20*x*

resultado da multiplicação será mostrado na tela.

*X = l.0e+20 **
1.3333 0.0000

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.

2.8 As Facilidades do HELP (Ajuda)

O MATLAB possui um comando de ajuda **(help)** que fornece informações sobre a maior parte dos tópicos. Digitando

>> help

obtêm-se uma lista desses tópicos disponíveis:

HELP topics:

c:\matlab -Establish MATLAB session parameters.

matlab\general -General purpose commands.

matlab\ops -Operators and special characters.

matlab\lang -Language constructs and debugging.

matlab\elmat -Elementary matrices and matrix manipulation.

matlab\specmat -Specialized matrices.

matlab\elfun -Elementary math functions.

matlab\specfun -Specialized math functions.

matlab\matfun -Matrix functions - numerical linear algebra.

matlab\datafun -Data analysis and Fourier transform functions.

matlab\polyfun -Polynomial and interpolation functions.

matlab\funfun -Function functions: nonlinear numerical methods.

matlab\sparfun -Sparse matrix functions.

matlab\plotxy -Two dimensional graphics.

matlab\plotxyz -Three dimensional graphics.

matlab\graphics -General purpose graphics functions.

matlab\color -Color control and lighting model functions.

matlab\sounds -Sound processing functions.

matlab\strfun -Character string functions.

matlab\iofun -Low-level file I/0 functions.

matlab\demos -Demonstrations and samples.

simulink\simulink -SIMULINK model analysis.

simulink\blocks -SIMULINK block library.

simulink\simdemos -SIMULINK demonstrations and samples.

nnet\examples - Neural Network Toolbox examples.

nnet\nnet - Neural Network Toolbox.

Para obter informações sobre um tópico específico, digite **help tópico.** Por exemplo,

>> help plotxy

que fornece uma lista de todos os comandos relacionados com gráficos bidimensionais:

Two dimensional graphics.

Elementary X-Y graphs
plot - Linear plot.
loglog - Log-log scafe plot.
semilogx - Semi-log scale plot.
semilogy - Semi-log scale plot.
fill - Draw filled 2-D polygons.

Specialized X-Y graphs.
polar - Polar coordinate plot.
bar - Bar graph.
stem - Discrete sequence or & "stemm" plot.
stairs - Stairstep plot.
errorbar - Error bar plot.
hist - Histogram plot.
rose - Angle histogram plot.
compass - Compass plot.
feather - Feather plot.
fplot - Plot function
comet - Comet-like trajectory.

Graph annotation.
title - Graph title.
xlabel - X-axis label.
ylabel - Y-axis label.
text - Text annotation.
gtext - Mouse placement of text.
grid - Grid lines.

See also PLOTXYZ, GRAPHICS

Finalmente, para obter informações sobre um comando específico, por exemplo **title**, digite:

>> help title

e informações mais detalhadas sobre este comando serão exibidas:

TITLE Titles for 2-D and 3-D plots.
TITLE ('text') adds text at the top of the current axis.
See also XLABEL, YLABEL, ZLABEL, TEXT.

Note que no exemplo mostrado para adicionar o título em um gráfico, **TITLE ('TEXT')** está escrito em letras maiúsculas somente para destacar. Deve-se lembrar que todos os comandos do MATLAB devem ser escritas em letras minúsculas, portanto, para adicionar o texto "*Título do Gráfico*" em um gráfico, digite:

>> title ('Título do Gráfico')

2.9 Funções

A importância do MATLAB vem de um conjunto extenso de funções. O MATLAB possui um grande número de funções intrínsecas que não podem ser alteradas pelo usuário. Outras funções estão disponíveis em uma biblioteca externa distribuídas com o programa original (MATLAB TOOLBOX), que são na realidade arquivos com a extensão ".m" criados a partir das funções intrínsecas. A biblioteca externa (MATLAB TOOLBOX) pode ser constantemente atualizada à medida que novas aplicações são desenvolvidas. As funções do MATLAB, intrínsecas ou arquivos ".m", podem ser utilizadas apenas no ambiente MATLAB.

As categorias gerais de funções matemáticas disponíveis no MATLAB incluem:

- Matemática elementar;
- Funções especiais;
- Matrizes elementares;
- Matrizes especiais;
- Decomposição e fatorização de matrizes;
- Análise de dados;
- Polinômios;
- Solução de equações diferenciais;
- Equações não-lineares e otimização;
- Integração numérica;
- Processamento de sinais.

As seções subsequentes mostram mais detalhes dessas diferentes categorias de funções.

3 OPERAÇÕES COM MATRIZES

As operações com matrizes no MATLAB são as seguintes:

- Adição;
- Subtração;
- Multiplicação;
- Divisão a direita;
- Divisão a esquerda;
- Exponenciação;
- Transposta;

A seguir cada uma dessas operações é mostrada com mais detalhe.

3.1 Matriz Transposta

O caractere apóstrofo, " ' " , indica a transposta de uma matriz. A declaração:

>> A = [1 2 3; 4 5 6; 7 8 0]

>> B = A'

que resulta em

```
A =     1   2   3
        4   5   6
        7   8   0
B =     1   4   7
        2   5   8
        3   6   0
```

>> x = [-1 O 2]'

```
x =
        -1
         0
         2
```

Se **Z** é uma matriz complexa, **Z'** será o conjugado complexo composto. Para obter simplesmente a transposta de **Z** deve-se usar **Z.'**, como mostra o exemplo

*>> Z = [1 2; 3 4] + [5 6; 7 8]*i*

>> Z1 = Z'

>> Z2 = Z.'

que resulta em

```
Z =
        1.0000 + 5.0000i    2.0000 + 6.0000i
        3.0000 + 7.0000i    4.0000 + 8.0000i
```

```
Z1 =
                1.0000 - 5.0000i        3.0000 - 7.0000i
                2.0000 - 6.0000i        4.0000 - 8.0000i

Z2 =            1.0000 + 5.0000i        3.0000 + 7.0000i
                2.0000 + 6.0000i        4.0000 + 8.0000i
```

3.2 Adição e Subtração de Matrizes

A adição e subtração de matrizes são indicadas, respectivamente, por "+" e "-". As operações são definidas somente se as matrizes possuem as mesmas dimensões. Por exemplo, a soma com as matrizes mostradas acima, **A + x**, não é correta porque **A** é 3x3 e **x** é 3x1. Porém,

```
>> C = A + B
```

é aceitável, pois B é a transposta de A, e o resultado da soma é

```
C =
                2       6       10
                6       10      14
                10      14      0
```

A adição e subtração também são definidas se um dos operadores é um escalar, ou seja, uma matriz 1 x l. Neste caso, o escalar é adicionado ou subtraído de todos os elementos do outro operador. Por exemplo:

```
>> y = x - 1
```

resulta em

```
y =
                -2
                -1
                1
```

3.3 Multiplicação de Matrizes

A multiplicação de matrizes é indicada por "*". A multiplicação **x*y** é definida somente se a segunda dimensão de x for igual à primeira dimensão de **y.** A multiplicação

```
>> x'* y
```

é aceitável, e resulta em

```
ans =
                4
```

É evidente que o resultado da multiplicação **y'*x** será o mesmo. Existem dois outros produtos que são transpostos um do outro.

```
>> x*y'

ans =
        2     l     -l
        0     0     0
        -4    -2    2

>> y*x'

ans =
        2     0     -4
        1     0     -2
        -1    0     2
```

O produto de uma matriz por um vetor é um caso especial do produto entre matrizes. Por exemplo **A** e **X,**

```
>> b = A'x
```

que resulta em

```
b =
        5
        8
        -7
```

Lembrando que:

$A_{3x3} * X_{3x1} = B_{3x1}$

Naturalmente, um escalar pode multiplicar ou ser multiplicado por qualquer matriz.

```
>> pi*x

ans =
        -3.1416
        0
        6.2832
```

3.4 Divisão de Matrizes

Existem dois símbolos para divisão de matrizes no MATLAB "\" e "/". Se A é uma matriz quadrada não singular, então **A\B** e **B/A** correspondem respectivamente à multiplicação à esquerda e à direita da matriz **B** pela inversa da matriz **A**, ou **inv(A)*B** e **B*inv(A)N,** mas o resultado é obtido diretamente. Em geral,

- X = A\B é a solução de A*X = B
- X = B/A é a solução de X*A = B

Por exemplo, como o vetor **b** foi definido como **A*x,** a declaração

>> z = A\b

resulta em

```
z =
        -1
         0
         2
```

3.5 Exponenciação de Matrizes

A expressão **A^p** eleva **A** à **p**-ésima potência e é definida se **A** é matriz quadrada e **p** um escalar. Se **p** é um inteiro maior do que um, a exponenciação é computada como múltiplas multiplicações. Por exemplo,

>> A^3

```
ans =
       279     360     306
       684     873     684
       738     900     441
```

3.6 Funções de Matrizes

3.6.1 Função eye

Esta função gera uma Matriz Identidade de tamanho definido pelo usuário.

```
>>eye(2)
ans =
   1   0
   0   1
```

3.6.2 Função zeros

Esta função cria uma Matriz de Zeros de tamanho definido pelo usuário.

```
>> zeros(2)
ans =
   0   0
   0   0
```

3.6.3 Função ones

Esta função cria uma matriz de 1's de tamanho definido pelo usuário

```
>> ones(2,3)
ans =
    1    1    1
    1    1    1

>> 3 * ones(2)
ans =
    3    3
    3    3
```

3.6.4 Função rand

Esta função cria uma matriz gerada com valores definidos aleatoriamente .

```
>> rand(2,3)
ans =
   0.9501    0.6068    0.8913
   0.2311    0.4860    0.7621
```

E gera uma matriz quadrada se for omitido o segundo parâmetro para a função:

```
>> rand(2)
ans =
   0.9501    0.6068
   0.2311    0.4860
```

3.6.5 Função randn

Esta função cria uma matriz gerada com valores reais aleatórios seguindo a Distribuição Normal.

```
>> randn(2,3)
ans =
  -0.4326    0.1253   -1.1465
  -1.6656    0.2877    1.1909
```

3.6.6 Função diag

Esta função cria uma matriz diagonal a partir de um vetor qualquer. Se x é um vetor, diag(x) é a matriz diagonal com x na diagonal;

```
>> x=[2,5,7]
x =
    2    5    7
>> diag(x)
ans =
    2    0    0
    0    5    0
    0    0    7
```

Se A é uma matriz quadrada, então o resultado de diag(A) é um vetor cujos componentes são os elementos da diagonal de A.

```
>> A=[3 11 5; 4 1 -3; 6 2 1];
>> diag(A)
ans =
    3
    1
    1
```

3.6.7 Função triu

Esta função devolve a parte triangular superior de uma matriz.

```
>> triu(A)
ans =
    3   11    5
    0    1   -3
    0    0    1
```

3.6.8 Função tril

Esta função devolve a parte triangular inferior de uma matriz.

```
>> tril(A)
ans =
    3    0    0
    4    1    0
    6    2    1
```

3.6.9 Função sum

Esta função devolve a soma dos elementos das colunas de uma matriz.

```
>> sum(A)
ans =
   13   14    3
```

Para encontrar a soma das linhas da matriz a, faz-se a soma da transposta da matriz:

```
>> sum(A')
ans =
   19    2    9
```

3.6.9 Função mean

Esta função devolve a média dos elementos das colunas de uma matriz.

```
>> mean(A)
>> mean(A)
ans =
   4.3333    4.6667    1.0000
```

Para encontrar a média das linhas da matriz a, faz-se a soma da transposta da matriz:

```
>> mean(A')
ans =
   6.3333    0.6667    3.0000
```

3.6.10 Função inv

Esta função devolve a matriz inversa de uma matriz definida.

```
>> inv(A)
ans =
  -0.0332    0.0047    0.1801
   0.1043    0.1280   -0.1374
  -0.0095   -0.2844    0.1943
```

3.6.11 Função sort

Esta função devolve uma matriz com os valores ordenados por coluna em ordem crescente de uma matriz definida.

```
A =
    3    11     5
    4     1    -3
    6     2     1
>> sort(A)
ans =
    3     1    -3
    4     2     1
    6    11     5
```

3.6.12 Função prod

Esta função devolve uma matriz com o produto das colunas de uma matriz definida.

```
A =
    3    11     5
    4     1    -3
    6     2     1
>> prod(A)
ans =
   72    22   -15
```

4 OPERAÇÕES COM CONJUNTOS

O termo *operações com conjuntos* é usado quando as operações aritméticas são realizadas entre os elementos que ocupam as mesmas posições em cada matriz (elemento por elemento). As operações com conjuntos são feitas como as operações usuais, utilizando-se dos mesmos caracteres ("*****", "**/**", "****", "^" e " ‘ ") precedidos por um ponto "**.**" ("**.***", "**./**", "**.**", ".^" e " **.‘** ").

4.1 Adição e Subtração

Para a adição e a subtração, a operação com conjuntos e as operações com matrizes são as mesmas. Deste modo os caracteres "+" e "-" podem ser utilizados tanto para operações com matrizes como para operações com conjuntos.

4.2 Multiplicação e Divisão

A multiplicação de conjuntos é indicada por "**.***". Se **A** e **B** são matrizes com as mesmas dimensões, então **A.*B** indica um conjunto cujos elementos são simplesmente o produto dos elementos individuais de **A** e **B**. Por exemplo, se

>> x = [1 2 3]; y = [4 5 6];

então,

>> z = x . y*

resulta em

z=

4 10 18

As expressões **A./B** e **A.\B** formam um conjunto cujos elementos são simplesmente os quocientes dos elementos individuais de **A** e **B.** Assim,

>> z = x .\ y

resulta em

z =

4.0000 2.5000 2.0000

4.3 Exponenciação

A exponenciação de conjuntos é indicada por "**.**^". A seguir são mostrados alguns exemplos usando os vetores **x** e **y.** A expressão

>> z = x .^ y

resulta em

z =

l 32 729

A exponenciação pode usar um escalar.

>> z = x.^2

z =

l 4 9

Ou, a base pode ser um escalar.

>> z = 2.^[x y]

z =

2 4 8 16 32 64

4.4 Operações Comparativas

Estes são os seis operadores usados para comparação de duas matrizes com as mesmas dimensões:

<	menor
<=	menor ou igual
>	maior
>=	maior ou igual
==	igual
~=	diferente

A comparação é feita entre os pares de elementos correspondentes e o resultado é uma matriz composta dos números **um** e **zero**, com o valor **um** representando **Verdadeiro** e **zero** representando o valor **Falso**. Por exemplo**,**

>> 2 + 2 ~= 4

ans =

0

Podem-se usar também os operadores lógicos **&** (e) e **I** (ou). Por exemplo,

```
>> 1= = 1 & 4 = = 3
ans =
            0

>> 1 = = 1 | 4 = = 3
ans =
            1
```

5 MANIPULAÇÃO DE VETORES E MATRIZES

O MATLAB permite a manipulação de linhas, colunas, elementos individuais e partes de matrizes.

5.1 Gerando Vetores

Os dois pontos, " **:** ", é um caractere importante no MATLAB. A declaração

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

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

```
x =
     1     2     3     4     5
```

Outros incrementos, diferentes de um, podem ser usados.

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

Pode-se, também, gerar vetores usando a função **linspace**. Por exemplo,

```
>> k = linspace (0, l, 6)
k =
     0    0.2000    0.4000    0.6000    0.8000    1.0000
```

gera um vetor linearmente espaçado de 0 a 1, contendo 6 elementos.

5.2 Manipulando Elementos das Matrizes

Um elemento individual da matriz pode ser indicado incluindo os seus subscritos entre parênteses. Por exemplo, dada a matriz A:

A =

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

a declaração

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

resulta em

A =

1	*2*	*3*
4	*5*	*6*
7	*8*	*10*

Um subscrito pode ser um vetor. Se X e V são vetores, então **X(V)** é **[X(V(1)), X(V(2)), ... X(V(n))]**. Para as matrizes, os subscritos vetores permitem o acesso à submatrizes contínuas e descontínuas. Por exemplo, suponha que A é uma matriz 10x10.

A =

92	*99*	*11*	*18*	*15*	*67*	*74*	*51*	*58*	*40*
98	*80*	*17*	*14*	*16*	*73*	*55*	*57*	*64*	*41*
14	*81*	*88*	*20*	*22*	*54*	*56*	*63*	*70*	*47*
85	*87*	*19*	*21*	*13*	*60*	*62*	*69*	*71*	*28*
86	*93*	*25*	*12*	*19*	*61*	*68*	*75*	*52*	*34*
17	*24*	*76*	*83*	*90*	*42*	*49*	*26*	*33*	*65*
23	*15*	*82*	*89*	*91*	*48*	*30*	*32*	*39*	*66*
79	*16*	*13*	*95*	*97*	*29*	*31*	*38*	*45*	*72*
10	*12*	*94*	*96*	*78*	*35*	*37*	*44*	*46*	*53*
11	*18*	*100*	*77*	*84*	*36*	*43*	*50*	*27*	*59*

>> A(1:5,3)

ans =

11
17
88
19
25

Que especifica uma submatriz 5x1, ou vetor coluna, que consiste dos cinco primeiros elementos da terceira coluna da matriz **A**. Analogamente, o comando

>> A(1:3,7:10)

ans =

74	*51*	*58*	*40*
55	*57*	*64*	*41*
56	*63*	*70*	*47*

Que é uma submatriz 3x4 que consiste das primeiras tres linhas e as últimas quatro colunas.

Utilizando os dois pontos no lugar de um subscrito denota-se todos elementos da linha ou coluna. Por exemplo,

```
>> A(1:2:5,:)
ans =
    92    99    11    18    15    67    74    51    58    40
    14    81    88     2    22    54    56    63    70    47
    86    93    25    12    19    61    68    75    52    34
```

O comando gera uma submatriz 3x10 que consiste da primeira, terceira e quinta linhas e todas colunas da matriz **A**.

Muitos efeitos sofisticados são obtidos usando submatrizes em ambos os lados das declarações. Por exemplo, sendo **B** uma matriz 10x10 unitária,

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

a declaração,

```
>> B(1:2:7,6:l0) = A(S:-1:2,1:5)
1     1     1     1     1    86    93    25    12    19
1     1     1     1     1     1     1     1     1     1
1     1     1     1     1    85    87    19    21    13
1     1     1     1     1     1     1     1     1     1
1     1     1     1     1    14    81    88    20    22
1     1     1     1     1     1     1     1     1     1
1     1     1     1     1    98    80    17    14    16
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     1     1     1     1     1
```

6 ARQUIVOS ".m"

Os comandos do MATLAB são normalmente digitados na Janela de Comando, onde uma única linha de comando é introduzida e processada imediatamente. O MATLAB é também capaz de executar seqüências de comandos armazenadas em arquivos.

Os arquivos que contêm as declarações do MATLAB são chamadas arquivos "**.m**", e consiste de uma sequência de comandos normais do MATLAB, possibilitando incluir outros arquivos ".m" escritos no formato texto (ASCII).

Para editar um arquivo texto na Janela de Comando do MATLAB selecione **New M-File** para criar um novo arquivo ou **Open M-File** para editar um arquivo já existente, a partir do menu **File**. Os arquivos podem, também, ser editados fora do MATLAB utilizando qualquer editor de texto.

Existem alguns comandos e declarações especiais para serem usados nos arquivos, por exemplo:

```
%Plota uma função y=ax^2 + bx + c no intervalo -5<x<5
clear
clc
a=input('a =');
b=input('b =');
c=input('c =');
x=-5:0.1:5;
y=a*x.^2+b*x+c;
plot(y)
figure(1)
pause
close
```

O caractere **%** é usado para inserir um comentário no texto, o comando **clear** apaga todos os dados da memória, o comando **input** é usado quando se deseja entrar com um dado a partir da Janela de Comando, **pause** provoca uma pausa na execução do arquivo até que qualquer tecla seja digitada, **clc** limpa a Janela de Comando, **figure(1)** mostra a Janela Gráfica número 1 e **close** fecha todas as Janelas Gráficas.

Comentários: são linhas usadas para explicações ou anotações no arquivo e não são executadas. Observe que texto apos o % na mesma linha não será executado.

```
% comentário de linha
```

Entrada de dados: Comando usado para receber um dado numérico do usuário

```
<variável> = input('<mensagem>');
```

Exemplo:

```
n1 = input('Digite um numero: ')
<variável> = input('<mensagem>', 's')
```

Exemplo:

n1 = input('Digite seu nome: ', 's')

Saída de mensagem: mostrar um texto ou conteúdo de uma variável:

disp('<mensagem>');

Exemplo:

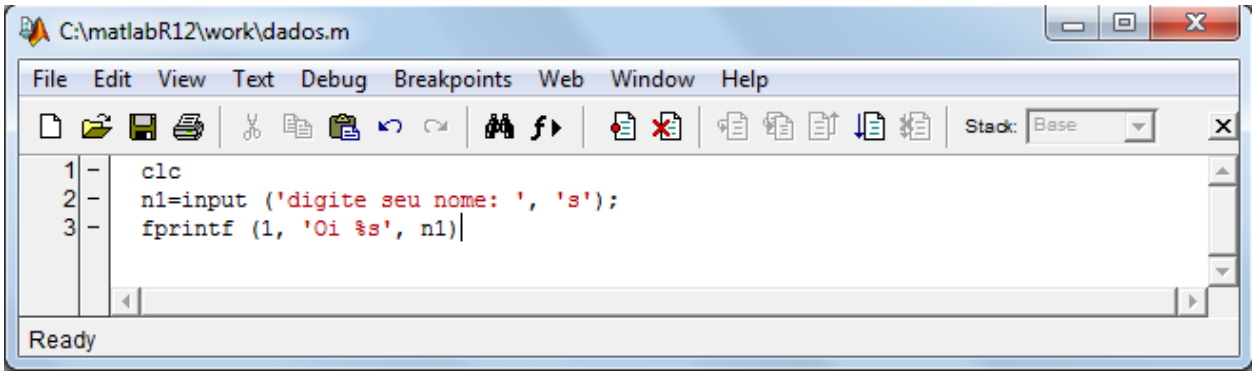
disp('Total calculado: ');
disp(soma);

Limpar a tela: comando que limpa a tela.

Exemplo:

clc;

Exemplo de arquivo:



Para executar o arquivo "dados.m", é só digitar o nome do arquivo no prompt de comando:

>>dados

A tela ficará limpa e aparecerá a solicitação:

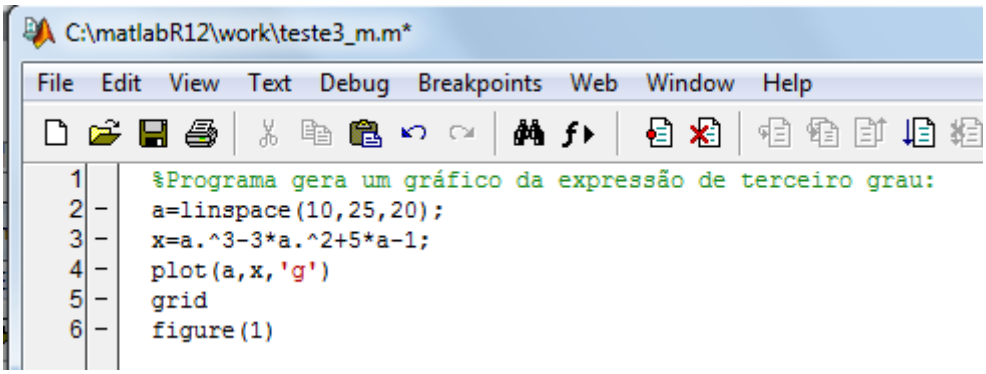
digite seu nome: walteno

E continuando a execução, o programa responderá:

Oi walteno
>>

Exemplo de programa:

1) Desenvolver um programa que gera um vetor com 20 itens no intervalo entre 10 e 25 e depois gera um gráfico da expressão $A^3 - 3A^2 + 5A - 1$

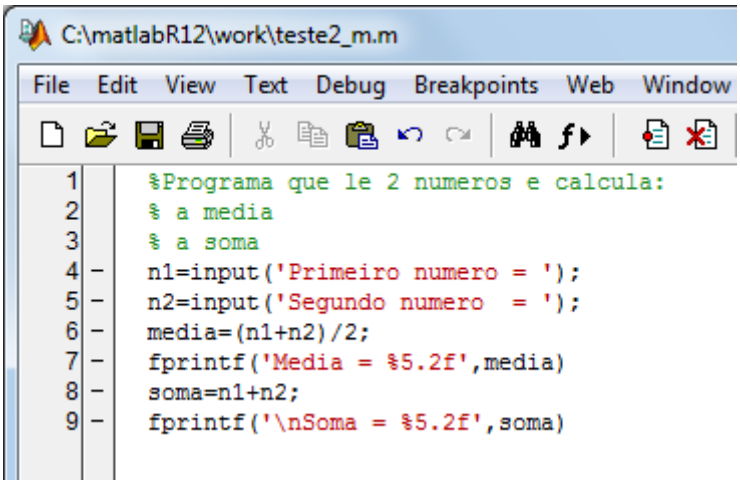


E a execução do programa é

>> teste3_m

2) Desenvolver um programa que lê dois números inteiros do teclado e depois imprime:

a) A media dosdois números;
b) A soma dos dois números;



E a execução do programa é

>> teste2_m
Primeiro numero = 10
Segundo numero = 20
Media = 15.00
Soma = 30.00

7 CONTROLE DE FLUXO

Os comandos que controlam o fluxo especificam a ordem em que a execução é feita. No MATLAB os comandos são semelhantes aos usados em C, mas com estrutura diferente.

7.1 Laço for

O laço **for** é o controlador de fluxo mais simples usado na programação MATLAB. Exemplo:

```
>> for i=1:5,
    X(i)=i^2;
    end
```

pode-se notar que o laço **for** é dividido em três partes:

- A primeira parte (**i=1**) é realizada uma vez, antes do laço ser inicializado.
- A segunda parte é o teste ou condição que controla o laço, (**i<=5**).
- Esta condição é avaliada; se verdadeira, o corpo do laço (**X(i)=i^2**) é executado.

A terceira parte acontece quando a condição se torna falsa e o laço termina.

O comando **end** é usado como limite inferior do corpo do laço.

É comum construções em que conjuntos de laços **for** são usados principalmente com matrizes:

```
>> for i=1:8
      for j=1:8
          a(i,j)=i+j;
          b(i,j)=i-j;
       end
     end
>> c=a+b
c =
    2    2    2    2    2    2    2    2
    4    4    4    4    4    4    4    4
    6    6    6    6    6    6    6    6
    8    8    8    8    8    8    8    8
   10   10   10   10   10   10   10   10
   12   12   12   12   12   12   12   12
   14   14   14   14   14   14   14   14
   16   16   16   16   16   16   16   16
```

7.2 Laço while

No laço **while** apenas a condição é testada. Por exemplo, na expressão:

```
a = 1; b = 15;
while a<b                 loop
   clc                    limpa a tela
   a = a+1                faz a operação soma
   b = b-1                faz a operação subtração
   pause(1)               dá uma pequena pausa na execução
end                       final do loop
disp('fim do loop')       imprime a mensagem e termina o programa
```

a condição **a<b** é testada. Se ela for verdadeira o corpo do laço será executado.

Então a condição é testada, e se verdadeira o corpo será executado novamente. Quando o teste se tornar falso o laço terminará, e a execução continuará no comando que segue o laço após o **end**.

7.3 Declarações if e break

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

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

Os valores de **i e j** variam de 1 a 5, varrendo toda a matriz **A.** Se **(if) i** for igual a **j**, **A(i,j)=2**, ou se **(elseif)** o valor absoluto de **i-j** for igual a 1, **A(i,j)=-1**, ou **(else) A(i,j)=0**, se nenhuma das condições anteriores forem satisfeitas.

É conveniente, às vezes, controlarmos a saída deu m laço de outro modo além do teste, no início ou no fim do mesmo. O comando **break** permite uma saída antecipada de um **for** ou **while.** Um comando **break** faz com que o laço mais interno seja terminado imediatamente.

Por exemplo, o programa que permite modificar os elementos de uma matriz A gerada com os elementos iguais a um, e no final imprime a nova matriz A.

C:\matlabR12\work\matriz_controle.m

File Edit View Text Debug Breakpoints Web Window Help

```
%modifica os elementos da matriz A
clc
x = 's';
A=ones(5);
for i = 1:5
  if x == 'q'
    break
    end
  j = 1;
  while j<=5
    ['A(',num2str(i),',',num2str(j),')=',num2str(A(i,j))]
    x = input('Modifica? (s-sim, n-não, p-próxima linha, q-sair) =>','s');
    if x == 's',
      A(i,j) = input('Entre com o novo valor de A(i,j) ==> ');
      j=j+1;
      clc
    end
    if x == 'n'
      j=j+1;
      clc
    end
    if x == 'p'
      clc
      break
    end
    if x == 'q',
      clc
      break
    end
  end
end
A
```

7.4 Exemplo de programa com o uso de controle de fluxo.

1) Desenvolver um programa que lê a altura e o peso de uma pessoa e calcula o IMC e depois solicita a informação se vai repetir a operação para novos valores:

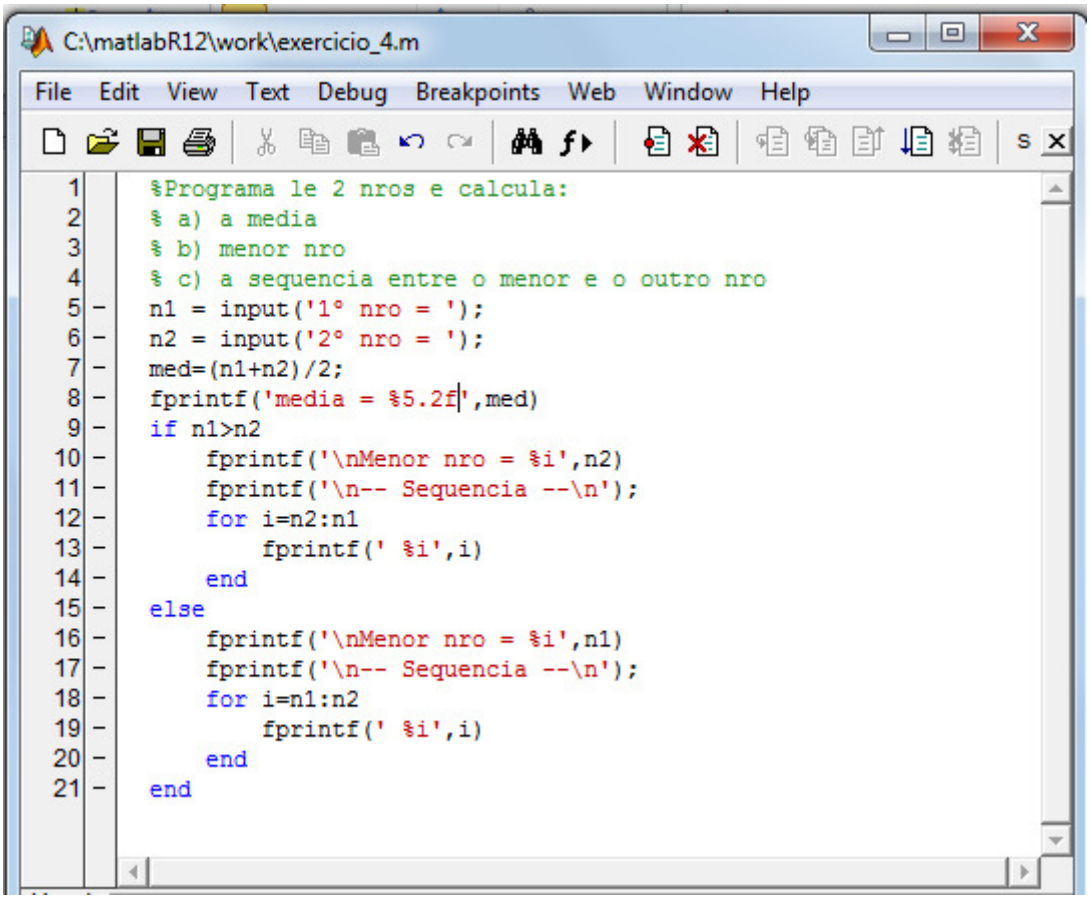
C:\matlabR12\work\aula_7_ex_5.m

File Edit View Text Debug Breakpoints Web Window Help

```
% Programa que calcula o IMC de uma pessoa
clc
resp = 's';
while (resp=='s')
    peso = input ('Peso = ');
    altura = input ('Altura  = ');
    imc = (peso/(altura^2));
    fprintf('IMC = %5.2f',imc);
    resp = input('\nRepetir o processamento (s/n)?','s');
end
```

2) Desenvolver um programa que lê dois números inteiros do teclado e depois imprime:

a) A média dos números;
b) O menor dos dois números;
c) A sequencia destes números do menor para o maior.



8 FUNÇÕES

8.1 Funções Científicas

O Matlab tem uma série de funções científicas pré-definidas. A maioria pode ser usada da mesma forma que seria escrita matematicamente. Por exemplo:

```
>> x=sqrt(2)/2
x =
0.7071

>> y=acos(x)
y =
0.7854

>> y_graus=y*180/pi
y_graus =
45.0000
```

Estes comandos calculam o arco cujo cosseno é √2/2, inicialmente em radianos, depois em graus. Abaixo segue uma lista de funções científicas disponíveis:

abs(x) - valor absoluto de x.
acos(x) - arco cujo cosseno é x.
asin(x) - arco cujo seno é x.
atan(x) - arco cuja tangente é x.
cos(x) - cosseno de x.
exp(x) - exponencial e^x.
gcd(x,y) - máximo divisor comum de x e y.
lcm(x,y) - mínimo múltiplo comum de x e y.
log(x) - logaritmo de x na base e.
log10(x) - logaritmo de x na base 10.
rem(x,y) - resto da divisão de x por y.
sin(x) - seno de x.
sqrt(x) - raiz quadrada de x.
tan(x) - tangente de x.

8.2 Outras Funções

Uma classe de comandos do MATLAB não trabalha com matrizes numéricas, mas com funções matemáticas. Esses comandos incluem:

· Integração numérica;

· Equações não-lineares e otimização;

· Solução de equações diferenciais.

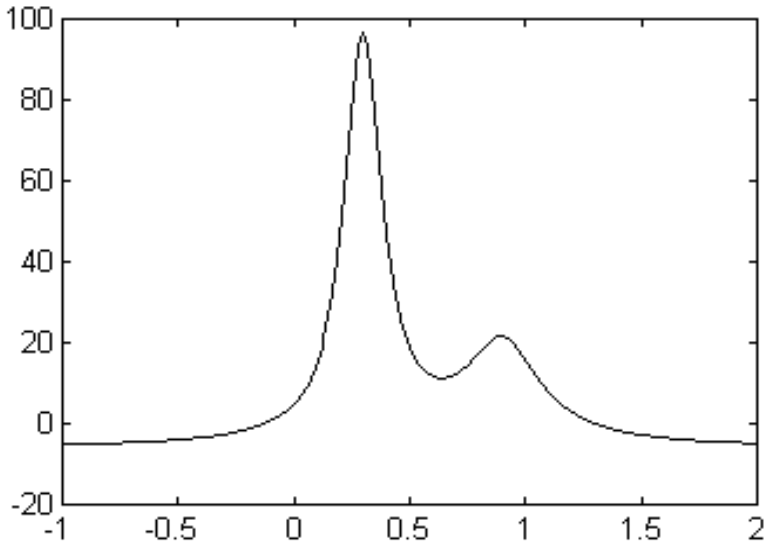
$$\text{humps}(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6$$

As funções matemáticas são representadas no MATLAB por arquivos ".m". Por exemplo, a função está disponível no MATLAB como um arquivo ".m" chamado **humps.m**:

```
function y = humps(x)
y = 1 ./ ((x-.3).^2 + .01) + 1./((x-.9).^2 + .04) - 6;
```

O gráfico da função é:

```
>> x = -1:0.01:2;
>> plot(x,humps(x))
```



8.2.1 Integração Numérica

A área abaixo da curva pode ser determinada através da integração numérica da função *humps(x)*, usando o processo chamado *quadratura*. Integrando a função *humps(x)* de -1 a 2:

```
>> q = quad ('humps',-1,2)
q =
        26.3450
```

Os dois comandos do MATLAB para integração usando quadratura são:

quad	Calcular integral numericamente, método para baixa ordem.
quad8	Calcular integral numericamente, método para alta ordem.

8.2.2 Equações Não-Lineares e Otimização

Os dois comandos para equações não-lineares e otimização incluem:

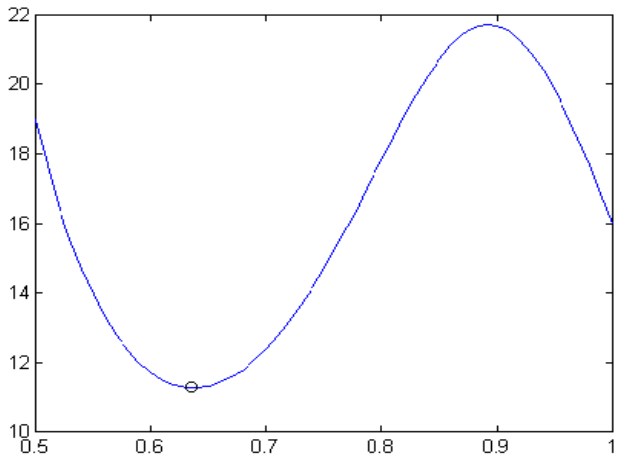
fmin	Minimizar função de uma variável.
fmins	Minimizar função de várias variáveis
fzero	Encontrar zero de função de uma variável.

Continuando o exemplo, a localização do mínimo da função *humps(x)* no intervalo de 0.5 a 1 é obtido da seguinte maneira,

```
>> xm = fmin('humps',0.5,1)
xm =
          0.6370
>> ym = humps(xm)
ym =
          11.2528
```

E o gráfico deste intervalo com o ponto de mínimo pode ser construído digitando:

```
>> x = 0.5:0.01:1
>> plot(x, humps(x), xm, ym, 'o')
```



Pode-se ver que a função *humps(x)* apresenta dois "zeros" no intercalo de -1 a 2. A localização do primeiro "zero" é próxima do ponto x = 0,

xzl = fzero('humps',0)

xzl =

-0.1316

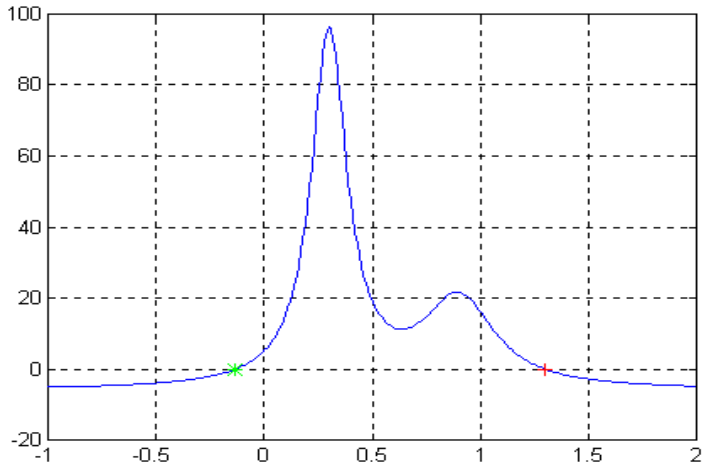
e a localização do segundo "zero" é próxima do ponto x= 1,

>> xz2=fzero('humps',1)
xz2 =

1.2995

O gráfico da função com os dois "zeros" é obtido através da expressão:

>> x = -1:0.01:2
>> plot(x, humps(x), xzl, humps(xzl),'', xz2, humps(xz2), '+'), grid*



8.2.3 Equações Diferenciais

Os comandos do MATLAB para resolver equações diferenciais ordinárias são:

ode23	Resolver equação diferencial. método baixa ordem.
ode23p	Resolver e plotar soluções.
ode45	Resolver equação diferencial. Método para alta ordem

Considere a equação diferencial de segunda ordem chamada de *Equação de Van der Pol*

$$\mathbf{x + (x^2 - 1) \, . \, x + x = 0}$$

Pode-se rescrever esta equação como um sistema acoplado de equações diferenciais de primeira ordem

$$x_1 = x_1 . (1-x_2^2) - x_2$$
$$x_2 = x_1$$

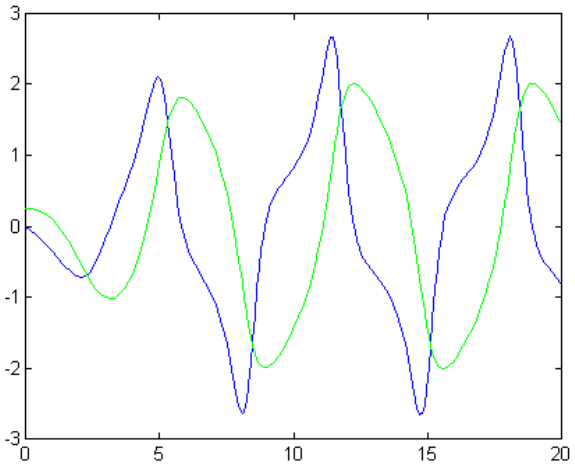
O primeiro passo para simular esse sistema é criar um arquivo ".m" contendo essas equações diferenciais. Por exemplo, o arquivo **volpol.m**:

```
function xdot=volpol(t,x)
xdot=[0 0]
xdot(l)=x(l).*(1- x(2).^2) - x(2);
xdot(2)=x(l);
```

Para simular a equação diferencial no intervalo 0 < t < 20, utiliza-se o comando **ode23**

```
>> t0 = 0; tf = 20;
>> x0 = [0 0.25];
>> [t,x] = ode23('volpol', t0, tf, x0);
>> plot(t,x)
```

e o resultado será o gráfico:



8.3 Resolvendo Equações Polinomiais

Achando as raízes de um polinômio. Por exemplo: $4x^2=0$ que tem duas raízes nulas.

```
>> p=[4 0 0]
p =
    4    0    0
>> p=[4 0 0];
>> r=roots(p)
r =
    0
    0
```

Achando as raízes de um polinômio. Por exemplo: $4x^2+5=0$ que não tem raízes reais.

```
>> p=[4 0 5];
>> r=roots(p)
r =
      0 + 1.1180i
      0 - 1.1180i
```

Achando as raízes de um polinômio. Por exemplo: $4x^2-12x=0$ que tem duas raízes reais: x'=3 e x"=0

```
>> p=[4 -12 0];
>> r=roots(p)
r =
    0
    3
```

Achando as raízes de um polinômio. Por exemplo: $x^4 - 12x^3 + 0x^2 + 25x + 116=0$

```
>> p=[1 -12 0 25 116];
>> r=roots(p)
r =
  11.7473
   2.7028
  -1.2251 + 1.4672i
  -1.2251 - 1.4672i
```

Construindo polinômio a partir de suas raízes: Considerando o vetor r da atividade anterior e pode-se observar que são restaurados os valores do polinômio utilizado.

```
>>pp = poly(r)
pp =
   1.0000  -12.0000   -0.0000   25.0000  116.0000
```

9 GRÁFICOS

A construção de gráficos no MATLAB é mais uma das facilidades do sistema. Através de comandos simples pode-se obter gráficos bidimensionais ou tridimensionais com qualquer tipo de escala e coordenada. Existe no MATLAB uma vasta biblioteca de comandos gráficos.

9.1 Gráficos Bidimensionais

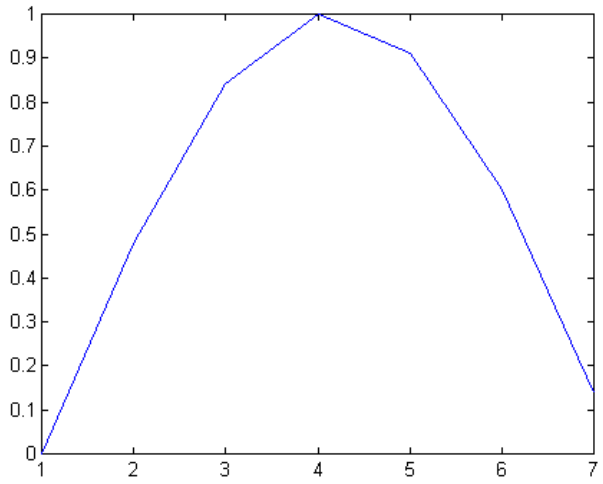
Estes são os comandos para plotar gráficos bidimensionais:

plot	Plotar linear.
loglog	Plotar em escala loglog.
semilogx	Plotar em semilog.
semilogy	Plotar em semilog.
fill	Desenhar polígono 2D.
polar	Plotar em coordenada polar.
bar	Gráfico de barras.
stem	Seqüência discreta.
stairs	Plotar em degrau.
errorbar	Plotar erro.
hist	Plotar histograma.
rose	Plotar histograma em ângulo.
compass	Plotar em forma de bússola.
feather	Plotar em forma de pena.
fplot	Plotar função.
comet	Plotar com trajetória de cometa.

Se **Y** é um vetor, **plot(Y)** produz um gráfico linear dos elementos de **Y** versos o índice dos elementos de **Y.** Por exemplo, para plotar os números [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0,14], entre com o vetor e execute o comando **plot:**

```
>> Y = [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0,14];
>> plot(Y)
```

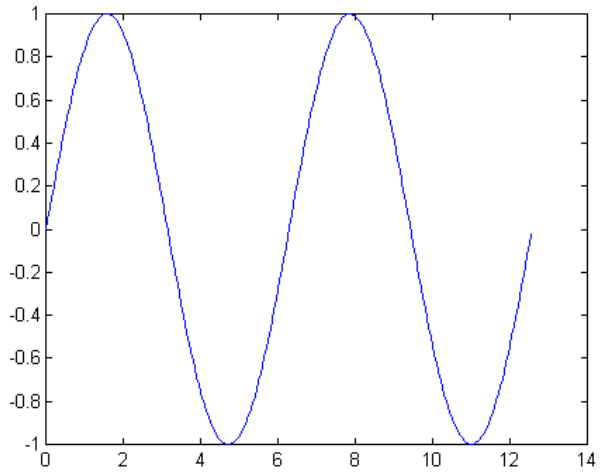
e o resultado é mostrado na Janela Gráfica:



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

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

Resulta em

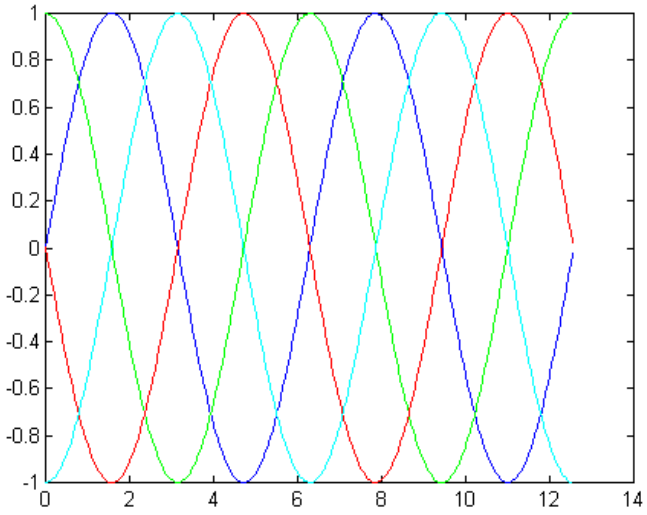


O MATLAB pode também plotar múltiplas linhas e apenas um gráfico. Existem duas maneiras, a primeira é usado apenas dois argumentos, como em **plot(X,Y)**, onde **X** e/ou **Y** são matrizes. Então:

- Se **Y** é uma matriz e **X** um vetor, **plot(X,Y)** plota sucessivamente as linhas ou colunas de **Y** versos o vetor **X**.
- Se **X** é uma matriz e **Y** é um vetor, **plot(X,Y)** plota sucessivamente as linhas ou colunas de **X** versos o vetor **Y**.
- Se **X** e Y são matrizes com mesma dimensão, **plot(X,Y)** plota sucessivamente as colunas de **X** versos as colunas de **Y**.
- Se **Y** é uma matriz, **plot(Y)** plota sucessivamente as colunas de **Y** versos o índice de cada elemento da linha de **Y**.

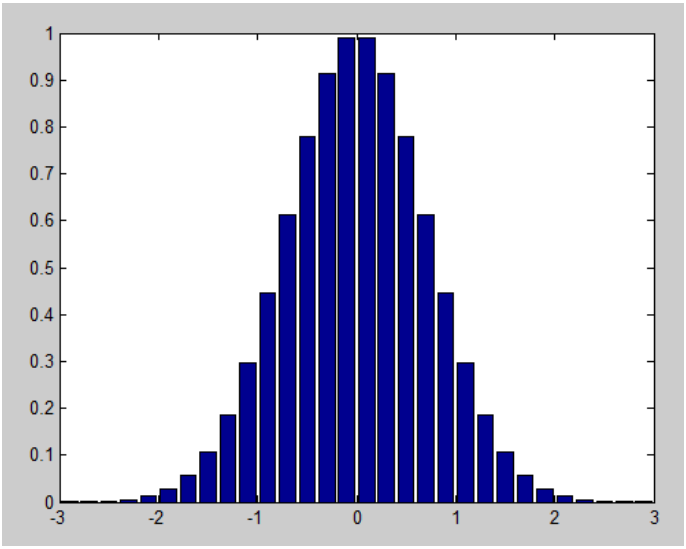
A segunda, e mais fácil, maneira de plotar gráficos com múltiplas linhas é usando o comando **plot** com múltiplos argumentos. Por exemplo:

>> plot(t, sin(t), t, cos(t), t, sin(t + pi), t, cos(t + pi))



Gerando um gráfico de barras:

>> x=-2.9:0.2:2.9;
*>> y=exp(-x.*x);*
>> bar(x,y)

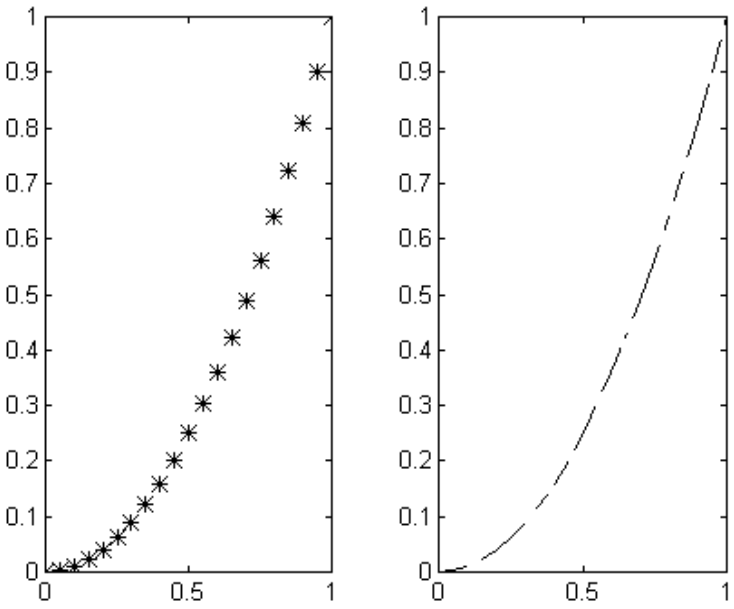


9.2 Estilos de Linha e Símbolo

Os tipos de linhas, símbolos e cores usados para plotar gráficos podem ser controlados se os padrões não são satisfatórios. Por exemplo:

>> X = 0:0.05:1;
>> subplot(l2l), plot(X,X.^2,'k')*
>> subplot(l22), plot(X,X.^2,'k --')

que resultam nos gráficos:



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

TIPO DE LINHA			TIPO DE PONTO	
_	_____		.	.....
--	----------		*	* * * * * * * *
-.	-.-.-.-.-.-.-.-.-.-.-.		o	o o oo o o o o o o
.	.....		+	++++++++++
			x	xx x x x x x x

CORES	
y	amarelo
m	lilás
c	azul claro
r	vermelho
g	verde
b	azul escuro
w	branco
k	preto

9.3 Números Complexos

Quando os argumentos para plotar são complexos, a parte imaginária é ignorada, exceto quando é dado simplesmente um argumento complexo. Para este caso especial é plotada a parte real versos a parte imaginária. Então, **plot(Z)**, quando **Z** é um vetor complexo, é equivalente a **plot(real(Z),imag(Z))**.

9.4 Escala Logarítmica, Coordenada Polar e Gráfico de Barras

O uso de **loglog**, **semilogx**, **semilogy** e **polar** é idêntico ao uso de **plot**. Estes comandos são usados para plotar gráficos em diferentes coordenadas e escalas:

- **polar(Theta,R)** plota em coordenadas polares o ângulo **THETA**, em radianos, versos o raio **R**;
- **loglog** plota usando a escala $\log_{10}$x$\log_{10}$;
- **semilogx** plota usando a escala semi-logarítmica. O eixo x é $\log_{10}$ e o eixo y é linear;
- **semilogy** plota usando a escala semi-logarítmica. O eixo x é linear e o eixo y é $\log_{10}$;

O comando **bar(X)** mostra um gráfico de barras dos elementos do vetor **X**, e não aceita múltiplos argumentos.

9.5 Plotando Gráficos Tridimensionais e Contornos

Estes são alguns comandos para plotar gráficos tridimensionais e contornos.

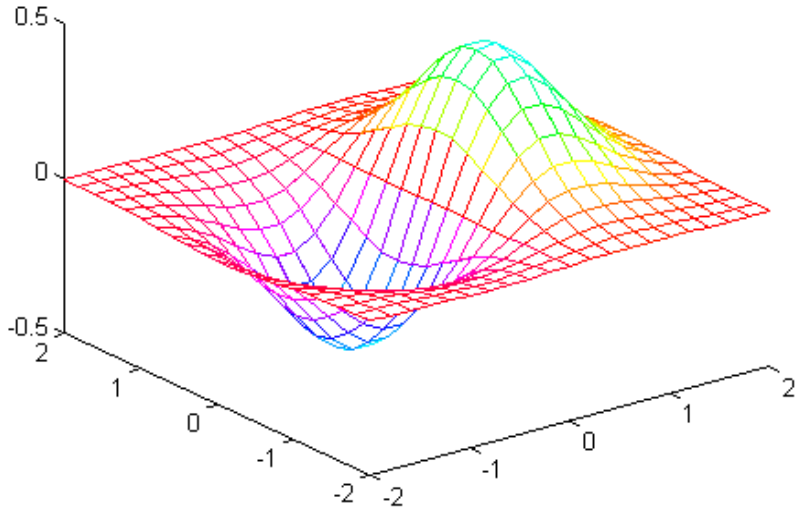
Plot3	Plotar em espaço 3D.
fill3	Desenhar polígono 3D.
comet3	Plotar em 3D com trajetória de cometa.
contour	Plotar contorno 2D.
contour3	Plotar contorno 3D.
clabel	Plotar contorno com valores.
quiver	Plotar gradiente.
mesh	Plotar malha 3D.
meshc	Combinação mesh/contour.
surf	Plotar superfície 3D.
surfc	Combinação surf/contour.

surfil	Plotar superfície 3D com iluminação.
slice	Plot visualização volumétrica.
cylinder	Gerar cilindro.
sphere	Gerar esfera.

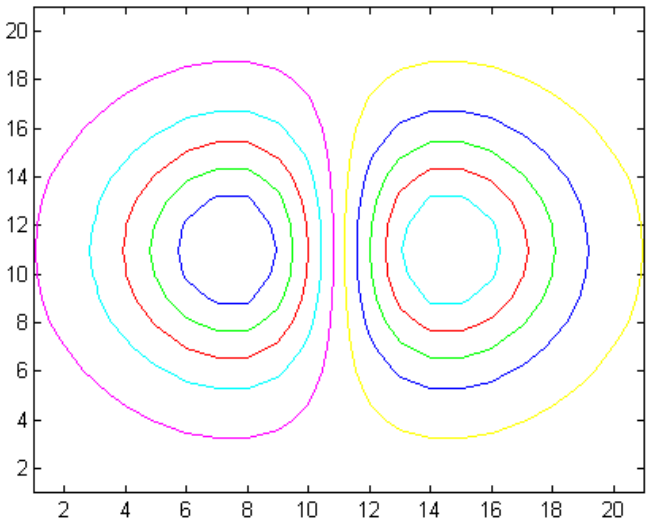
O comando **mesh(X,Y,Z)** cria uma perspectiva tridimensional plotando os elementos da matriz **Z** em relação ao plano definindo pelas matrizes **X** e **Y.** Por exemplo,

```
>> [X,Y] = meshdom(-2:.2:2, -2:.2:2);
>> Z = X.* exp(-X..^2 - Y.^2);
>> mesh(X,Y,Z)
```

que resulta no gráfico:



e o comando **contour(Z,10)** mostra a projeção da superfície acima no plano xy com 10 iso-linhas:



9.6 Anotações no Gráfico

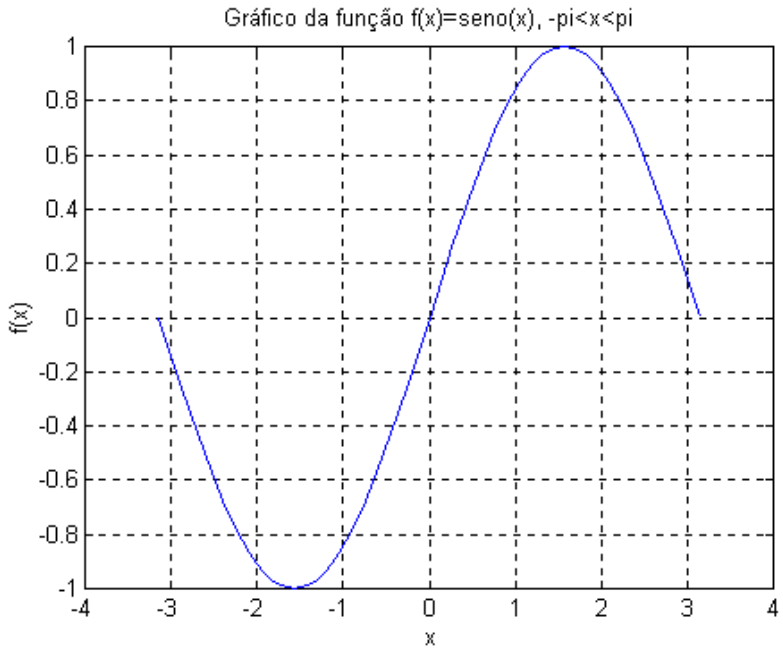
O MATLAB possui comandos de fácil utilização para adicionar informações em um gráfico:

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".
grid	Linhas de grade.

Por exemplo:

```
>> fplot('sin', [-pi pi])
>> title('Gráfico da função f(x)=seno(x), -pi<x<pi')
>> xlabel('x')
>> ylabel('f(x)')
>> grid
```

que resulta no gráfico:



9.7 Exemplos de Gráficos

1) Desenvolver um programa que lê três números inteiros do teclado e depois imprime o gráfico da expressão e solicita a informação se vai repetir a operação para novos números:

C:\matlabR12\work\fluxo_m.m

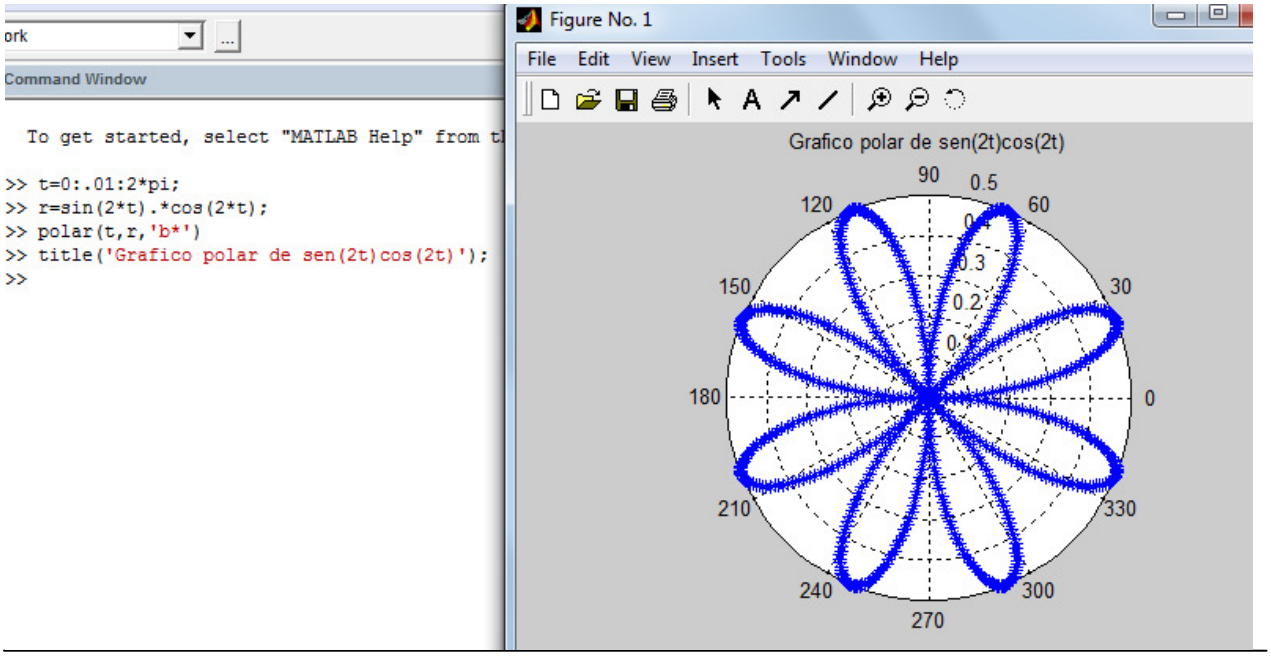
File Edit View Text Debug Breakpoints Web Window Help

```
%Plota uma função y=ax^2 + bx + c no intervalo -5<x<5
clear
clc
aux='s';
while aux=='s'
a=input('a = ');
b=input('b = ');
c=input('c = ');
x=-5:0.1:5;
y=a*x.^2+b*x+c;
plot(y)
figure(1)
pause
close
aux=input('Plotar outro ? (s/n) = => ','s');
end
```

Execução do programa:

>> fluxo_m
a =1
b =2
c =3
Plotar outro ? (s/n) = => n

2) Gerando um gráfico de coordenadas polares a partir do prompt de comando do MatLab..



10 OPERAÇÕES COM O DISCO

Os comandos **load** e **save** são usados, respectivamente, para importar dados do disco (rígido ou flexível) para a área de trabalho do MATLAB e exportar dados da área de trabalho para o disco. Outras operações com o disco podem ser efetuadas, como executar programas externos, trocar o diretório de trabalho, listagem do diretório, e serão detalhadas a seguir.

10.1 Manipulação do Disco

Os comandos **cd**, **dir**, **delete**, **type** e **what** do MATLAB são usados da mesma maneira que os comandos similares do sistema operacional.

cd	troca o diretório de trabalho atual
dir	lista o conteúdo do diretório atual
delete	exclui arquivo
type	mostra o conteúdo do arquivo texto
what	lista arquivos ".m", ".mat" e ".mex".

Para maiores detalhes sobre estes comandos utilize o **help**.

10.2 Executando Programas Externos

O caracter ponto de exclamação, **!**, é um desvio e indica que o restante da linha será um comando a ser executado pelo sistema operacional. Este procedimento vem sendo historicamente utilizado em todos as versões do MATLAB como "prompt" para indicar a execução de um colando do DOS, sendo muito útil nas versões que usavam somente o DOS. No ambiente Windows, entretanto, este comando é desnecessário, mas foi mantido nas versões do MATLAB para Windows.

Para entrar com o caracter de desvio no "prompt" do MATLAB, deve-se coloca-lo no Início do comando do DOS ou Windows que se deseja executar. Por exemplo, para carregar um aplicativo como o programa **Notepad** do Windows (Bloco de Notas), sem sair do MATLAB, entre com

>> ! Notepad

Uma nova janela é aberta, o Notepad é carregado, podendo ser utilizado da maneira usual.

Pode-se usar, também, qualquer comando implícito do DOS, por exemplo: copy, fomat, ren, mkdjr, rmdir, ...

10.3 Importando e Exportando Dados

Os dados contidos na Área de Trabalho do MATLAB podem ser armazenados em arquivos, no formato texto ou binário, utilizando o comando **save**. Existem diversas maneiras de utilizar este comando. Por exemplo. para armazenar as variáveis X, Y e Z pode-se fazer:

save	salva os dados no arquivos binário "matlab.mat".
save X	salva a matriz X no arquivo o binário "x.mat".
save arql X Y Z	salva as matrizes X, Y e Z no arquivo binário "arq1.mat".
save arq2.sai X Y Z -ascii	salva as matrizes X., Y e Z no arquivo texto "arq2.sai" com 8 dígitos.
Save arq3.sai X Y Z -ascii -double	salva as matrizes X., Y e Z no arquivo texto "arq3.sai" com 16 dígitos.

Os dados obtidos por outros programas podem ser importados pelo MATLAB, desde que estes dados sejam gravados em disco no 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 em linguagem C, depois de executado, monta o arquivo "teste.sai" (mostrado abaixo) que contém uma matriz.

1.0000 2.0000 3.0000
4.0000 5.0000 6.0000
7.0000 8.0000 9.0000

Executando o comando:

>> load teste.sai

o MATLAB importa a matriz, que passa a se chamar **teste**:

>> teste

teste =
1 2 3
4 5 6
7 8 9

Obviamente, o MATLAB 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:

save	load
save X	load x

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

Deve-se ressaltar que o comando **save**, quando usado para exportar os dados do MATLAB em formato texto, exporta apenas um bloco contendo todas as variáveis. E quando importamos estes comandos através do comando **load**, apenas uma variável com nome do arquivo é importada. Por 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

>> load arq2.sai

>> arq2

Resultado:

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*

11 LISTA DE EXERCÍCIOS

1 -Calcule a raiz da equação $f(x)=x^3-9x+3$ pelo Método da Bissecação (ref.[6] pág.34) no intervalo **I**=[0,1] com $\varepsilon=10^{-3}$ e número máximo igual a 15.

2 -Calcule as raízes da equação $f(x) =x^3-9x+3$ pelo Método de Newton-Raphson (ref.[6] pág.57) nos intervalos $\mathbf{I_1}$=(-4,-3), $\mathbf{I_2}$=(0,1) e $\mathbf{I_3}$=(2,3) com $\varepsilon=10^{-3}$ e com número máximo de iterações igual a a 10.

3 -Resolva o sistema linear abaixo usando o Método de Eliminação de Gauss (ref.[6] pág.96) e compare com o resultado obtido pelo MATLAB.

$$\begin{cases} 17x1 + 24\,x2 + x3 + 8\,x4 + 15\,x5 = 175 \\ 33x1 + 5\,x2 + 7\,x3 + 14\,x4 + 16\,x5 = 190 \\ 4x1 + 6\,x2 + 13\,x3 + 20\,x4 + 22\,x5 = 245 \\ 10x1 + 12\,x2 + 19\,x3 + 21\,x4 + 3\,x5 = 190 \\ 11x1 + 18\,x2 + 25\,x3 + 2\,x4 + 9\,x5 = 175 \end{cases}$$

4 -Usando Fatoração LU (ref.[6] pag.108) resolva o sistema linear mostrado no exercício 3 e compare com os resultados obtidos pelo MATLAB (comando **lu**).

5 -Uma grande placa de 300 mm de espessura (k=37,25 kcal / m ºC) contém fontes de calor uniformemente distribuídas (q= 9x10^5 kcal / h m^3). A temperatura numa face é 1000ºC e calor é transferido para essa superfície, $q(0)$, a 2500 kcal/hm^2. Escreva um programa para determinar a distribuição de temperatura em regime permanente na placa (ref.[7] pag.37) plotando os resultados. A placa deve ser dividida em fatias iguais e deve ser um dado de entrada do programa.

6 -A distribuição de temperatura ao longo de uma aleta em forma de piano circular é dada pela equação

$$\frac{T - T_\infty}{T_S - T_\infty} = \frac{\cosh(m()L\text{-}x) + (h_L/mk)\,\mathrm{senh}(m(L\text{-}x))}{\cosh(mL) + (h_L/mk)\,\mathrm{senh}(mL))}$$

Escreva um programa para determinar a distribuição de temperatura ao longo da aleta (ref.[7] pag.46) plotando os resultados.

Dados:

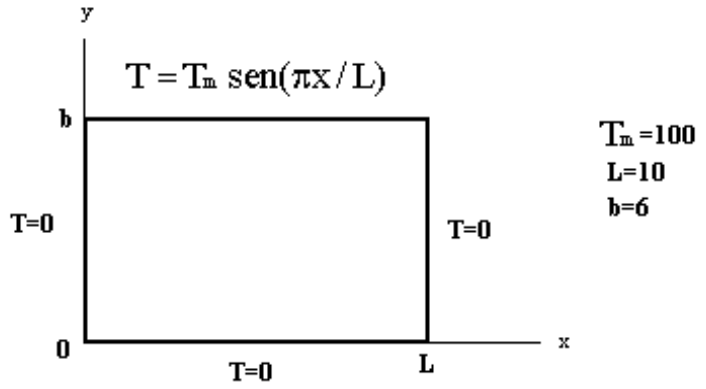
m = 1 L=10
k = 1 T_1= 100
h_L = 2 T8 = 25

7 -A distribuição de temperatura ao longo da placa mostrada na figura é dada pela equação (ref.[7] pag.67):

$$T(x., y) = TM\,\frac{\mathrm{senh}(p\ y/L)}{\mathrm{senh}(p\ b/L)}\,\mathrm{sen}(p\ x/L)$$

Plote a distribuições de temperatura ao longo da placa (comando **mesh**), mostre-as as isotermas (comando **contour**) e o gradiente de temperatura na placa (comandos **gradient** e

quiver).



8 -Uma chaminé de tijolos (ref.[7] pag.57) de 60 m de altura, com um diâmetro de 1,80 m, tem uma camada de tijolos refratários (k = 0,9 kcal / h m ºC) de 110 mm de espessura e uma parede externa de tijolos de alvenaria (k = 0.5 kcal / h m ºC) que varia linearmente de uma espessura de 600 mm na base até uma espessura de 200 mm no topo. O coeficiente de transmissão de calor entre o gás da chaminé e a parede é 50 kcal / h m^2 ºC, e entre a parede externa e o ar é 15 kcal / h m^2 C. Se o gás da chaminé está a 300 ºC e o ar está a 4 ºC, calcule numericamente a perda de calor da chaminé dividindo-a em pedaços que representem um anel circular com raio crescente. Calcule a resistência térmica total e plote os resultados para intervalos que vão de 1 a 20.

% LISTA DE EXERCÍCIOS - COMANDOS BÁSICOS DO MATLAB

% EXECUTE OS SEGUINTES COMANDOS E INTERPRETE OS RESULTADOS

```
a = 2500/20
a = 2500/20;
b = [1 2 3 4 5 6 7 8 9]
c = [1 2 3 ; 4 5 6 ; 7 8 9]
c = [c ; [10 11 12]
c(2,2) = 0
l = length(b)
[m,n] = size(b)
[m,n] = size(c)
who
whos
clear
who
b = l + 2 + 3 + 4 + ...
5 + 6 - 7
x = 1 : 2 : 9
x = (0.8 : 0.2 : 1.4);
y = sin(x)
help sin
dir
```

```
a = 2^3
a = 4/3
format long
a = 4/3
format short
clear
a=[1 2 3 ; 4 5 6 ; 7 8 9];
b = a'
c = a + b
c = a - b
a(l,:) = [-1 -2 -3]
c = a(:,2)
c = a(2:3, 2:3)
x = [- 1 0 2];
y = [-2 -1 1]';
x*y
c = x + 2
a = [1 0 2; 0 3 4 ; 5 6 0];
size(a)
b = inv(a);
c = b*a
c = b/a
c = b\a
clear a b x y
whos
```

```
% A instrução seguinte abre o arquivo notas.dry e grava todas as instruções
% digitadas na seqüência
diary notas.dry
x = [1 -2 3]
y = [4 3 2]
z = x.*y
z = x.^y
y.^2
diary off % Encerra a gravação da instrução diary em notas.dry
dir
type notas.dry
clear
help diary
help sqrt

% Trabalhando com números complexos
i = sqrt(-1)
a = [1 2;3 4] + i*[5 6;7 8]
realz = real(z)
imagz = imag(z)
modz = abs(z)
fasez = angle (z)
```

```
% Multiplicação de polinômios
% x3 = (x^2 + 3x + 2).(x^2 - 2x + 1)
x3 = conv([1 2 3],[1 -2 1]) % Como ele faz isto?



% Determinação das raízes de um polinômio
roots([1 3 2])
roots([1 -2 1])
roots(x3)

% Utilitários para matrizes
a = eye(4)
a = rand(5)
help rand
b = [2 0 0;0 3 0;0 0 -1];
d = det(b)
l = eig(b)
help det
help eig
clear
```

% RECURSOS DE GRAVAÇÃO (ARMAZENAGEM) DE DADOS

```
help save
help load
a = [1 2 3 4 5 6 7 8];
b = a*2;
c = a - 1;
save arquivo 1 a b c
dir
clear
whos
load arquivo 1
whos
% Em que arquivo estão gravados os vetores a, b e c?
clear
```

% RECURSOS GRÁFICOS

```
y = [0 2 5 4 1 0];
plot(y)
help pi
t = 0:.4:4*pi
```

```
y = sin(t)
z = cos(t);
plot(t, y, '.', t, z "-.")
title('Funções')
xlabel("t")
ylabel("Seno e Cosseno")
text(3, 0.5, 'Seno')
% Após o próximo comando, selecione a posição que deseja colocar o texto 'Cosseno' com
% o mouse
gtext('Cosseno')
```

% AJUSTE DE CURVAS DE DADOS EXPERIMENTAIS

```
t = (-1:.1:1);
x = t.^2;
xr = x+0.2(rand(size(x))-.5);
figure(1); plot(t, xr, 'g*')
p = polyfit(t, xr, 2)
xa = po1yval(p, t);
figure(l); plot(t, xr, 'g*', t, xa)
% Após a próxima instrução, clique em dois pontos do gráfico, e os valores
% das coordenadas serão retornados em [x,y]
[x, y] = ginput(2)
```

% PROGRAMANDO COM O MATLAB

```
% Abra um arquivo a partir do Matlab (File, New, M-File)
% e você estará trabalhando no Bloco de Notas (Notepad) do Windows.
% Digite os seguintes comandos e grave o arquivo com o nome
% testel.m, no diretório de usuários (alunos).
n = 3 ;
m = 3;
for i = 1: m
   for j= 1 : n
      a(i, j) = i + j;
   end;
end
disp('Matriz A')
disp(a)
%final do programa testel.m
```

% CRIANDO UMA SUBROTINA

```
% Abra outro arquivo, salvando-o com nome de teste2.m
% Digite os seguintes comandos neste arquivo
v = 1:1:10;
```

```
m = media(v);
s = sprintf('\n A média é: %4.2f', m);
disp(s);
% final do programa teste2.m
```

Agora crie o seguinte arquivo, com o nome de media.m

```
function x = media(u)
% function x = media(u) calcula a média do vetor u, colocando o resultado em x
x = sum(u)/length(u);
% final da subrotina media.m
% Na linha de comando do Matlab, digite:
teste2
echo on
teste2
echo off
```

% CRIANDO UM PROGRAMA EXEMPLO DE GRÁFICO 3D

```
% Abra outro arquivo, salvando-o com nome de teste3.m
% Digite os seguintes comandos neste arquivo
clear
n = 30;
m = 30;
for i = 1:m
   for j = 1:n
      a(i,j) = sqrt(i+j);
   end
end
b = [a+0.5 a'-0.5;
(a.^2)/5 ((a'-0.1).^2)/2];
mesh(b)
```

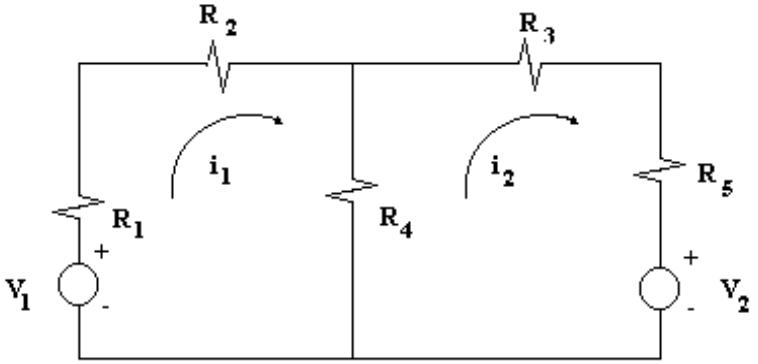
% EXERCÍCIOS COM O MATLAB

Exercício 1 - Faça um programa que desenhe unia pirâmide 3D. Utilize o mesh().

Exercício 2 - Copie o gráfico de uma senóide para um arquivo texto do "WORD". Siga os seguintes passos: 1º após ter gerado o gráfico, faça **print -dmeta** (no MATLAB); 2º Pressione ALT-TAB até entrar no "WORD" ou então abra o '"WORD"; 3º Posicione o cursor no local do texto onde o gráfico deva entrar; 4º Digite Ctrl-V; 5º Ajuste a escala vertical do gráfico com o editor de gráficos do "WORD".

Exercício 3 - Repita o exercício 2 com **-dbitmap** no lugar de **-dmeta** e compare o tamanho (em Kb) dos dois arquivos texto.

Exercício 4 - Resolva o circuito dado na figura abaixo (encontre i_1 e i_2) utilizando a inversão de matrizes do MATLAB. Faça um programa para isto. Adote $R_1 = 5W$, $R_2 = 10W$, $R_3 = 5W$, $R_4 = l5W$, $R_5 = 20W$, $V_1 = 10,0$ V, $V_2 = 20,0$ V.



Resp.: $i_1 = 0,01026$ A e $i_2 = 0,4615$ A.

Exercício 5 - Supondo que a fonte V_2 esteja em curto, ou seja, $V_2 = 0,0$ V, quais os valores de i_1 e i_2 ?
Resp.: $i_1 = 0,4103$ A e $i_2 = -0,1538$ A.

Exercício 6 - Gere um vetor com N elementos aleatórios. Escreva uma função que tenha como entrada o vetor, e retome o índice e o valor do maior elemento do vetor, utilizando o comando if.

Exercício 7 - Escreva um programa (utilizando o comando while) que aceite entradas numéricas pelo teclado. Os valores devem ser números entre 0 e 5, e caso o usuário digite algum valor fora deste intervalo, o programa é encerrado.

Exercício 8 - Em uma sala estão 8 pessoas, reunidas em uma mesa circular. Cada uma escolhe um número aleatoriamente e pega o seu número e soma com os números das pessoas ao lado, a sua esquerda e direita. Passa-se as 8 somas para você, que estava fora da reunião. Como você descobre o número que cada pessoa escolheu ? Utilize o MATLAB.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MATLAB for Windows User's Guide, The Math Works Inc., 1991.

[2] DONGARRA J.J., MOLER C.B., BUNCH, J.R, STEWART, G.W., LINPACK User's Guide, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[3] SMITH, B.T., BOYLE, J.M., DONGARRA, J.J., GARBOW, B.S., IKEBE, Y., KLEMA, V.C., MOLER, C.B., Matriz Eigensystem Routines - EISPACK Guide, Lecture Notes in Computer Science, volume 6, second edition, Springer-Verlag, 1976.

[4] GARBOW, B.S., BOYLE, J.M., DONGARRA, J.J., MOLER, C.B., Matriz Eigensystem Roulines EISPACK Gide Extension, Lecture Notes in Computer Science, volume 51, Springer-Verlag, 1977.

[5] GOLUB, G.H., VAN LOAN, C.F., Matriz Computations, Johns Hopkins University Press, 1983.

[6] RUGGIERO, M.A.G., LOPES, V.L.R., Cálculo Numéricos - Aspectos Teóricos e Computacionais, Ed. MacGraw-HiII, São Paulo, 1988.

[7] KREITH, F., Princípios da Transmissão de Calor, Ed. Edgard Blücher Ltda., São Paulo, 1977.

[8] Curso de MATLAB for Windows, Departamento de Engenharia Mecânica, UNESP, Campus de Ilha Solteira.

Atenção:

Esta apostila foi desenvolvida a partir de um arquivo disponibilizado na internet pelo grupo PET da Engenharia Elétrica da UFMS. E também de um conjunto de slides da professora Patrícia Jaques, aos quais agradeço a disponibilização na internet e que propiciou a produção deste material.