UNIVERSIDADE DA BEIRA INTERIOR

DEPARTAMENTO DE ENGENHARIA ELECTROMECÂNICA

CONTROLO DE SISTEMAS

APONTAMENTOS DE MATLAB

Introdução ao MATLAB

Pedro Dinis Gaspar
António Espírito Santo
J. A. M. Felippe de Souza

Edição Abril 2002

ÍNDICE

1. - APRESENTAÇÃO

Em variados meios industriais e académicos utiliza-se o MATLAB por constituir um *software* interactivo de alta performance direccionado para o cálculo numérico. O MATLAB permite a realização de aplicações ao nível da análise numérica, de análise de dados, cálculo matricial, processamento de sinais e construção de gráficos, entre outras, abordando um banda larga de problemas científicos e de engenharia.

Ao possuir um ambiente de trabalho na óptica do utilizador, prima pela facilidade de utilização, além de possuir uma estrutura em que os problemas e soluções são expressos somente como são escritos matematicamente, ao contrário da programação tradicional.

O MATLAB é um sistema interactivo cujo elemento básico de informação é uma matriz que não requer dimensionamento. Este sistema permite a resolução de problemas numéricos em apenas uma fracção do tempo que se gastaria para escrever um programa semelhante numa linguagem de programação clássica.

1.1. OBJECTIVO

Introduzir comandos básicos do MATLAB para permitir um rápido acesso às potencialidades do ambiente. O utilizador iniciante poderá dispor de uma referência rápida para algumas possibilidades de uso do MATLAB.

1.2. OBSERVAÇÃO

O documento como está, não tenciona cobrir todos os tópicos do MATLAB, porque o ambiente, devido à sua arquitectura formada por caixas de ferramentas (toolboxes), possui manuais específicos para cada uma destas, tornando impraticável a tentativa de abranger todos os tópicos em um único documento referência.

Este trabalho baseia-se em documentos públicos disponíveis na Internet, nos manuais do MATLAB, no livro *MATLAB – Student Edition* e nas informações disponíveis na página da internet da companhia Mathworks.

1.3. AMBIENTE DE TRABALHO

Quando o MATLAB é inicializado, duas janelas são exibidas: a *Janela de Comando* (Command Windows) e *Janela Gráfica* (Graphic Windows). A *Janela de Comando* é activada quando se inicia o MATLAB, e o "prompt" padrão (>>) é exibido na tela. A partir deste ponto, o MATLAB espera as instruções do utilizador. Para introduzir uma pequena matriz, por exemplo utiliza-se :

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

Colocando parêntesis rectos em torno 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, utiliza-se :

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

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. No caso de pretender efectuar o cálculo de :

$$\log\left(\sqrt{\tan\left(\frac{\pi}{5}\right)}\right)$$

Introduzir na tela:

```
>> 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 function or variable sqt.
```

O comando com a resposta apropriada seria:

```
>> log (sqrt(tan(pi/5)))
     ans =
               -0.1597
```

NOTA : Todas as funções que façam uso de um argumento em ângulo, são calculadas com o dito argumento expresso em radianos.

2. - INTRODUÇÃO

O MATLAB trabalha essencialmente com um tipo de objecto, uma matriz numérica rectangular podendo conter elementos complexos (Relembrar que um escalar é uma matriz de dimensão *1x1* e que um vector é uma matriz que possui somente uma linha ou uma coluna).

2.1. INTRODUÇÃO DE MATRIZES SIMPLES

As matrizes podem ser introduzidas no MATLAB de diferentes modos:

- Introduzida na *Janela de Comando* (lista explícita de elementos),
- Geradas por comandos e funções,
- Criadas em ficheiros ".m",
- Carregadas a partir de um ficheiro de dados externo.

O método mais fácil de introduzir pequenas matrizes no MATLAB é utilizando 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 parêntesis rectos em volta do grupo de elementos que formam a matriz com o objectivo de a limitar. Por exemplo, introduzindo a expressão :

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

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

A matriz ***A*** é guardada 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 introduzir matrizes no MATLAB é através de um ficheiro no formato texto com extensão "**.m**". Por exemplo, se um ficheiro 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 ficheiro e introduz a matriz ***A***.

```
>>gera
```

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

Exemplos de funções utilizadas para gerar matrizes: ***rand***, ***magic***, ***hilb***.

rand(n) gera matriz quadrada de dimensões *n x n*, aleatória com distribuição uniforme, no intervalo **[0,1]**.

```
>>x=rand(3)
```

rand(m,n) gera uma matriz aleatória de dimensões *m x n*.

```
>>x=rand(3,4)
```

magic(n) cria uma matriz quadrada integral de dimensões *n x n* (linhas e colunas têm a mesma soma).

```
>>x=magic(4)
```

NOTA : Em todos os exemplos, ***m*** e ***n*** são inteiros positivos.
Para especificar um único elemento da matriz utilizam-se os índices referentes ao número de linha e número de coluna entre parêntesis da matriz onde se encontra o elemento : ***A(n,m)***.
Outra forma de gerar matrizes é através de ciclos ***for***.

2.2. ELEMENTOS DE MATRIZES

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

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

    x =
    -1.3000        1.4142       23.0400
```

Um elemento individual da matriz pode ser referenciado com índice entre parêntesis. Continuando o exemplo,

```
>> x(6) = abs(x(1))

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

Note-se que a dimensão do vector ***x*** é aumentada automaticamente de *1x3* para *1x6* de modo a acomodar o novo elemento e que os elementos indefinidos do intervalo são estabelecidos como zero.

Matrizes de maior dimensão podem ser construídas a partir de pequenas matrizes. Por exemplo, pode-se anexar outra linha à matriz ***A*** usando:

```
>> r= [ 10 11 12];
>> A= [A;r]

    A =
         1     2     3
         4     5     6
         7     8     9
        10    11    12
```

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

Podem ser extraídas matrizes pequenas da matriz original utilizando "**;**". Por exemplo,

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

Selecciona as três primeiras linhas e todas as colunas da matriz ***A*** actual, modificando-a da sua forma original.

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, introduzindo a expressão:

>> 1900/81

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 ficheiros com extensão "**.m**" e em situações onde o resultado é uma matriz de grandes dimensões e há 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 = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7...
>> - 1/8 + 1/9 - 1/10 + 1/11 - 1/12 + 1/13;

Calcula o resultado da série, atribuindo o somatório 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.

NOTA : Para que o MATLAB não faça nenhuma distinção entre as letras minúsculas e maiúsculas utiliza-se a função ***casesen***.

2.4. INFORMAÇÕES SOBRE A 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 nos exemplos, incluindo ***ans***.

Uma informação mais detalhada que indica a dimensão de cada uma das variáveis correntes é obtida com ***whos***, que para o 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 a matriz ***A*** de dimensão *3x3* faz uso de 72 bytes e todas as variáveis utilizadas um total de 160 bytes.

2.5. NÚMEROS E EXPRESSÕES ARITMÉTICAS

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

3	*-99*	*0.00001*
9.637458638	*1.602E-20*	*6.06375e23*

As expressões podem ser construídas através dos operadores aritméticos usuais e das regras de precedência:

1	^	Potenciação
2	/	Divisão a direita
2	\	Divisão a esquerda
3	*	Multiplicação
4	+	Adição
4	-	Subtracção

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.

São usados parêntesis na sua forma padrão para alterar a precedência usual dos operadores aritméticos.

2.6. NÚMEROS E MATRIZES COMPLEXAS

Os Números Complexos são permitidos em todas operações e funções no MATLAB. Os números complexos são introduzidos utilizando as funções especiais ***i*** ou ***j*** que correspondem a parte imaginária de um número complexo. Por exemplo:

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

Ou,

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

Sabendo que um número complexo é constituído por uma parte real {Re} e por uma parte imaginária {Im}, tal que:

$$z = \{Re\} + \{Im\}*i$$

Pode ser expresso na forma:

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

Em que :

```
>> r = abs(z)
>> theta = angle(z)*180/pi
```

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

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

Ou,

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

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. FORMATAÇÃO DE SAÍDA

O formato numérico exibido na tela pode ser modificado utilizando o comando ***format***, que afecta somente o modo como as matrizes são mostradas, e não como elas são calculadas ou guardadas (o MATLAB efectua todas as operações em dupla precisão).

Se todos os elementos da matriz são inteiros exactos, a matriz é mostrada num formato sem qualquer ponto decimal. Por exemplo,

```
>> x = [-1 0 1]

    x =
         -1     0     1
```

Se pelo menos um dos elementos da matriz não é inteiro exacto, existem várias possibilidades de formatar a saída. O formato por defeito, 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, 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 superior a 1000 ou inferior a 0.001, é aplicado um factor de escala comum para que a matriz completa seja mostrada. Por exemplo,

```
>> X = 1e20*x

    X =    1.0e+20    *
           1.3333     0.0000
```

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

2.8. UTILIZAÇÃO DO HELP (Ajuda)

O MATLAB possui um comando de ajuda: ***help*** que fornece informações sobre a maior parte dos tópicos. Introduzindo:

```
>> help
```

Obtêm-se uma lista desses tópicos disponíveis Para obter informações sobre um tópico específico, referir ***help tópico***. Por exemplo,

```
>> help plotxy
```

Fornece uma lista de todos os comandos relacionados com gráficos bidimensionais.

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

```
>> 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-se que no exemplo mostrado para adicionar o título a um gráfico, ***TITLE ('TEXT')*** está escrito em letras maiúsculas somente para destacar. Recordar que todos os comandos do MATLAB devem ser escritos em letras minúsculas, portanto, para adicionar o texto "***Título do Gráfico***" a um gráfico, deve-se introduzir:

>> title ('Título do Gráfico')

2.9. FUNÇÕES

As potencialidades do MATLAB residem no seu extenso conjunto de funções. O MATLAB possui um grande número de funções intrínsecas que não podem ser alteradas pelo utilizador. Outras funções estão disponíveis numa biblioteca externa distribuídas com o programa original (MATLAB TOOLBOX), que são na realidade ficheiros com a extensão "**.m**" criados a partir das funções intrínsecas. As categorias gerais de funções matemáticas disponíveis no MATLAB incluem:

- Matemática elementar;
- Funções especiais;
- Matrizes elementares;
- Matrizes especiais;
- Decomposição e factorização de matrizes;
- Análise de dados;
- Polinómios;
- Solução de equações diferenciais;
- Equações não-lineares e optimização;
- Integração numérica;
- Processamento de sinais;
- Entre outras ...

As secções subsequentes mostram mais detalhadamente as diferentes categorias de funções.

Qualquer informação adicional sobre os conteúdos das diversas categorias poderá ser encontrado no respectivo ***help***.

3. - OPERADORES ARITMÉTICOS + - * / \ ^ ‘

O MATLAB possui dois tipos diferentes de operações aritméticas. As operações aritméticas matriciais são definidas pelas regras da Álgebra Linear. As operações aritméticas com arrays (conjuntos) são efectuadas elemento por elemento. O caractere de ponto decimal “**.**” distingue as operações matriciais das operações com arrays. No entanto, como as operações matriciais e com arrays são iguais para a Soma e para a Subtracção, o par de caracteres “**.+**” e “**.-**“ não são usados.

3.1. OPERAÇÕES COM MATRIZES

As operações com matrizes no MATLAB são as seguintes:

- Transposta;
- Adição;
- Subtracção;
- Multiplicação;
- Divisão a direita;
- Divisão a esquerda;
- Potenciação;

A seguir cada uma destas operações é apresentada com maior detalhe.

3.1.1. Transposta

O caractere apóstrofo, " **'** " , indica a transposta de uma matriz.

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

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

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

E,

```
>> x = [-1 0 2]'
```

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

Se ***Z*** é uma matriz complexa, ***Z'*** será o complexo conjugado 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.'

   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
```

3.1.2. Adição e Subtracção

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

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

É aceitável, e o resultado da soma será:

```
   C =
                2     6    10
                6    10    14
               10    14    18
```

A adição e subtracção também são definidas se um dos operadores for um escalar, ou seja, uma matriz *1x1*. Neste caso, o escalar é adicionado ou subtraído de todos os elementos do outro operador. Por exemplo:

```
>> y = x - 1

   y =
               -2
               -1
                1
```

3.1.3. Multiplicação

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
   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     1    -1
     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 vector é um caso especial do produto entre matrizes. Por exemplo ***A*** e ***x***,

```
>> b = A*x

b =
     5
     8
    11
```

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

```
>> pi*x

ans =
   -3.1416
         0
    6.2832
```

Se os tamanhos das matrizes são incompatíveis para a operação matricial, será gerada uma mensagem de erro, com excepção do caso de operações entre escalares e matrizes (para adição, subtracção, divisão e multiplicação).

3.1.4. Divisão

A divisão de matrizes requer especial atenção, pois existem dois símbolos para divisão de matrizes no MATLAB "\" e "/". Se ***A*** é uma matriz inversível quadrada e ***b*** é um vector coluna (ou linha) compatível., 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)*,** mas o resultado é obtido directamente:

X = A\b é a solução de *A*X = b*
X = b/A é a solução de *X*A = b*

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

```
>> z = A\b

z =
    -1
     0
     2
```

Na "divisão à esquerda", se ***A*** é quadrada, então é factorizada através da eliminação Gaussiana e estes factores são usados para resolver ***A*x = b***. Se ***A*** não for quadrada, é factorizada através da ortogonalização de Householder com pivoteamento por coluna e os factores são usados para resolver o sistema sub ou sobre determinado no sentido dos mínimos quadrados. A divisão à direita é definida em termos da divisão à esquerda por ***b/A = (A'\b')***.

3.1.5. Potenciação

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 potenciação é calculada como múltiplas multiplicações. Por exemplo,

```
>> A^3

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

Se ***P*** é uma matriz e ***a*** é um escalar, ***a^P*** calcula o escalar elevado à matriz ***P*** fazendo uso dos Valores e Vectores Próprios.

NOTA : X^P, sendo ***X*** e ***P*** matrizes, apresenta erro.

3.2. OPERAÇÕES COM CONJUNTOS (ARRAYS)

O termo *operações com conjuntos* é utilizado 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 efectuadas como as operações usuais, utilizando-se os mesmos caracteres ("*****", "/", "\", "^" e " ‘ ") precedidos por um ponto "**.**" ("**.***", "**./**", "**.**", ".^" e " **.‘** ").

3.2.1. Adição e Subtracção

Para a adição e a subtracção, as operações com conjuntos e as operações com matrizes são iguais. Deste modo os caracteres "+" e "**-**" são empregues do mesmo modo e considerando as mesmas restrições de utilização.

3.2.2. 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];
>> z = x .* y

    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

    z =
                4.0000    2.5000    2.0000
```

3.2.3. Potenciação

A potenciação de conjuntos é indicada por "**.**^". A seguir são mostrados alguns exemplos utilizando os vectores ***x*** e ***y***. A expressão:

```
>> z = x .^ y

    z =
                1    32   729
```

A potenciação pode usar um escalar.

```
>> z = x.^2

    z =
                1     4     9
```

Ou, a base pode ser um escalar.

```
>> z = 2.^[x y]

    z =
                2     4     8    16    32    64
```

4. - OPERADORES RELACIONAIS < <= >= == ~=

Os operadores usados para comparação de duas matrizes com as mesmas dimensões são expressos por:

<	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 de números um (*1*) e zero (*0*), representando respectivamente **VERDADEIRO** e **FALSO**. Por exemplo**,**

```
>> 2 + 2 ~= 4

     ans =
                0
```

Os operadores "<", "<=" , ">", e ">=" apenas usam a parte real dos operandos para comparação. Os operadores "==" e "~=" testam tanto a parte real como a imaginária.

Estes operadores têm precedência entre os operadores lógicos e aritméticos.

Outros exemplos relativos ao uso dos operadores relacionais reflectem o controlo de fluxo do programa elaborado de um modo mais simples:

```
>> x=2;          y=3*(x==5)

     y =
                0

>> x=2;          y=3*(x~=5)

     y =
                3

>> u= x*(x>=0)

     u =
                2
```

5. - OPERADORES LÓGICOS & | ~ xor

Os símbolos "**&**", " **|** ", e "**~**" correspondem respectivamente aos operadores lógicos **AND**, **OR** e **NOT**. Permitem efectuar operações lógicas entre matrizes com a mesma dimensão compostas por elementos zero *(0)* e um *(1)* que correspondem a **FALSO** e a **VERDADEIRO**, respectivamente. ***A & B*** realiza o AND lógico, ***A | B*** efectua o OR lógico, e ***~A*** apresenta o complemento dos elementos de ***A***. A função ***xor(A,B)*** implementa a operação OR exclusivo.

Inputs		*AND*	*OR*	*XOR*
		&	*|*	*xor*
0	*0*	*0*	*0*	*0*
0	*1*	*0*	*1*	*1*
1	*0*	*0*	*1*	*1*
1	*1*	*1*	*1*	*0*

Os operadores lógicos possuem a precedência mais baixa relativamente aos operadores aritméticos e relacionais.

A precedência entre os diversos operadores lógicos é dada por:

- NOT possui a precedência mais elevada.
- AND e OR têm igual precedência, e são avaliados da esquerda para a direita.

Pode-se usar, também os operadores lógicos ***&*** (AND) e **|** (OR). Por exemplo,

```
>> 1= = 1 & 4 = = 3

        ans =
                    0


>> 1 = = 1 | 4 = = 3

        ans =
                    1
```

6. - MANIPULAÇÃO DE VECTORES E MATRIZES

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

6.1. MATRIZES ELEMENTARES

Além das já descritas, o MATLAB dispõe de diversas funções que permitem a rápida elaboração de matrizes padrão, como sejam:

6.1.1. Geração de vectores

O caractere dois pontos, " **:** ", permite a geração de vectores no MATLAB. A declaração:

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

Gera um vector linha contendo os números de 1 a 5 com incremento unitário. Produzindo:

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

Outros incrementos, diferentes da unidade podem ser utilizados, como seja o caso do seguinte exemplo que impõe um incremento de *pi/4*.

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

y =
    0.0000    0.7854    1.5708    2.3562    3.1416
```

Também são possíveis incrementos negativos.

```
>> z = 6 : -1 : 1

z =
     6     5     4     3     2     1
```

Pode-se, também, gerar vectores linearmente espaçados fazendo uso da função ***linspace***. Por exemplo,

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

k =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

Cria um vector linearmente espaçado de 0 a 1, contendo 6 elementos.

Ao pretender-se um espaçamento logarítmico desde o valor inicial x_1 até ao valor final x_2 do vector composto por *k* elementos, a função a utilizar será:

```
>> logspace (x1 , x2 , k)
```

6.1.2. Matriz identidade

De modo a construir uma matriz identidade (quadrada ou com dimensão *n x m*, a função a utilizar é dada por:

```
>> eye(n,m)
```

6.1.3. Matriz composta por elementos unitários

No caso de ser necessário a obtenção de uma matriz composta apenas por elementos unitários, temos que:

```
>> ones(n,m)
```

6.1.4. Matriz nula (composta por elementos nulos)

Para efectuar qualquer tipo de manipulação matricial, poderá ser útil a construção de uma matriz composta por elementos nulos:

```
>> zeros(n,m)
```

6.1.5. Matriz aleatória

A elaboração de testes a qualquer programa desenvolvido no MATLAB, ou para utilização de um outro qualquer modo, poderá fazer uso de matrizes compostas por números aleatórios uniformemente distribuídos no intervalo entre 0 e 1:

```
>> rand(n,m)
```

No caso de se pretender uma distribuição normal dos números aleatórios compreendidos entre 0 e 1, a função a utilizar será:

```
>> randn(n,m)
```

6.1.6. Elementos de matrizes

Um elemento individual da matriz pode ser indicado incluindo os seus subscritos entre parêntesis. 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(3,1)
```

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

Um subscrito pode ser um vector. Se ***X*** e ***V*** são vectores, então ***X(V)*** é ***[X(V(1)), X(V(2)), ... X(V(n))]***. Para as matrizes, os subscritos vectores permitem o acesso às 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*

Então:

>> A(1:5,3)

ans =

11
17
88
19
25

Especifica uma submatriz *5x1*, ou vector coluna, que consiste nos cinco primeiros elementos da terceira coluna da matriz ***A***. Analogamente,

>> A(1:5,7:10)

ans =

74	*51*	*58*	*40*
55	*57*	*64*	*41*
56	*63*	*70*	*47*
62	*69*	*71*	*28*
68	*75*	*52*	*34*

É uma submatriz *5x4* que consiste nas primeiras cinco linhas e nas 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*

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

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

>> 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 seguinte declaração produzirá,

>> B(1:2:7,6:10) = A(5:-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

6.2. MANIPULAÇÃO DE MATRIZES

Do mesmo modo, estão implementadas no código diversas funções que permitem a manipulação e construção de matrizes genéricas.

6.2.1. Matriz diagonal ou diagonal de uma matriz

Se ***x*** é um vector, *diag(x)* é a matriz diagonal com ***x*** na diagonal;

>> x=[1 2 3 1 -1 4];
>> diag(x)

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

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

6.2.2. Matrizes triangulares

O código possui duas funções que permitem a extracção das matrizes triangular superior ***triu*** e da matriz triangular inferior ***tril*** da matriz ***A***.

>> triu(A)

No caso de se pretender extrair os elementos da matriz ***A*** sem ser relativamente à diagonal principal, utiliza-se:

>> triu(A,k)

Em que *k=0* corresponde à diagonal principal, *k>0* indica a *k*-ésima diagonal acima da diagonal principal e *k<0* o seu contrário. A obtenção da matriz triangular inferior processa-se do mesmo modo:

>> tril(A)
>> tril(A,k)

6.2.3. Troca de elementos da matriz

O MATLAB contêm funções que possibilitam a troca de colunas da esquerda para a direita relativamente a um eixo vertical : ***fliplr***, bem como permite a troca de linhas de uma matriz de cima para baixo relativamente a um eixo horizontal : ***flipud***.

```
A =
     1     4
     2     5
     3     6

>>fliplr(A) =
     4     1
     5     2
     6     3

>>flipud(A) =
     3     6
     2     5
     1     4
```

6.2.4. Redimensionamento de matrizes

Dada a matriz ***A*** composta por *n* linhas e *m* colunas, poderá ser redimensionada desde que o produto de *n x m* se mantenha constante.

>> B = reshape(A,n,m)

6.2.5. Rotação dos elementos da matriz

Os elementos da matriz ***A*** poderão ser rodados 90º no sentido dos ponteiros do relógio (ou rodados *k*90º*) através do uso da função:

>>rot90(A,k)

7. - FUNÇÕES

7.1. FUNÇÕES ESCALARES

Algumas funções no MATLAB operam essencialmente sobre escalares, mas operam sobre cada elemento se forem aplicadas a uma matriz. As funções escalares mais comuns são:

exp	Exponencial;
abs	Valor absoluto;
log	Logaritmo natural;
log10	Logaritmo base 10;
sqrt	Raiz quadrada;
sin	Seno;
asin	Arco seno;
cos	Coseno;
acos	Arco coseno;
tan	Tangente;
atan	Arco tangente;
round	Arredondamento ao inteiro mais próximo;
floor	Arredondamento ao inteiro mais próximo na direcção de menos infinito (- ∞);
ceil	Arredondamento ao inteiro mais próximo na direcção de mais infinito (+ ∞);
rem	Resto da divisão;
sign	Função sinal.

7.2. FUNÇÕES VECTORIAIS

Outras funções do MATLAB operam essencialmente sobre vectores (**linha** e **coluna**), mas numa matriz *n x m*, agem sobre coluna por coluna para produzir um vector linha com o resultado da sua aplicação para cada coluna. É possível operar sobre linha por linha transpondo a matriz, por exemplo: ***mean(A')'***.

```
A=[ 0.9103   0.3282   0.2470   0.0727   0.7665 ; 0.7622   0.6326   0.9826   0.6316   0.4777 ; 0.2625   0.7564
0.7227   0.8847   0.2378 ; 0.0475   0.9910   0.7534   0.2727   0.2749 ; 0.7361   0.3653   0.6515   0.4364   0.3593]

>>mean(A)

        ans =[ 0.5437   0.6147   0.6714   0.4596   0.4232]

>>mean(A')'

        ans =   0.4649
                0.6973
                0.5728
                0.4679
                0.5097
```

Outros exemplos de funções vectoriais que permitem a análise da informação contida nas colunas das matrizes são:

max	Valor máximo dos elementos de cada coluna;
sum	Soma dos elementos de cada coluna;
median	Valor mediano dos elementos de cada coluna;
any	Devolve 1 se qualquer elemento de cada coluna for diferente de 0;
min	Valor mínimo dos elementos de cada coluna;
prod	Produto dos elementos de cada coluna;
all	Devolve 1 se todos os elemento de cada coluna forem diferentes de 0;
sort	Organização dos elementos da coluna por ordem decrescente de valor;
std	Desvio padrão dos elementos de cada coluna.

7.3. FUNÇÕES MATRICIAIS

Grande parte da versatilidade do MATLAB vem das suas funções matriciais. As mais usadas são:

eig	Valores Próprios e Vectores Próprios;
chol	Factorização de Cholesky;
svd	Decomposição em valor singular;
inv	Inversa;
lu	Factorização triangular LU;
qr	Factorização ortogonal QR;
hess	Forma de Hessenberg;
schur	Decomposição de Schur;
expm	Matriz exponencial;
sqrtm	Matriz de raiz quadrada;
poly	Polinómio característico;
det	Determinante;
size	Tamanho;
norm	Norma 1, Norma 2, Norma F, Norma infinita;
cond	Número de condição na norma 2;
rank	Número de linhas linearmente independentes;

As funções no MATLAB podem ter argumentos simples ou múltiplos. Por exemplo, ***y = eig(A)*** ou ***eig(A)*** produzem um vector coluna contendo os Valores Próprios da matriz ***A***.

```
>>y = eig(A)

    y =
          2.7594
          0.9232
          -0.3618 + 0.1449i
          -0.3618 - 0.1449i
          -0.0613
```

Já ***[U,D] = eig (A)*** produz uma matriz ***U*** cujas colunas são os Vectores Próprios de ***A*** e a matriz diagonal ***D*** com os Valores Próprios de ***A*** na sua diagonal.

```
[U,D] = eig (A)
```

```
U =
[   -0.3454        -0.7798        0.1164-0.1629i        0.1164+0.1629i        -0.4895
    -0.5604         0.0010        0.0766+0.2393i        0.0766-0.2393i        -0.4205
    -0.4815         0.4525       -0.5920-0.0723i       -0.5920+0.0723i         0.3115
    -0.4198         0.3858        0.6555-0.2277i        0.6555+0.2277i         0.0416
    -0.3983        -0.1960       -0.0666+0.2348i       -0.0666-0.2348i         0.6963]

D =
[    2.7594         0                0                     0                     0
         0          0.9232           0                     0                     0
         0          0               -0.3618+0.1449i        0                     0
         0          0                0                    -0.3618-0.1449i        0
         0          0                0                     0                    -0.0613]
```

7.4. POLINÓMIOS

Embora o MATLAB não permita trabalhar directamente com polinómios, dispõe de um conjunto de funções dedicadas à sua manipulação-

Os Polinómios são representados no MATLAB por vectores linha que contêm os coeficientes das sucessivas potências do polinómio ordenados por ordem significativa decrescente. Por exemplo, o polinómio:

$$p(s) = s^3 - 6s^2 - 72s - 27$$

Deverá ser introduzido:

```
>> p = [1  –6  –72  –27]
```

A função ***poly*** gera o polinómio característico da matriz ***A***: *p(s) = det(s I – A)*

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

    p =
              1          -6         -72         -27
```

Além disso, esta função permite obter um polinómio a partir de um vector (coluna) que contenha as suas raízes.

As raízes de um polinómio podem ser obtidas através de:

```
>> r = roots(p)

    r =
        12.1229
        -5.7345
        -0.3884
```

As operações aritméticas de adição e subtracção de polinómios efectuam-se como nas operações com conjuntos, no entanto, a multiplicação e divisão de polinómios faz uso das funções de ***conv*** e de ***deconv***, respectivamente. Por exemplo, dados os polinómios:

$$a(s) = s^2 + 2s + 3$$
$$b(s) = 4s^2 + 5s + 6$$

A sua multiplicação é obtida a partir de:

```
>> a = [1 2 3]
>>b = [4 5 6]
>>m = conv(a,b)

    m =
          4          13          28          27          18
```

Isto é : $$m(s) = 4s^4 + 13s^3 + 28s^2 + 27s + 18$$

A divisão é obtida através da função ***deconv***, que no seguimento do caso anterior:

$$d(s) = \frac{m(s)}{a(s)} = \frac{4s^4 + 13s^3 + 28s^2 + 27s + 18}{s^2 + 2s + 3}$$

```
>>[d,r] = deconv(m,a)

    d =
          4           5           6

    r =
          0           0           0           0           0
```

Neste caso, ***d(s) = b(s)*** e ***r*** corresponde ao resto da divisão.

7.5. FUNÇÕES DE 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 optimização;
- Solução de equações diferenciais.

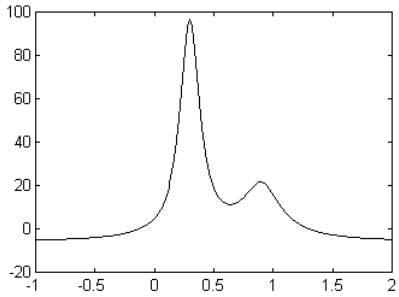
$$humps(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

As funções matemáticas são representadas no MATLAB por ficheiros "**.m**". Por exemplo, a função está disponível no MATLAB como um ficheiro "**.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))
```



7.5.1. Integração numérica

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

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

    q =
                26.3450
```

Os dois comandos do MATLAB para integração numérica são:

quad	Calcular numericamente o integral, método de baixa ordem. (Adaptação recursiva da Regra de Simpson);
quad8	Calcular numericamente o integral, método de alta ordem. (Regra de Newton).

7.5.2. Equações não-lineares e de optimização

Os comandos para equações não-lineares e de optimização incluem:

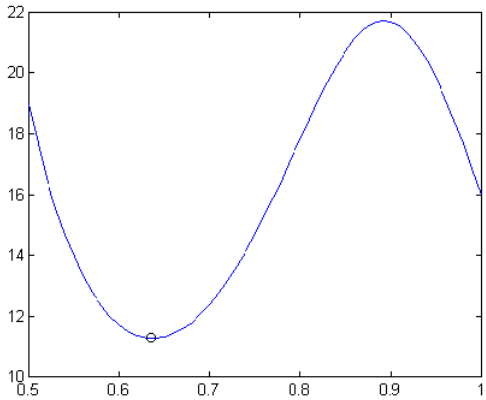
fmin	Minimizar função de uma variável;
fmins	Minimizar função de várias variáveis;
fzero	Encontrar zero de função de uma variável.

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:

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



Pode visualizar-se 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,

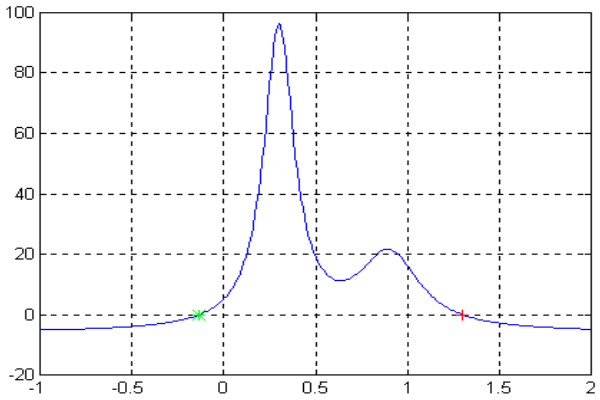
```
>>xz1 = fzero('humps',0)

    xz1 =
                -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
```



7.5.3. Solução de equações diferenciais

Os comandos do MATLAB para resolver equações diferenciais ordinárias são:

ode23	Resolver equação diferencial. Método de baixa ordem; (Método de Runge-Kutta de 2ª e 3ª Ordem).
ode23p	Resolver e visualizar soluções;
ode45	Resolver equação diferencial. Método de alta ordem; (Método de Runge-Kutta-Fehlberg de 4ª e 5ª Ordem).

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

$x + (x^2 - 1) \cdot x + x = 0$

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

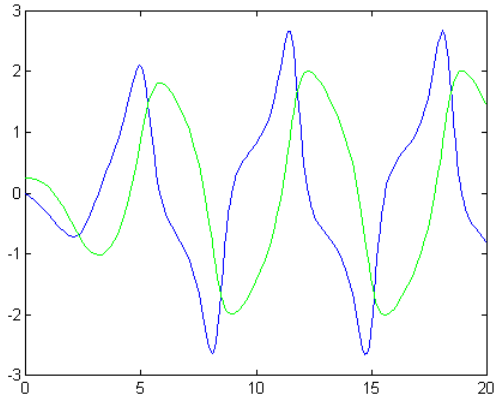
$x_1 = x_1 \cdot (1-x_2^2) - x_2$
$x_2 = x_1$

O primeiro passo para simular esse sistema é criar um ficheiro "**.m**" que contenha as equações diferenciais. Por exemplo, o ficheiro ***volpol.m***:

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

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)
```



8. - GRÁFICOS

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

8.1. GRÁFICOS BIDIMENSIONAIS

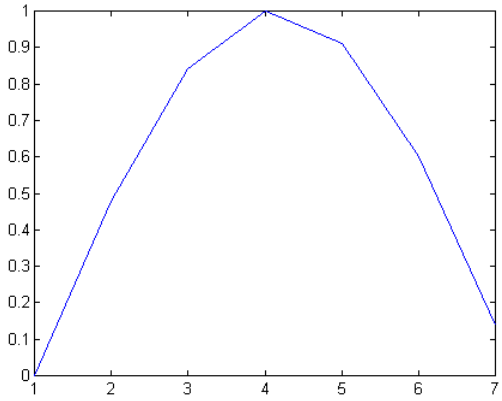
Comandos de geração de gráficos bidimensionais:

plot	Gráfico linear.
loglog	Gráfico em escala logarítmica.
semilogx	Gráfico em escala semi-logarítmica (eixo x).
semilogy	Gráfico em escala semi-logarítmica (eixo y).
fill	Desenhar polígono 2D.
polar	Gráfico em coordenadas polar.
bar	Gráfico de barras.
stem	Gráfico de sequência discreta.
stairs	Gráfico em degrau.
errorbar	Gráfico do erro.
hist	Histograma.
rose	Histograma em ângulo.
compass	Gráfico em forma de bússola.
feather	Gráfico em forma de pena.
fplot	Gráfico da função.
comet	Gráfico com trajectória de cometa.

Se ***Y*** é um vector, ***plot(Y)*** produz um gráfico linear dos elementos de ***Y*** versus o índice dos elementos de ***Y***. Por exemplo, para exibir os números [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0,14], basta introduzir o vector e executar o comando ***plot***:

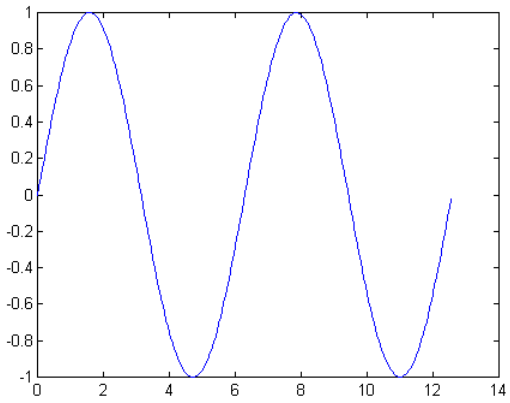
```
>> Y = [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0,14];
>> plot(Y)
```

O resultado é mostrado na *Janela Gráfica*:



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

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



O MATLAB pode também exibir múltiplas linhas em apenas um gráfico. Existem duas maneiras, a primeira resulta no uso de apenas dois argumentos, como em ***plot(X,Y)***, onde ***X*** e/ou ***Y*** são matrizes. Então:

- Se ***Y*** é uma matriz e ***X*** um vector, ***plot(X,Y)*** exibe sucessivamente as linhas ou colunas de **Y** versus o vector **X**.

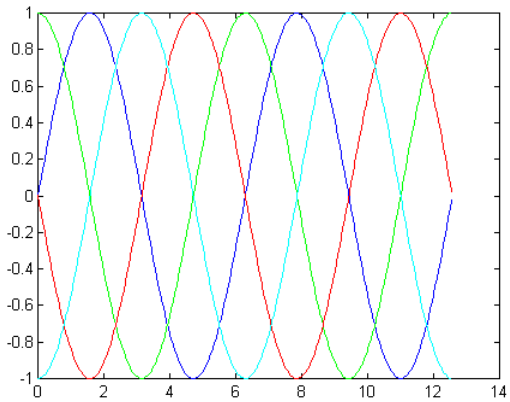
- Se ***X*** é uma matriz e ***Y*** é um vector, ***plot(X,Y)*** exibe sucessivamente as linhas ou colunas de ***X*** versus o vector ***Y***.

- Se ***X*** e ***Y*** são matrizes com mesma dimensão, ***plot(X,Y)*** exibe sucessivamente as colunas de ***X*** versus as colunas de ***Y***.

- Se ***Y*** é uma matriz, ***plot(Y)*** exibe sucessivamente as colunas de ***Y*** versus o índice de cada elemento da linha de ***Y***.

O segundo método (e mais fácil) de exibir gráficos com múltiplas linhas faz uso do comando ***plot*** com múltiplos argumentos. Por exemplo:

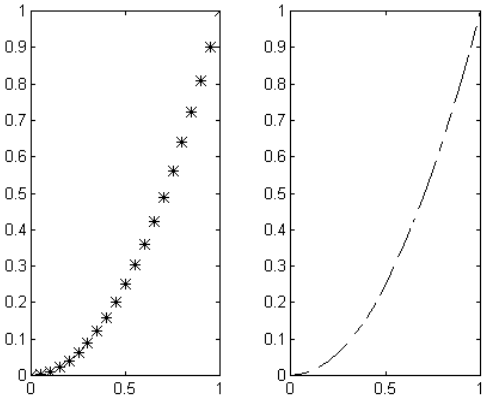
```
>> plot(t, sin(t), t, cos(t), t, sin(t + pi), t, cos(t + pi))
```



8.2. ESTILOS DE LINHAS E SIMBOLOS

Os tipos de linhas, símbolos e cores usados para exibir gráficos podem ser controlados se os padrões não forem suficientes. Por exemplo,

```
>> X = 0:0.05:1;
>> subplot(121), plot(X,X.^2,'k*')
>> subplot(122), plot(X,X.^2,'k --')
```



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

TIPO DE LINHA	
_	_____
--	----------
-.	-.-.-.-.-.-.-.-.-.-.-.-.
.	.....

TIPO DE PONTO	
.	**.....**
*	* * * * * * * *
o	o o o o o o o o o o
+	+ + + + + + + + + +
x	x x 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

8.3. NÚMEROS COMPLEXOS

Quando os argumentos a exibir são complexos, a parte imaginária é ignorada, excepto quando é dado simplesmente um argumento complexo. Para este caso especial é exibida a parte real versus a parte imaginária. Então, ***plot(Z)***, quando ***Z*** é um vector complexo, é equivalente a ***plot(real(Z),imag(Z))***.

8.4. ESCALA LOGARÍTMICA, COOR. POLARES E GRÁFICOS DE BARRAS

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

- ***polar(Theta,R)*** : Exibe o gráfico em coordenadas polares o ângulo ***Theta***, em radianos, versus o raio ***R***;

- ***loglog(x,y)*** : Exibe o gráfico em escala $\log_{10}$ x $\log_{10}$;

- ***semilogx(x,y)*** : Exibe o gráfico em escala semi-logarítmica. O eixo x é $\log_{10}$ e o eixo y é linear;

- ***semilogy(x,y)*** : Exibe o gráfico em escala semi-logarítmica. O eixo x é linear e o eixo y é $\log_{10}$;

- ***bar(X)*** : Mostra um gráfico de barras dos elementos do vector ***X***, e não aceita múltiplos argumentos.

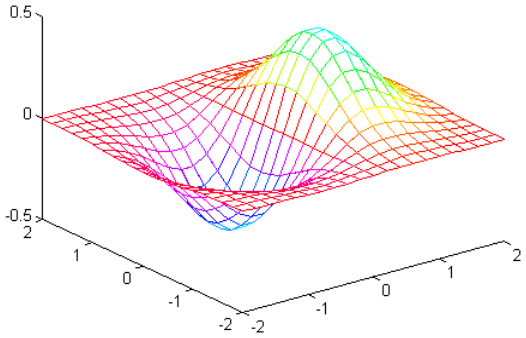
8.5. EXIBIÇÃO DE GRÁFICOS TRIDIMENSIONAIS E DE CONTORNO

Estes são alguns dos comandos de exibição de gráficos tridimensionais e de contornos.

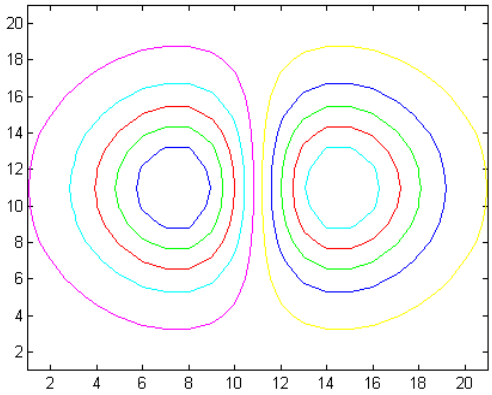
plot3	Exibição do gráfico em espaço 3D.
fill3	Desenhar polígono 3D.
comet3	Exibir em 3D com trajectória de cometa.
contour	Exibir contorno 2D.
contour3	Exibir contorno 3D.
clabel	Exibir contorno com valores.
quiver	Exibir gradiente.
mesh	Exibir malha 3D.
meshc	Combinação mesh/contour.
surf	Exibir superfície 3D.
surfc	Combinação surf/contour.
surfil	Exibir superfície 3D com iluminação.
slice	Plot visualização volumétrica.
cylinder	Gerar cilindro.
sphere	Gerar esfera.

O comando ***mesh(X,Y,Z)*** cria uma perspectiva tridimensional exibindo 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)
```



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



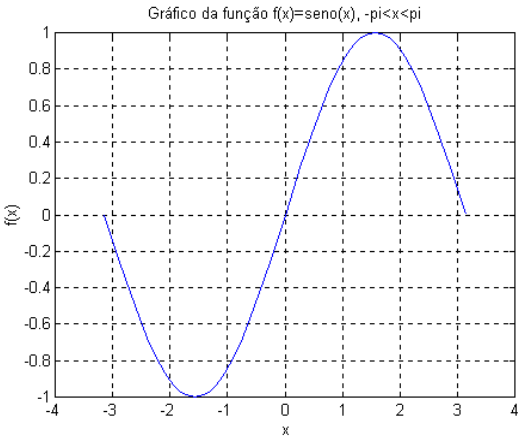
8.6. ANOTAÇÕES NO GRÁFICO

O MATLAB possui comandos de fácil utilização para adicionar informações a gráficos:

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 grelha.

Por exemplo:

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



9. - CONTROLO DO FLUXO DE PROGRAMA

Os comandos que controlam o fluxo especificam a ordem e a estrutura de cálculo do programa. No MATLAB estes comandos são semelhantes aos usados na linguagem C, mas com uma estrutura diferente.

9.1. CICLO for

O ciclo ***for*** é o controlador de fluxo mais simples e usado na programação MATLAB. Analisando a expressão:

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

Pode notar-se que o ciclo ***for*** é dividido em três partes:

- A primeira parte (**i=1**) é realizada uma vez, antes do ciclo ser inicializado.
- A segunda parte é o teste ou condição que controla o ciclo, (**i<=5**). Esta condição é avaliada; se verdadeira, o corpo do ciclo (**X(i)=i^2**) é executado.
- A terceira parte acontece quando a condição se torna falsa e o ciclo termina. O comando ***end*** é usado como limite inferior do corpo do ciclo.

São comuns construções em que conjuntos de ciclos ***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;
```

9.2. CICLO while

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

```
>>a = 1; b = 15;
        while a<b,
                clc
                a = a+1
                b = b-1
                pause(1)
        end
    disp('fim do loop')
```

A condição **a<b** é testada. Enquanto a condição for verdadeira o corpo do ciclo será executado.

Quando o teste se tornar falso o ciclo terminará, e a execução do programa continuará no comando que segue após o ***end*** do ciclo.

9.3. DECLARAÇÕES if E break

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

```
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;
                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.

Por vezes é conveniente controlarmos a saída dum ciclo de outro modo além do teste, no início ou no fim do mesmo. O comando ***break*** permite uma saída antecipada de um ciclo ***for*** ou ***while***. Um comando ***break*** faz com que o ciclo mais interno seja terminado imediatamente. Por exemplo,

```
%modifica a matriz A
clc
x = 's';
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) => ');
                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
```

10. - FICHEIROS ".m"

Os comandos do MATLAB são normalmente introduzidos através da *Janela de Comando*, onde uma única linha de comando é introduzida e processada imediatamente. O MATLAB é também capaz de executar sequências de comandos armazenadas em ficheiros.

Os ficheiros que contêm as declarações do MATLAB são chamados ficheiros "**.m**", e consistem em sequências de comandos normais do MATLAB, possibilitando incluir outros ficheiros "**.m**" escritos no formato texto (ASCII).

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

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

```
%Exibir uma função y=ax^2 + bx + c no intervalo -5<x<5
clear
aux='s';
while aux= = 's',
        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
        clc
        close
        aux=input('Exibir outro ? (s/n) = => ','s');
end
```

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 possibilitar ao utilizador a introdução de um dado do problema a partir da *Janela de Comando*, ***pause*** provoca uma pausa na execução do ficheiro 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*.

11. - 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 efectuadas, como executar programas externos, trocar o directório de trabalho, listagem do directório, e serão detalhadas a seguir.

11.1. 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 directório de trabalho actual
dir	lista o conteúdo do directório actual
delete	exclui ficheiro
type	mostra o conteúdo do ficheiro texto
what	lista ficheiros ".m", ".mat" e ".mex".

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

11.2. EXECUÇÃO DE PROGRAMAS EXTERNOS

O caractere 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 procedimento de programação a partir 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 introduzir o caractere 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*** (Bloco de Notas) do *WINDOWS*, sem sair do MATLAB, introduzir:

```
>> ! 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***, ***format***, ***ren***, ***mkdjr***, ***rmdir***, ...

11.3. IMPORTAÇÃO E EXPORTAÇÃO DE FICHEIROS

Os dados contidos na *Área de Trabalho* do MATLAB podem ser armazenados em ficheiros, 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*** :

save	Guardar dados no ficheiro binário "matlab.mat".
save X	Guardar a matriz X no ficheiro binário "x.mat".
save arql X Y Z	Guardar as matrizes X, Y e Z no ficheiro binário "arq1.mat".
save arq2.sai X Y Z -ascii	Guardar as matrizes X., Y e Z no ficheiro texto "arq2.sai" com 8 dígitos.
save arq3.sai X Y Z -ascii -double	Guardar as matrizes X., Y e Z no ficheiro 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, supondo que um programa em linguagem C, depois de executado, elabora o ficheiro "***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
```

Após a execução do comando:

```
>> load teste.sai
```

O MATLAB importa a matriz, que passa a ser chamada de ***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***:

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 realçar 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 os dados são importados através do comando ***load***, apenas uma variável com nome do ficheiro 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

    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
```

12. - REFERÊNCIAS BIBLIOGRÁFICAS

[1] *The Student Edition of MATLAB*, The Math Works Inc., Prentice Hall, 1992.

[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.