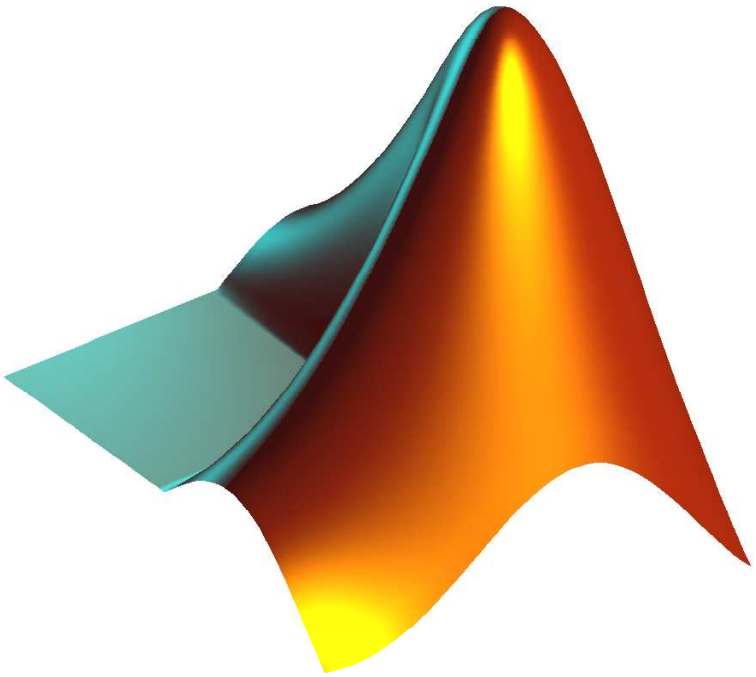
Iniciação ao MATLAB

Maria Irene Falcão



Universidade do Minho

2001

Conteúdo

1 Introdução

1.1 A *package* MATLAB

O MATLAB é uma *package* destinada ao estudo de problemas científicos e de engenharia, ou, de uma forma mais geral, ao estudo de quaisquer problemas em que haja um trabalho computacional significativo a realizar que possa (ou deva) utilizar matrizes. Este *software* é produzido pela companhia americana The Math Works, Inc e o seu nome deriva do inglês "matrix laboratory".

O MATLAB trabalha com matrizes quadradas ou rectangulares, com elementos reais ou complexos e derivou dos projectos LINPACK e EISPACK que são considerados como a origem de algum do melhor *software* numérico disponível para computação com matrizes. Para além da manipulação fácil de matrizes, a *package* permite o acesso a um número crescente de rotinas incorporadas, potencialidades gráficas a 2 e 3 dimensões e a possibilidade de configurar o *software* às necessidades de cada utilizador.

Este texto pretende ser uma breve introdução ao MATLAB e foi escrito com o objectivo de servir de apoio a disciplinas no âmbito da Análise Numérica. Os tópicos apresentados reflectem, de algum modo, este objectivo.

1.2 Iniciar uma sessão em MATLAB

Quando se invoca o MATLAB, é criada uma janela *MATLAB Command Window* através da qual é possível comunicar com o interpretador do MATLAB. Quando aparece a *prompt* $\gg$, o MATLAB está pronto a receber instruções.

O texto de cada sessão de trabalho pode ser guardado através do comando

$\gg$ **diary** *sessao*

Todo o texto relativo à sessão que se inicia imediatamente a seguir à instrução **diary** *sessao* e termina com o comando

$\gg$ **diary off**

é guardado num ficheiro de texto chamado *sessao*.

As variáveis definidas ou obtidas numa sessão de trabalho podem também ser guardadas num ficheiro para serem posteriormente usadas. Assim, o comando

$\gg$ **save** *variaveis*

guarda as variáveis do espaço de trabalho num ficheiro chamado *variaveis.mat*. Para recuperar estes valores basta fazer

≫ **load** *variaveis*

É ainda possível guardar apenas algumas variáveis. Por exemplo, o comando

≫ **save** *apenas* X Y

guarda as variáveis X e Y num ficheiro chamado *apenas.mat*.

1.3 Declarações e variáveis

As declarações no MATLAB são frequentemente da forma

variável=expressão

ou simplesmente

expressão.

No primeiro caso, o valor da expressão é atribuído à variável *variável* para uso futuro. Quando o nome da variável é omitido (assim como o sinal =), o MATLAB cria automaticamente uma variável com o nome *ans* (de *answer*). Por exemplo, as declarações

≫ a=2*3

≫ 2*4

originam, respectivamente

a=6

ans=8

O nome de uma variável deve sempre começar por uma letra, seguido de um conjunto de letras ou números (ou ainda o sinal _), até ao máximo de 19 caracteres. O MATLAB **distingue as letras minúsculas das maiúsculas**. Assim, a e A não representam a mesma variável. **Todas as funções têm nomes escritos em minúsculas** (ver Secção 2.6).

Sempre que se pretenda que o resultado de uma operação ou atribuição não apareça no ecrã (evitando assim o aparecimento de uma quantidade de informação pouco útil), deve usar-se o símbolo ;. A declaração

≫ a=2*3;

atribui o valor 6 à variável a, mas não mostra no ecrã o resultado dessa atribuição.

Quando a expressão é muito complicada, necessitando de mais de uma linha para ser definida, deve usar-se o símbolo ... antes de mudar de linha, para indicar que a expressão continua.

1.4 Definição e manipulação de matrizes

O MATLAB trabalha essencialmente com um tipo de objecto, uma matriz numérica rectangular com elementos eventualmente complexos. Escalares são entendidos como matrizes 1×1 e os vectores como matrizes $1 \times n$ ou $n \times 1$. Assim, as operações e comandos em MATLAB devem ser entendidos duma forma natural, enquanto operações entre matrizes.

A maneira mais simples de definir uma matriz pequena é introduzir explicitamente os seus elementos da seguinte forma: cada elemento da matriz é separado pelo símbolo , (ou espaço) e cada linha da matriz é separada por ; (ou mudança de linha). Os elementos da matriz devem estar rodeados dos símbolos [(no início) e] (no fim).

Por exemplo as atribuições

$$\gg A = [1, 1, 1; 2, 2, 2; 3, 3, 3]$$

e

$$\gg B = \begin{array}{l} [1 \quad 1 \quad 1 \\ \ \ 2 \quad 2 \quad 2 \\ \ \ 3 \quad 3 \quad 3] \end{array}$$

produzem exactamente a mesma matriz: $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$

Para obter um determinado elemento da matriz A, basta indicar a sua linha e coluna. Por exemplo, o comando

$$\gg a23 = A(2, 3)$$

tem como resultado

$$a23 = 2.$$

Uma das vantagens do MATLAB é a de não ser necessário dimensionar a priori as matrizes. A cada nova instrução é feito o redimensionamente da matriz. Assim, as declarações

$$\gg A(5, 5) = 1$$

e

$$\gg B(5, 1) = 1$$

produzem as novas matrizes

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{e} \qquad B = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Outra forma simples de obter matrizes maiores à custa de matrizes menores já definidas é a seguinte: se pretendermos acrescentar à matriz A inicial a linha 4 4 4, basta fazer

$\gg l = [4, 4, 4]$

$\gg A = [A; l]$

ou apenas

$\gg A = [A; 4\ 4\ 4]$,

para se obter a nova matriz $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{pmatrix}$

Para obter submatrizes de uma dada matriz deve usar-se o símbolo : para indicar quais as linhas e colunas da matriz inicial a considerar. Por exemplo, considerando a matriz

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

a declaração

$\gg B = A(2:3, 1:3)$

resultará numa matriz que contém as linhas 2 e 3 e as colunas 1 a 3 da matriz inicial, enquanto que a atribuição

$\gg C = A(:, 1:2)$

resultará numa matriz que tem as mesmas linhas e as 2 primeiras colunas de A, i.e

$$B = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{e} \qquad C = \begin{pmatrix} 1 & 2 \\ 4 & 5 \\ 7 & 8 \\ -1 & -2 \end{pmatrix}$$

Também é possível apagar linhas e colunas de uma matriz, usando os símbolos [ ]. Para retirar a segunda linha da matriz A definida anteriormente basta fazer

$\gg A(2,:) = [\ ]$

Vectores com componentes inteiras podem também ser usados como índices. Por exemplo,

$\gg D = A([2\ 4],:)$

produzirá uma matriz D cujas linhas são as linhas 2 e 4 da matriz A.

$D = \begin{pmatrix} 4 & 5 & 6 \\ -1 & -2 & -3 \end{pmatrix}$

Além disso, vectores como índices podem aparecer em ambos os lados de uma atribuição. Por exemplo, sendo E a matriz

$$E = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{pmatrix},$$

a atribuição

$\gg A([1\ 4],:) = E([2\ 4],:)$

resultará na matriz

$A = \begin{pmatrix} 2 & 2 & 2 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 4 & 4 & 4 \end{pmatrix}.$

O símbolo : permite ainda definir vectores e/ou matrizes de modo muito simples. Por exemplo,

$\gg$ x=-2:1:2

define o vector

$$x = (-2\ \ -1\ 0\ 1\ 2).$$

Do mesmo modo,

$\gg$ y=[0:3:9;2:2:8;5:5:20]

define a matriz

$$y = \begin{pmatrix} 0 & 3 & 6 & 9 \\ 2 & 4 & 6 & 8 \\ 5 & 10 & 15 & 20 \end{pmatrix}$$

1.5 Números e expressões aritméticas

O MATLAB usa a notação decimal convencional para representar números, sendo também possível incluir-se, na representação de um número, potências de base 10. Assim, as expressões 0.001 e 1E-3 representam o mesmo número. A notação usada para a unidade imaginária é o i ou j. As expressões 1+2i e 1+2j representam o número complexo cuja parte real é 1 e a parte imaginária é 2.

As expressões podem ser construídas usando os operadores aritméticos usuais e correspondentes precedências. Assim, tem-se

+ adição

- subtracção

* multiplicação

/ divisão à direita [1]

\ divisão à esquerda

ˆ potenciação

O MATLAB tem incorporadas funções matemáticas elementares tais como *abs*, *sqrt*, *log*, etc. Para uma lista completa destas e outras funções, veja Secção 2.6.

Internamente, o MATLAB efectua todas as operações em precisão dupla (a unidade de erro de arredondamento **eps**$\approx 10^{-16}$). Para controlar o formato de saída dos resultados pode usar-se o comando **format**. Por defeito, o MATLAB apresenta os resultados no **format short** que corresponde ao uso de aproximadamente 5 algarismos significativos. É possível alterar este formato, usando formatos que permitem mais algarismos significativos ou usam a notação científica, como se indica na tabela seguinte

format	o mesmo que short.
format short	notação de vírgula fixa, com 5 dígitos.
format long	notação de vírgula fixa, com 15 dígitos.
format short e	notação de vírgula flutuante, com 5 dígitos.
format long e	notação de vírgula flutuante, com 15 dígitos.
format short g	o melhor dos formatos de vírgula fixa ou flutuante, com 5 dígitos.
format long g	o melhor dos formatos de vírgula fixa ou flutuante, com 15 dígitos.

[1] As operações com matrizes tornam conveniente o uso de dois símbolos para a divisão; ver Secção 2.4.

Por exemplo:

≫ X=[4/3 sqrt(2)/2.5E4]

X=[1.3333 0.0001]

≫ format short;X

X=[1.3333 0.0001]

≫ format short e;X

X=[1.3333e+000 5.6569e-005]

≫ format short g;X

X=[1.3333 5.6569e-005]

≫ format long;X

X=[1.33333333333333 0.00005656854249]

2 Operações com matrizes

2.1 Transposta de uma matriz

O símbolo ' denota a transposta de uma matriz, no caso em que esta é real ou a transconjugada de uma matriz, se esta for complexa.[2] Assim, as declarações

≫ A=[1 2 3;4 5 6;7 8 9]

≫ B=A'

≫ X=[-1+i 2-i 3]'

resultam nas matrizes

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \qquad X = \begin{pmatrix} -1-i \\ 2+i \\ 3 \end{pmatrix}$$

[2] A transposta de uma matriz complexa A pode obter-se fazendo $A.'$ ou **conj**(A').

2.2 Soma de matrizes

Os símbolos + e - designam soma e subtracção de matrizes e estas operações estão definidas de forma usual. Assim, a atribuição

≫ C=A+B

resulta na matriz

$$C = \begin{pmatrix} 2 & 6 & 10 \\ 6 & 10 & 14 \\ 10 & 14 & 18 \end{pmatrix}$$

No MATLAB é ainda possível definir a operação A+B ou A-B sempre que uma das matrizes tem ordem 1, i.e é um escalar. Neste caso, a matriz resultante é a que se obtém somando ou subtraindo o escalar a todos os elementos da outra matriz.

As declarações

≫ D=A-1

≫ D=-1+A

originam exactamente a mesma matriz

$$D = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$$

2.3 Produto de matrizes

O símbolo * denota multiplicação de matrizes e esta operação está definida da forma usual. Por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \qquad x = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, \qquad y = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix},$$

então as atribuições

≫ M1=A*B

≫ M2=4*A

≫ M3=x'*y

originam as matrizes

$$M1 = \begin{pmatrix} 22 & 28 \\ 49 & 64 \\ 76 & 100 \end{pmatrix}, \qquad M2 = \begin{pmatrix} 4 & 8 & 12 \\ 16 & 20 & 24 \\ 28 & 32 & 36 \end{pmatrix}, \qquad M3 = \begin{pmatrix} 4 & 5 & 6 \\ 8 & 19 & 12 \\ 12 & 15 & 18 \end{pmatrix},$$

enquanto que

$\gg$ M4=B*A

$\gg$ M5=x*y

$\gg$ M6=A*x

produzem

??? Error using ==¿ * Inner matrix dimensions must agree.

2.4 Quociente de matrizes

No MATLAB existem dois símbolos para representar a "divisão" de duas matrizes: \ e /. Se A é uma matriz quadrada invertível, então $A\backslash B$ e B/A correspondem formalmente ao produto à esquerda e à direita, de B pela inversa de A, i.e.

$$A\backslash B = A^{-1} * B$$

$$B/A = B * A^{-1}$$

Como seria de esperar, o resultado destas operações não passa pelo cálculo explícito da inversa de A, mas sim pela resolução de sistemas. Em geral

$$X = A\backslash B \text{ é solução da equação } A * X = B$$

$$Y = B/A \text{ é solução da equação } Y * A = B$$

Seja

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{e} \qquad B = \begin{pmatrix} 3 & -3 & 1 \\ 15 & 12 & 13 \\ 24 & 18 & 19 \end{pmatrix}$$

Então, se

$\gg X = A\backslash B$

obtém-se

$$X = \begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & 1 \\ 1 & 3 & 2 \end{pmatrix}$$

Naturalmente que se as matrizes forem de ordem 1, i.e. escalares, as duas divisões corresponden à divisão usual. Assim, $2\backslash 8 = 8/2 = 4$ e $8\backslash 2 = 2/8 = 0.25$.

O problema da resolução de sistemas será retomado no Anexo A.

2.5 Operações elemento a elemento

As operações elemento a elemento diferem das operações usuais com matrizes, mas podem ter grande aplicação prática. Para indicar que uma dada operação é para ser feita elemento a elemento deve usar-se o ponto (.) imediatamente antes do operador.

Por exemplo, se

$$A = X.\hat{}\,Y,\ B = X.*Y \ \text{ e } \ C = X./Y,$$

então

$$a_{ij} = (x_{ij})^{y_{ij}},\ b_{ij} = x_{ij} * y_{ij} \ \text{ e } c_{ij} = x_{ij}/y_{ij}.$$

Para que estas operações possam ser executadas, as matrizes (ou vectores) X e Y têm que ter a mesma ordem. Note-se que $X. + Y$ e $X. - Y$ não estão definidas (Porquê?).

Consideremos as matrizes

$$X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \text{e} \qquad Y = \begin{pmatrix} 2 & 3 \\ 3 & 0 \end{pmatrix}$$

As seguintes operações elemento a elemento

$\gg E_1 = X.*Y$

$\gg E_2 = Y./X$

$\gg E_3 = X.\hat{}\,Y$

produzem as matrizes

$$E_1 = \begin{pmatrix} 2 & 6 \\ 9 & 0 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 2 & 1.5 \\ 1 & 0 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 1 & 8 \\ 27 & 1 \end{pmatrix}$$

2.6 Matrizes especiais

O MATLAB tem incorporadas algumas funções muito úteis que permitem definir matrizes muito usadas. Para definir, por exemplo a matriz identidade, a matriz nula ou a matriz "constante"podem usar-se as funções **eye**, **zeros** ou **ones**[3], cujo modo de utilização pode ser obtido invocando o *help* do MATLAB.

[3]O nome das funções, tal como já foi referido, deve ser escrito em minúsculas

```
≫ help eye
 EYE   Identity matrix.
       EYE(N) is the N-by-N identity matrix.
       EYE(M,N) or EYE([M,N]) is an M-by-N matrix with 1's on
       the diagonal and zeros elsewhere.
       EYE(SIZE(A)) is the same size as A.

≫ help zeros
 ZEROS   All zeros.
         ZEROS(N) is an N-by-N matrix of zeros.
         ZEROS(M,N) or ZEROS([M,N]) is an M-by-N matrix of zeros.
         ZEROS(SIZE(A)) is the same size as A and all zeros.

≫ help ones
 ONES   All ones.
        ONES(N) is an N-by-N matrix of ones.
        ONES(M,N) or ONES([M,N]) is an M-by-N matrix of ones.
        ONES(SIZE(A)) is the same size as A and all ones.
```

As declarações

```
≫ A1=eye(3)

≫ A2=zeros(2,3)

≫ A3=2*ones(size(A2))
```

produzem as matrizes

$$A1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad A2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad A3 = \begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix}$$

2.7 Funções

O MATLAB contém um conjunto grande de funções já definidas. Algumas dessas funções são funções matriciais, como por exemplo det(A), que calcula o determinante de uma matriz quadrada A, outras estão definidas elemento a elemento. Por exemplo, cos(A) é uma matriz cujos elementos c_{ij} são os valores do cosseno dos elementos de $A = (a_{ij})$, i.e. $c_{ij} = \cos(a_{ij})$.

Segue-se uma lista de funções dividida por 5 categorias: funções matemáticas elementares, matrizes elementares e manipulação de matrizes, funções matriciais, polinómios e análise de dados. Uma lista completa de todas as funções existentes em

cada uma destas categorias pode ser obtida invocando o *help*: help elfun, help elmat, help matfun, help polyfun, help datafun.[4]

Funções Matemáticas Elementares

♢ Funções Trigonométricas

sin - *Seno.*
sinh - *Seno hiperbólico.*
asin - *Arco-seno.*
asinh - *Arco-seno hiperbólico.*
cos - *Cosseno.*
cosh - *Cosseno hiperbólico.*
acos - *Arco-cosseno.*
acosh - *Arco-cosseno hiperbólico.*
tan - *Tangente.*
tanh - *Tangente hiperbólica.*
atan - *Arco-tangente.*
atanh - *Arco-tangente hiperbólica.*
sec - *Secante.*
sech - *Secante hiperbólica.*
asec - *Arco-secante.*
asech - *Arco-secante hiperbólica.*
csc - *Cossecante.*
csch - *Cossecante hiperbólica.*
acsc - *Arco-cossecante.*
acsch - *Arco-cossecante hiperbólica.*
cot - *Cotangente.*
coth - *Cotangente hiperbólica.*
acot - *Arco-cotangente.*
acoth - *Arco-cotangente hiperbólica.*

♢ Função Exponencial

exp - *Exponencial.*
log - *Logaritmo neperiano.*
log10 - *Logaritmo na base 10.*
sqrt - *Raíz quadrada.*

♢ Funções Complexas

abs - *Módulo.*
conj - *Conjugado.*
imag - *Parte imaginária.*
real - *Parte real.*

[4] As diversas categorias de funções encontram-se em MATLAB\TOOLBOX\MATLAB.

Matrizes Elementares e Manipulação de Matrizes

◇ Matrizes Elementares

zeros - *Matriz nula.*
ones - *Matriz com todos os elementos 1.*
eye - *Matriz identidade.*

◇ Variáveis especiais e constantes

ans - *Resposta mais recente.*
eps - *Unidade de erro de arredondamento.*
realmax - *Maior número em vírgula flutuante.*
realmin - *Menor positivo em vírgula flutuante.*
pi - *3.1415926535897....*
i, j - *Unidade imaginária.*
inf - *Infinito.*
NaN - *Not-a-Number.*

◇ Manipulação de Matrizes

diag - *Criar ou extrair diagonais.*
reshape - *Redimensionar uma matriz.*
tril - *Extrair parte triangular inferior.*
triu - *Extrair parte triangular superior.*

Funções Matriciais

◇ Análise de Matrizes

cond - *Número de condição de uma matriz.*
norm - *Norma matricial ou vectorial.*
rank - *Característica de uma matriz.*
det - *Determinante de uma matriz.*
trace - *Traço de uma matriz.*
null - *Núcleo de uma matriz.*

◇ Equações Lineares

\ e / - *Solução de uma equação linear; use "help slash".*
chol - *Decomposição de Cholesky.*
lu - *Decomposição LU.*
inv - *Inversa de uma matriz.*

◇ Valores Próprios

eig - *Valores e vectores próprios.*
poly - *Polinómio característico.*
hess - *Forma de Hessenberg.*

◊ Funções

expm	- *Exponencial de uma matriz.*
logm	- *Logaritmo de uma matriz.*
sqrtm	- *Raiz quadrada de uma matriz.*

Polinómios

roots	- *Raízes de um polinómio.*
poly	- *Construção de um polinómio, dadas as suas raízes.*
polyval	- *Avaliar um polinómio.*
polyvalm	- *Avaliar um polinómio, cujo argumento é uma matriz.*
polyder	- *Derivada de um polinómio.*
conv	- *Produto de polinómios.*
deconv	- *Divisão de polinómios.*

Análise de Dados

max	- *Máximo.*
min	- *Mínimo.*
mean	- *Média.*
median	- *Mediana.*
std	- *Desvio padrão.*
sort	- *Ordenação (ascendente).*
sum	- *Soma dos elementos.*
prod	- *Produto dos elementos.*
cross	- *Produto vectorial.*
dot	- *Produto escalar.*

Para conhecer o modo de utilização de cada função, basta fazer **help** seguido do nome da função pretendida. Por exemplo,

≫ help roots

indica

```
ROOTS   Find polynomial roots.
        ROOTS(C) computes the roots of the polynomial whose coefficients
        are the elements of the vector C. If C has N+1 components,
        the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).

        See also POLY.
```

Outra facilidade do MATLAB pode ser obtida recorrendo ao uso de **lookfor** *palavra* que permite encontrar todas as funções que contêm a palavra *palavra* na primeira linha

de comentário. Por exemplo,

≫ lookfor roots

tem como resultado

ROOTS Find polynomial roots.

3 Programar em MATLAB

As instruções em MATLAB são geralmente dadas e executadas linha a linha. É, no entanto, possível executar uma sequência de comandos que está guardada num ficheiro. Ficheiros que contêm código em linguagem MATLAB são chamados *M-files* e são do tipo **nome_ficheiro.m**. Uma *M-file* consiste numa sequência de comandos MATLAB que pode inclusivamente fazer referência a outras *M-files*. As *M-files* podem criar-se usando um qualquer editor de texto. A versão 5.3 do MATLAB tem já um editor de texto associado. Para aceder a este editor basta seleccionar a opção **New** do menu **FILE** para criar uma nova M-file ou **Open** para editar uma M-file já existente. Qualquer texto a seguir ao símbolo % é considerado comentário.

Há dois tipos de *M-files* que podem ser usados: *scripts* e funções.

3.1 *Scripts*

Quando uma *script* é invocada, o MATLAB executa os comandos encontrados no ficheiro e as variáveis que aparecem na *script* são globais. Este tipo de ficheiro é muito usado quando há necessidade de executar um grande número de operações.

Suponhamos, por exemplo, que se pretende calcular o raio espectral da matriz[5]

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

Para o efeito, no menu do MATLAB, seleccione a opção **New M-file** do menu **File**, para criar um ficheiro de texto chamado, por exemplo, *raio.m* com a seguinte sequência de comandos

[5] O valor $\rho(A) = \max_i\{|\lambda_i| : \lambda_i$ é valor próprio de $A\}$ é chamado o raio espectral de A

```
    A=[1 2 3;4 5 6;7 8 9];
%   Cálculo dos valores próprios de A
    vp=eig(A);
%   Cálculo do maior dos vp em módulo
    r=norm(vp,inf)
```

Para executar esta *script* basta fazer, na janela do MATLAB

≫ raio

obtendo-se

raio=16.1168

A matriz A, o vector vp e a variável r estão disponíveis e podem ser de novo utilizadas.

3.2 Funções

Uma *M-file* que contém a palavra **function** no início da primeira linha é chamada uma **Função**. As funções diferem das *scripts* na medida em que podem ser usados argumentos e as variáveis definidas e manipuladas dentro de uma função são locais, i.e não operam globalmente no espaço de trabalho.

Uma função é definida do seguinte modo:

function *parâmetros_saida=nome_função(parâmetros_entrada)*

Para resolver o problema do cálculo do raio espectral de uma dada matriz pode criar-se uma função *fraio.m* do género

```
            function raio=fraio(A)
% FRAIO     Calcula o raio espectral
%           de uma dada matriz
            vp=eig(A);
            raio=norm(vp,inf);
```

A função **fraio** passa a fazer parte das funções do MATLAB. As primeiras linhas de comentário que aparecem na função podem ser acedidas via *help*.

≫ help fraio

```
 FRAIO   Calcula o raio espectral
         de uma dada matriz
```

Para calcular o raio espectral da matriz A definida anteriormente basta fazer

≫ A=[1 2 3;4 5 6;7 8 9];

≫ fraio(A)

Neste caso as variáveis vp e raio não estão definidas no espaço de trabalho.

O caso de funções com mais de um parâmetro de entrada ou saída é tratado de modo análogo. A função seguinte calcula as raízes de um polinómio do segundo grau.

```
                function [x1,x2]=raizes(a,b,c)
% RAIZES        Calcula as raízes x1, x2
%               da equação ax^2+bx+c=0,
%               dados os valores de a,b e c
                d=sqrt(b*b-4*a*c);
                x1=(-b+d)/(2*a);
                x2=(-b-d)/(2*a);
```

Para se resolver a equação $x^2 + 5x + 6$ basta fazer

≫ [r1,r2]=raizes(1,5,6)

para se obter

$$r1 = -2, \quad r2 = -3$$

3.3 *Input* e *Output*

É possível introduzir um texto ou um valor numérico através do teclado, de uma forma interactiva. Para tal, pode usar-se a função **input** cuja forma é:

variavel_numerica=**input***(' texto')*

ou

variavel_string=**input***(' texto',' s').*

No primeiro caso, aparece no ecrã o texto *texto* e o utilizador deve introduzir um valor numérico que será atribuído à variável *variavel_numerica*. O segundo caso é usado quando se pretende introduzir um valor não numérico.

Como exemplo da utilização desta função, considerem-se as instruções

≫ vn = input (' introduza o número de iterações a efectuar')

e

≫ vs = input (' Pretende continuar? (SIM/NAO)','s')

Se for introduzido o valor 10 no primeiro caso e SIM no segundo, então as instruções anteriores originam $vn = 10$ e $vs =$ SIM.

A função **disp** permite escrever um texto ou valor no ecrã. Assim, o comando

≫ disp(A)

escreve a matriz A no ecrã.

Para escrever uma frase no ecrã deve rodear-se essa frase do símbolo '. Por exemplo,

≫ disp(' Estou a escrever esta frase')

Quando se pretender escrever um conjunto de várias frases, devem separar-se as frases por vígulas e rodear o conjunto com os símbolos [e]. Por exemplo,

≫ disp([' Estou a escrever esta frase ',' há ', ' minutos'])

Para combinar texto e valores numéricos devem converter-se estes últimos a "texto" através da função **num2str**. O comando

≫ disp([' Estou a escrever esta frase ',' há',num2str(n), ' minutos'])

escreverá no ecrã, no caso $n = 10$, a frase

≫ Estou a escrever esta frase há 10 minutos.

Para os alunos mais curiosos e que gostam de escrever programas com um formato de entrada/saída atractivo, aqui fica o convite para invocarem o *help* do MATLAB para conhecerem melhor o uso das instruções **fprintf**, **sprintf**, **fwrite**, **fscanf**, etc.

3.4 Instruções de controle

Como já foi referido, normalmente as instruções em MATLAB são executadas sequencialmente, isto é, depois de uma instrução ter sido executada, é executada a instrução imediatamente a seguir. As seguintes transferências de controle podem ser usadas para alterar a sequência de execução das instruções de uma *script* ou função: ciclos **FOR** e **WHILE**, instruções **IF** e **SWITCH** e ainda **BREAK**, **ERROR** e **RETURN**. Estas instruções são semelhantes às encontradas na maioria das linguagens de programação, pelo que nos limitaremos a descrever a sua forma geral.

Ciclos FOR

A instrução **FOR** permite que um grupo de instruções seja executado repetidas vezes. Tem a forma

for *variavel=expressão*
instruções
end

onde *expressão* é usualmente da forma **m:n** ou **m:i:n**, sendo m o valor inicial, i o valor incremental e n o valor final da *variavel*.

Suponhamos que pretendemos calcular um vector com o valor da função *seno* em 1001 pontos igualmente espaçados do intervalo $[0, 10]$ A maneira mais óbvia de obter este vector é:

```
i=0
for t=0:.01:10
   i=i+1;
   y(i)=sin(t);
end
```

Ciclos WHILE

A instrução **WHILE** permite que um grupo de instruções seja executado um número indefinido de vezes, enquanto uma condição for satisfeita. Tem a forma

while *expressão*
instruções
end

onde *expressão* é uma expressão de relação da forma *e1Re2*, sendo *e1* e *e2* expressões aritméticas e *R* um dos seguintes operadores de relação:

== igual

<= menor ou igual

>= maior ou igual

~= diferente

< menor

> maior

Adicionalmente podem ainda formar-se expressões lógicas mais complicadas combinando elementos lógicos e expressões de relação, com os seguintes operadores lógicos:

& operador e

| operador ou

$\sim$ operador de negação

O exemplo seguinte ilustra o uso de um ciclo **WHILE**. Pretende-se calcular o primeiro inteiro n para o qual $n!$ é um número com 4 dígitos.

```
n=1;
while prod(1:n)< 1000
   n=n+1;
end
n
```

Nota: O tempo de execução de programas em MATLAB pode ser substancialmente reduzido, se se tiver a preocupação de "vectorizar" os algoritmos. Vectorizar significa converter ciclos em operações com vectores ou matrizes. Por exemplo as instruções,

$$t = 0 : .01 : 10;$$

$$y = \sin t;$$

correspondem a uma versão vectorizada do exemplo ilustrativo do ciclo FOR considerado anteriormente.

Instruções IF - ELSEIF - ELSE

A instrução **IF** permite alterar a ordem de execução de uma sequência de instruções, se determinada condição for (ou não) satisfeita. Tem a forma

if *expressão1*
    instruções
elseif *expressão2*
    instruções
elseif *expressão3*
    instruções
    ...
    ...
else
    instruções
end

Cada uma das expressões *expressao1*, *expressao2*, etc é uma expressão de relação da forma *e1Re2*, sendo *e1* e *e2* expressões aritméticas e *R* um dos operadores de relação definidos anteriormente. Podem também combinar-se estes operadores com os operadores lógicos.

Suponhamos que pretendemos definir uma matriz de ordem $m \times n$, cujos elementos diagonais são iguais a 3, os elementos acima da diagonal são 5 e abaixo são 4 e além disso pretendemos obter a soma de todos os elementos da matriz. A *script exemplo.m* definida abaixo pode ser usada para resolver o problema.

```
soma=0;
for i=1:m
   for j=1:n
      if i==j
         a(i,j)=3;
      elseif i<j
         a(i,j)=5;
      else
         a(i,j)=4;
      end
      soma=soma+a(i,j);
   end
end
```

Para se obter a matriz de ordem 4×6 nas condições indicadas, pode fazer-se

```
>> m=4;n=6;
>> exemplo
>> a
>> soma
```

Nota: Este exemplo pretende apenas ilustrar o uso da instrução **IF**. Na verdade, é possível resolver o problema em causa de uma forma muito simples, usando funções já definidas no MATLAB. As seguintes instruções resolvem exactamente o mesmo problema!

```
a=3*eye(6,4)+triu(5*ones(6,4),1)+tril(4*ones(6,4),-1);
soma=sum(sum(a));
```

Instruções SWITCH e CASE

A instrução **SWITCH** executa um grupo de instruções, dependendo do valor de uma variável ou expressão. Tem a forma

switch *expressao*
  case *caso1*
    instruções
  case {*caso2a,caso2b, ...*}
    instruções
      ...
      ...
  otherwise
    instruções
end

Se $n = 23$, qual será o resultado da seguinte *script*?

```
switch rem(n,4)
   case 0
      a=ones(n)
   case {1,2}
      a=eye(n)
   otherwise
      a=zeros(n)
end
```

Instruções BREAK, ERROR e RETURN

Além das instruções de controle já definidas, o MATLAB dispõe de três comandos - **break**, **error** e **return**, para controlar a execução de *scripts* e funções. Uma breve descrição destas instruções é feita de seguida.

A instrução **BREAK** permite sair de um ciclo **FOR** ou **WHILE**.

O comando

error ('*mensagem_de_erro*')

dentro de uma função ou *script*, aborta a execução, exibe a mensagem de erro *mensagem_de_erro* e faz regressar o controle ao teclado.

O comando **RETURN** faz regressar o controle à função invocadora ou ao teclado.

Suponhamos que se pretende utilizar a equação iterativa

$$x_{k+1} = \cos x_k, \ k = 0, 1, \cdots,$$

para obter uma aproximação, com precisão TOL, para a solução da equação $\cos x = x$, usando como aproximação inicial $x_0 = 0$. Este objectivo pode ser atingido, usando a seguinte *script*.

```
x0=0;
dif=1;n=1;
nmax=input('Qual o número máximo de iterações?');
if nmax<= 0
   error('nmax< 0!!!')
end
while dif >TOL
   x1=cos (x0);
   dif=abs(x1-x0);
   n=n+1;
   x0=x1;
   if n>nmax
      disp('o número de iterações ultrapassou o limite estabelecido')
      break
   end
end
disp('acabou esta script')
```

A *script* anterior inclui, como convém a qualquer processo iterativo, um limite máximo para o número de iterações a realizar, limite esse indicado pelo utilizador.

É evidente que o mesmo objectivo pode ser atingido evitando o recurso à instrução BREAK (Como?).

O que acontece se substituir o comando BREAK por RETURN?

4 Gráficos

O MATLAB dispõe de um grande número de facilidades gráficas que podem ser usadas directamente ou acedidas a partir de funções ou *scripts*. Tendo em conta os objectivos específicos destes apontamentos, centraremos a nossa atenção apenas em gráficos básicos 2-D. Todos os pormenores relativos às ferramentas de visualização incluídas nesta versão do MATLAB podem ser obtidos no guia *Using MATLAB Graphics* ou invocando o *help* para graph2d, graph3d, specgraph ou graphics.

O comando mais simples e, talvez o mais útil para produzir gráficos 2-D tem a forma

plot(*Abcissas,Ordenadas,'estilo'*)

onde *Abcissas* e *Ordenadas* são vectores (com a mesma dimensão) que contêm as abcissas e ordenadas de pontos do gráfico e *estilo* é um argumento opcional que especifica o estilo da linha ou ponto a desenhar. Na tabela seguinte estão indicados alguns dos estilos que é possível definir.

	cor		**ponto**		**linha**
y	amarela	**.**	ponto(.)	**-**	contínua
m	magenta	**o**	círculo (o)	**:**	ponteada
c	cião	**x**	cruz(x)	**-.**	'traço-ponto'
r	vermelha	+	mais(+)	**- -**	tracejada
g	verde	*	estrela(*)		
b	azul	**s**	quadrado		
w	branca	**d**	losango($\diamond$)		
k	preta	**v**	triângulo (∇)		

A função **plot** também permite o uso de apenas um vector como argumento. Se $Pontos = [x_1\ x_2 \cdots x_n]$, o comando

plot(*Pontos*)

desenha os pontos de coordenadas $(i, x_i),\ i = 1, \cdots, n$.

Títulos, designação dos eixos, legendas e outras características, podem ser acrescentadas a um dado gráfico, usando as funções **title**, **xlabel**, **ylabel**, **grid**, **text**, **legend**, etc. Estas funções têm a forma seguinte.

title('*título*')	produz um *título* na parte superior do gráfico
xlabel('*nome_x*')	o eixo dos xx é designado por *nome_x*
ylabel('*nome_y*')	o eixo dos yy é designado por *nome_y*
grid	coloca uma quadrícula no gráfico
text(x,y,'*texto_em_x_y*')	escreve o texto *texto_em_x_y* na posição (x, y)
gtext('*texto*')	permite colocar *texto* numa posição a indicar com o rato
legend('*texto1*','*texto2*')	produz uma legenda com *texto1* e *texto2*
legend off	retira legenda

É também possível controlar os limites dos eixos através do comando **axis**. A função **axis** tem várias opções que permitem personalizar os limites, a escala, a orientação, etc, de um gráfico. Escrevendo

$$\textbf{axis}([xmin \quad xmax \quad ymin \quad ymax])$$

os limites passam a ser $xmin$ e $xmax$ para o eixo dos xx e $ymin$ e $ymax$ para o eixo dos yy. Este comando deve aparecer depois do comando **plot**. A função **axis** também aceita palavras chave para controlar os eixos. Por exemplo, **axis square**, **axis equal**, **axis auto**, **axis on**, etc. (Faça *help axis*).

Numa mesma figura podem sobrepôr-se vários gráficos, recorrendo ao comando **hold**. As instruções

hold on

plot(x_1, y_1)

plot(x_2, y_2)

plot(x_3, y_3)

hold off

originam a sobreposição de três gráficos. Este objectivo também pode ser atingido, usando

$$\textbf{plot}(x_1, y_1, x_2, y_2, x_3, y_3)$$

O comando **hold** é especialmente útil quando o conjunto de pontos a desenhar não está disponível todo ao mesmo tempo.

O exemplo seguinte, onde estão representadas as funções sen t, t e $t - \frac{t^3}{3!} + \frac{t^5}{5!}$ ilustra o uso das funções que acabamos de descrever.

```
t=linspace(0,2*pi);
y1=sin(t);
y2=t;
y3=t-(t.^3)/6+(t.^5)/120;
plot(t,y1,'r',t,y2,'b--',t,y3,'go')
axis equal
axis([0 6 -1 5])
grid
xlabel('t')
ylabel('Aproximações para sen(t)')
title('Exemplo 1')
text(3.5,0,'sen(t)')
gtext('Aproximação linear')
gtext('Primeiros 3 termos')
gtext('da série de Taylor')
```

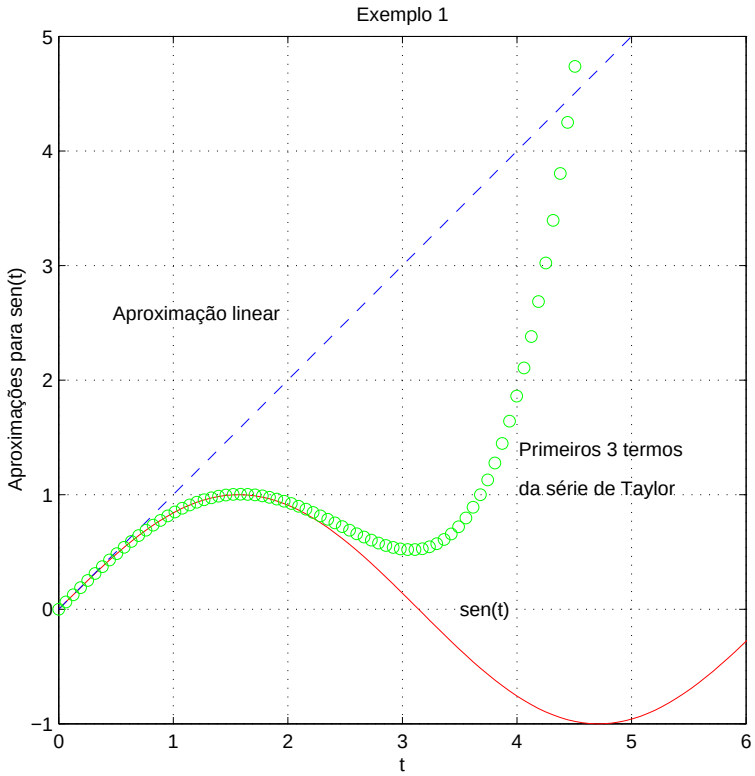


Fig. 1: Figura obtida com a *script* anterior

A função **fplot** permite ao utilizador desenhar uma função pré definida, num dado intervalo. Suponhamos que a função exemplo é dada por

```
function v=exemplo(x)

v=sin(x.^3)
```

A *script* seguinte desenha três gráficos da função sen x^3, no intervalo $[2, 4]$, usando a função **plot** e **fplot**.

```
t=linspace(2,4,50);
y1=sin(t.^3);
subplot(3,1,1)
plot(t,y1,'b')
gtext('Uso da função plot');
subplot(3,1,2)
fplot('exemplo(x)',[2, 4],'r')
gtext('Uso da função fplot - Exemplo 1');
subplot(3,1,3)
fplot('sin(x.^3)',[2, 4],'g')
title('Uso da função fplot - Exemplo 2');
```

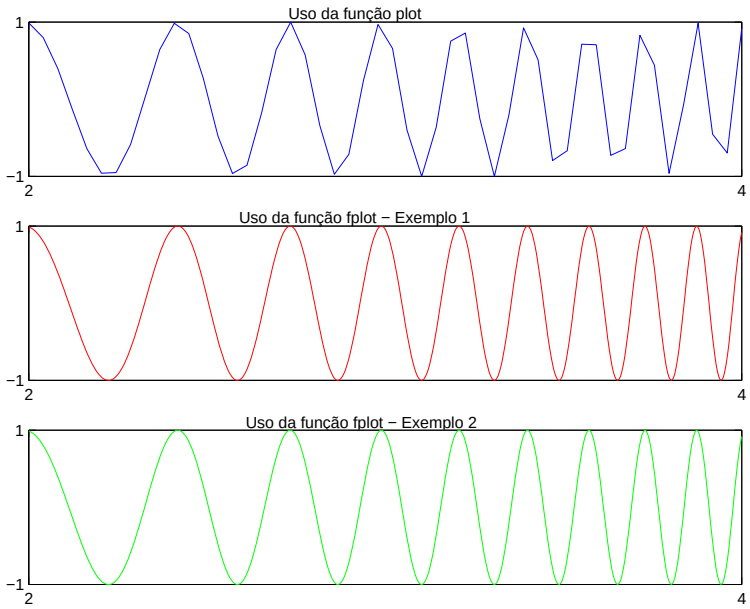


Fig. 2: Uso da função **plot** e **fplot**

5 *Notebook*

O *Notebook* do MATLAB permite aceder às potencialidades numéricas e gráficas do MATLAB a partir de um processador de texto - Microsoft Word. [6]

Usando o *Notebook*, é possível criar um documento contendo texto, comandos MATLAB e os resultados da execução desses comandos.

[6]Para a versão MATLAB 5 é necessário ter instalado o Microsoft Word 95 (Word 7) ou Microsoft Word 97

Nesta secção, iremos apenas referir as instruções básicas para começar a usar o *Notebook*. O guia do utilizador *MATLAB Notebook User's Guide* (Versão 5) contém uma descrição detalhada das capacidades do *Notebook*.

5.1 Criar e editar um *M-book*

Um documento *Notebook* é chamado um *M-book*. Para criar um *M-book*, basta executar o Word, escolher **New** no menu **File** do Word e seleccionar o modelo *M-book*.

O Word cria então um novo documento usando o modelo *M-book* e abre o MATLAB (se este não se encontrar activado). Na barra do menu do Word aparece um novo menu chamado **Notebook** que contém comandos para o *Notebook*.

Para editar um *M-book* já existente, basta escolher o comando **Open** no menu **File** do Word e seleccionar o documento *M-book* desejado.

5.2 Conteúdo de um *M-book*

O texto e os comandos MATLAB que constituem um *M-book* podem ser escritos da mesma forma que qualquer texto de outro documento Word. O estilo do documento pode, por isso, ser alterado de acordo com o gosto pessoal do utilizador.

O *Notebook* usa células de entrada (*input cells*) para definir comandos MATLAB. Estas células de entrada podem ser constituídas por um comando de uma ou mais linhas e por texto.

Definir e Executar Comandos MATLAB

1. Escreva o comando como texto, deixando o cursor no final da linha de texto.

   (Não use a tecla **Enter** ou **Return**.)

2. Seleccione o comando **Evaluate Cell** do menu do **Notebook** ou pressione **Ctrl-Enter**.

O *Notebook* define, então, o comando MATLAB introduzido como uma célula de entrada e cria uma célula de saída (*output cell*) com o resultado da execução do comando. Os caracteres de uma célula de entrada aparecem com a cor verde e os de

uma célula de saída, com a cor azul. O início e fim de células de entrada/saída estão assinalados com os marcadores [e], respectivamente.

O *Notebook* permite ainda definir uma sequência de comandos MATLAB e avaliá-la como um grupo (*cells group*).

Definir e Executar uma Sequência de Comandos MATLAB

1. Escreva a sequência de comandos a executar.
2. Seleccione com o rato todo o conjunto de comandos.
3. Pressione **Ctrl-Enter**.

Exemplo 1

1. Escreva o texto

   disp('Isto é uma célula de entrada')

2. Pressione **Ctrl-Enter** para obter

   [**disp('Isto é uma célula de entrada')**]

   [**Isto é uma célula de entrada**]

Exemplo 2

1. Escreva o texto

   disp('Isto é o primeiro comando')
   disp('de um grupo')
   disp('de três comandos')

2. Seleccione todo o texto com o rato.

3. Pressione **Ctrl-Enter** para obter

   [**disp('Isto é o primeiro comando')**
   disp('de um grupo')
   disp('de três comandos')]

   [**Isto é o primeiro comando**
   de um grupo
   de três comandos]

6 Exercícios

6.1 Iniciação ao MATLAB

1. Defina as matrizes

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & -1 \\ 1 & -5 & 4 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix},$$

$$C = \begin{bmatrix} 1+i & 0 & -1 \\ -i & 1 & i \\ 0 & 1+i & 2-i \end{bmatrix}, \qquad D = \begin{bmatrix} 2i & 0 & 0 \\ 0 & 2i & 0 \\ 0 & 0 & 2i \end{bmatrix}$$

e os vectores

$$x = (1,1,1)^T, \quad y = (1,0,-1)^T, \quad z = (1+i,1,-i)^T, \quad w = (i,i,i)^T.$$

2. Obtenha as seguintes matrizes, usando a definição da matriz A.

$$A_1 = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 1 \\ 1 & 5 & 4 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 4 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & -1 & 2 & 1 \\ 3 & 2 & -1 & 2 \\ 1 & -5 & 4 & 3 \end{bmatrix}, \qquad A_4 = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & -1 \\ 1 & -5 & 4 \\ 1 & 2 & 3 \end{bmatrix},$$

$$A_5 = \begin{bmatrix} 2 & 1 & -1 & 2 \\ 0 & 1 & -1 & 2 \\ 0 & 3 & 2 & -1 \\ 0 & 1 & -5 & 4 \end{bmatrix}, \qquad A_6 = \begin{bmatrix} 1 & -1 & 2 & 0 & 0 \\ 3 & 2 & -1 & 0 & 0 \\ 1 & -5 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix},$$

$$A_7 = \begin{bmatrix} 1 & -1 & 2 & 1 & -1 & 2 \\ 3 & 2 & 1 & 3 & 2 & 1 \\ 1 & 5 & 4 & 1 & 5 & 4 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_8 = \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix}.$$

3. Calcule:

a) $x^T y$, zw^*, xz^T.

b) A^T, C^T, C^*.

c) $\det A$, $\det C$, A^{-1}, C^{-1}.

d) $3A$, $(2+i)C$, $A+B$, $iC+D$, Ax, $z^T C$, $A^T * B$, $A * C^{-1}$, $C * D^*$.

4. Calcule os valores próprios e os respectivos vectores próprios (à esquerda e à direita) das matrizes A e C.

As matrizes A e C são diagonalizáveis? Porquê?

5. Resolva os sistemas

a) $\begin{cases} 2x + y - z = -3 \\ x - 2y + 2z = 1 \\ 2x + y + z = 5 \end{cases}$

b) $\begin{cases} x + y + 2z + 3t = -1 \\ -x - 2y - 3z - 4t = 0 \\ 2x + 3y + 5z + 7t = 1 \\ 3x + 4y + 7z + 10t = 2 \end{cases}$

c) $\begin{cases} x + 2y - 3z - 16t = 4 \\ y + 2z - 3t = 6 \\ -x - y + z + 9t = -2 \\ y - 2z - 7t = 2 \end{cases}$

6. **a)** Obtenha a decomposição LU da matriz

$$\begin{bmatrix} 2 & 2 & 1 & 4 \\ 1 & -3 & 2 & 3 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 2 \end{bmatrix}.$$

b) Seja $b = (5, 2, -1, 2)^T$. Resolva o sistema

$$(\mathrm{LU})\ x = b.$$

7. Para cada um dos vectores x, y e z, calcule $\|.\|_1$, $\|.\|_2$, $\|.\|_\infty$.

8. **a)** Para cada uma das matrizes A e C, calcule $\|.\|_1$, $\|.\|_2$, $\|.\|_\infty$.

b) Verifique a seguinte propriedade:

$$\|A\|_2 = \sqrt{\rho\ (AA^*)}$$

9. Em MATLAB, os polinómios são representados como vectores linha contendo os coeficientes ordenados segundo as potências decrescentes de x. Use *help* polyfun para obter uma lista de funções para operações com polinómios.

a) Defina os polinómios $p(x) = 2x^3 - 3x^2 + 5x - 2$ e $q(x) = x^3 + 1$.

b) Calcule $p * q$, p', p/q, p(5), $q'(10)$.

c) Determine as raízes de p e q.

d) Sobreponha os gráficos dos polinómios p e q.

6.2 Programar em Matlab

1. Dada a matriz A de ordem 4×5, escreva uma *script* para obter a soma de cada coluna de A, usando:

a) a instrução **for**.

b) a função **sum**.

2. Uma equação iterativa para resolver a equação $x^2 - x - 1 = 0$ é dada por

$$x_{r+1} = 1 + \frac{1}{x_r}, \ r = 0, 1, \cdots$$

Dado $x_0 = 2$, escreva uma *script* em MATLAB para resolver esta equação. Admita que a solução tem precisão suficiente se $|x_{r+1} - x_r| < 0.0005$. Compare a solução obtida com a solução exacta.

3. Escreva uma *script* em MATLAB para gerar uma matriz tridiagonal de ordem $n \times n$ que tem o valor d ao longo da diagonal principal e o valor c nas diagonais abaixo e acima da diagonal principal. Esta *script* deve permitir o uso de qualquer valor de d, c e n.

4. Pode provar-se que a série

$$(I - A)^{-1} = I + A + A^2 + A^3 + \cdots,$$

onde A é uma matriz de ordem $n \times n$, converge se os valores próprios de A são todos menores que a unidade. A seguinte matriz satisfaz esta condição se $a + 2b < 1$.

$$\begin{pmatrix} a & b & & & \\ b & a & b & & \\ & b & a & \ddots & \\ & & \ddots & \ddots & b \\ & & & b & a \end{pmatrix}$$

Faça várias experiências com esta matriz, usando diferentes valores de n, a e b. Confirme que a série converge se a condição indicada for satisfeita.

5. Escreva uma *script* que lhe permita, dada uma sequência $a = x_0 < x_1 < \cdots < x_n = b$ de números reais e um número $x \in [a, b]$, determinar o subintervalo $[x_{i-1}, x_i]$; $i = 1, \cdots, n$ a que x pertence.

(Nota: A *script* deve indicar uma mensagem de erro se $x \notin [a, b]$ e, para $x = x_n$, deve indicar o intervalo $[x_{n-1}, x_n]$.)

6. Desenhe uma circunferência de centro no ponto (2,3) e raio 2, usando:

a) a função **plot**

b) a função **fplot**

c) a função **ezplot**

7. Desenhe uma circunferência de centro num ponto C e raio r.

8. Obtenha o seguinte gráfico.

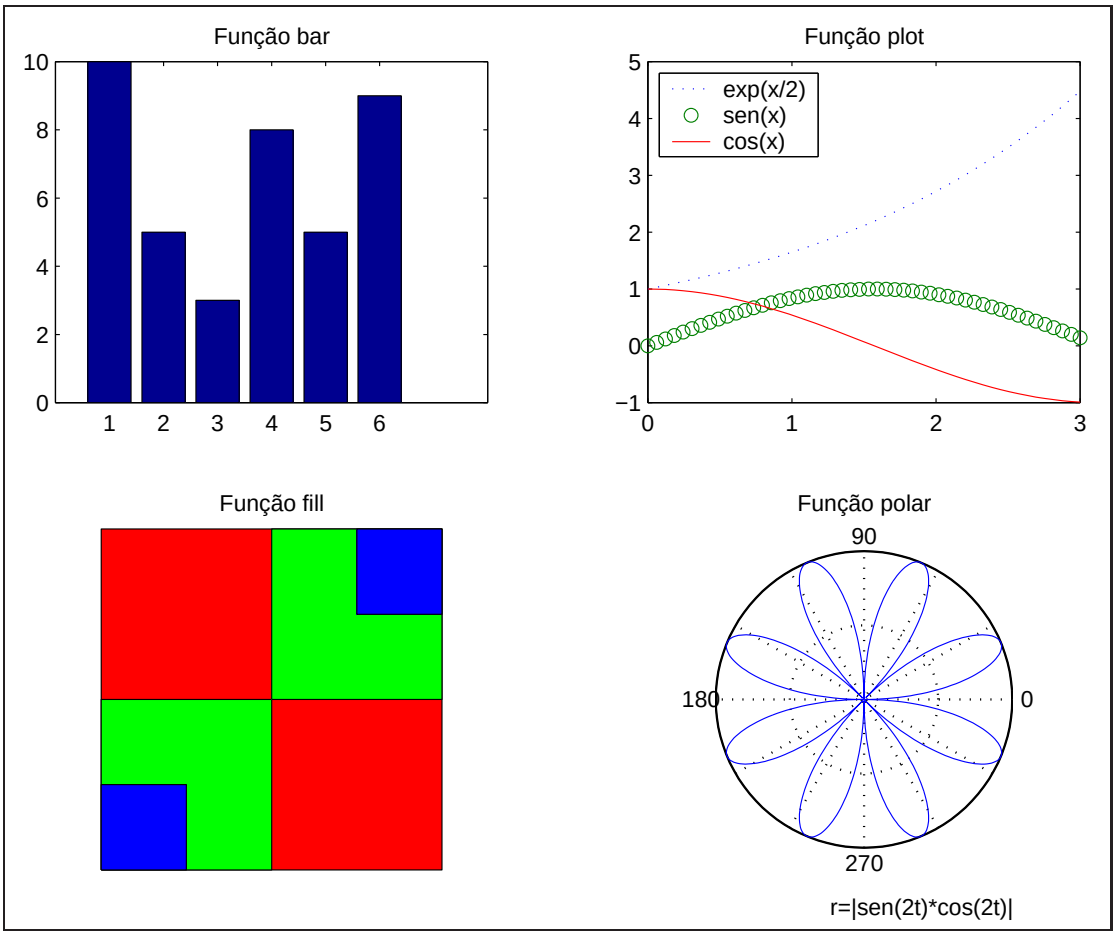


Gráfico do Exercício 8

9. A seguinte *script* define um vector b, usando um ciclo FOR.

```
% Definir o vector b = (b_i) = √i;  i = 1, ···, 1000
% usando um ciclo FOR
%
 clear;
 tic;
 for i=1:1000
   b(i)=sqrt(i);
 end
 t=toc;
 disp('Tempo de execução do ciclo FOR é',num2str(t)]);
```

a) Apresente uma versão "vectorizada" desta *script*.

b) Compare o tempo de execução das duas *scripts*.

(Use as funções **tic** e **toc**)

A Funções matriciais

Algumas das funções matricias incluídas no MATLAB têm um papel muito importante na resolução de problemas que aparecem frequentemente em Análise Numérica. Por este motivo, será feito, nesta secção, um estudo mais detalhado de algumas dessas funções.

A.1 Decomposição LU de uma matriz

A decomposição LU de uma matriz A consiste em escrever a matriz A como um produto de duas matrizes: a matriz L triangular inferior de diagonal unitária e a matriz U triangular superior.[7]

A maior parte dos algoritmos para obter esta decomposição baseiam-se na eliminação gaussiana. No caso em que esta eliminação é feita com escolha parcial de pivot, a matriz L que se obtém pode não ser triangular inferior, mas sim uma permutação de uma matriz triangular inferior. Assim, quando não há lugar à troca de linhas na eliminação de Gauss, obtém-se

$$A = LU,$$

caso contrário, obtém-se

$$PA = LU,$$

onde P é uma matriz de permutação que traduz as trocas de linhas efectuadas durante a eliminação de Gauss, com escolha parcial de pivot.

Do ponto de vista da resolução do sistema de equações lineares, esta factorização permite escrever

$$Ax = b \Rightarrow (LU)x = b \Rightarrow Ly = b \quad \text{e} \quad Ux = y$$

ou

$$Ax = b \Rightarrow PAx = Pb \Rightarrow (LU)x = Pb \Rightarrow Ly = Pb \quad \text{e} \quad Ux = y.$$

Conforme se vê, a determinação da solução x,do sistema $Ax = b$, uma vez obtida a decomposição LU da matriz A, passa pela resolução em sucessão de dois sistemas triangulares: um, inferior, e o outro, superior.[8]

[7] ver, p.e. Maria Raquel Valença, *Métodos Numéricos.*

[8] Relembre que esta forma de resolução de sistemas é particularmente eficiente se tivermos de resolver vários sistemas com a mesma matriz dos coeficientes e diferentes vectores b, i.e. se o sistema for da forma $AX = B$, onde A é uma matriz de ordem $n \times n$ e B é uma matriz de ordem $n \times m$.

No MATLAB, a solução do sistema $Ax = b$ pode ser obtido, como já foi dito, através de

$\gg x = A\backslash b.$

Importa referir que no caso em que a matriz A é uma matriz triangular, esta resolução é feita usando o método de substituição. No caso de uma matriz A quadrada qualquer, a solução do sistema é obtida recorrendo à decomposição LU da matriz A, da forma acabada de descrever. Também o cálculo da inversa e o determinante de uma matriz, usando, respectivamente, as funções **inv** e **det** passam pela decomposição da matriz na forma LU.

No MATLAB, a função que efectua a decomposição LU de uma matriz chama-se **lu**. Esta função pode ter dois parâmetros de saída: a matriz L e a matriz U ou pode ainda ter três parâmetros: a matriz L, a matriz U e a matriz de permutação P. O modo de utilização desta função pode, como de costume, ser conhecido invocando o *help* do MATLAB.

≫ help lu

```
LU    Factors from Gaussian elimination.
      [L,U] = LU(X) stores a upper triangular matrix in U and a
      "psychologically lower triangular matrix", i.e. a product
      of lower triangular and permutation matrices, in L , so
      that X = L*U.

      [L,U,P] = LU(X) returns lower triangular matrix L, upper
      triangular matrix U, and permutation matrix P so that
      P*X = L*U.

      By itself, LU(X) returns the output from LINPACK'S ZGEFA routine.
```

Consideremos a matriz A e o vector b

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \\ 2 & 1 & 1 \end{pmatrix} \qquad \text{e} \qquad b = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}.$$

O uso da função **lu** é ilustrado de seguida

≫ A=[1 1 1;1 -1 2;2 1 1];

≫ b=[4 4 5]';

$\gg$ lu(A)

$$ans = \begin{pmatrix} 2.0000 & 1.0000 & 1.0000 \\ -0.5000 & -1.5000 & 1.5000 \\ -0.5000 & 0.3333 & 1.0000 \end{pmatrix}$$

$\gg$ [l u]=lu(A)

$$l = \begin{pmatrix} 0.5000 & -0.3333 & 1.0000 \\ 0.5000 & 1.0000 & 0 \\ 1.0000 & 0 & 0 \end{pmatrix} \qquad u = \begin{pmatrix} 2.0000 & 1.0000 & 1.0000 \\ 0 & -1.5000 & 1.5000 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

$\gg$ [l u p]=lu(A)

$$l = \begin{pmatrix} 1.0000 & 0 & 0 \\ 0.5000 & 1.0000 & 0 \\ 0.5000 & -0.3333 & 1.0000 \end{pmatrix} \qquad u = \begin{pmatrix} 2.0000 & 1.0000 & 1.0000 \\ 0 & -1.5000 & 1.5000 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

$$p = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Para resolver o sistema $Ax = b$, basta fazer

$\gg y = l\backslash(p * b);$

$\gg x = u\backslash y$

obtendo-se então a solução

$$x = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix},$$

que é precisamente a resposta que se obteria fazendo

$\gg x = A\backslash b$

A.2 Valores e vectores próprios de uma matriz

No MATLAB, a função que permite obter os valores próprios e correspondentes vectores próprios de uma matriz A chama-se **eig**. Esta função tem dois parâmetros de saída: o primeiro é uma matriz cujas colunas são os vectores próprios de A e o segundo é uma matriz diagonal cujos elementos da diagonal são precisamente os valores próprios de A. A posição dos vectores e valores próprios é correspondente, i.e. um valor próprio que esteja na posição (j,j) da segunda matriz, tem como vector próprio associado o vector que está na coluna j da primeira matriz. O uso do *help* esclarece a aplicação da função **eig**

$\gg$ help eig

```
EIG   Eigenvalues and eigenvectors.
      EIG(X) is a vector containing the eigenvalues of a square
      matrix X.

      [V,D] = EIG(X) produces a diagonal matrix D of
      eigenvalues and a full matrix V whose columns are the
      corresponding eigenvectors so that X*V = V*D.

      [V,D] = EIG(X,'nobalance') performs the computation with
      balancing disabled, which sometimes gives more accurate results
      for certain problems with unusual scaling.
```

Consideremos a matriz

$$A = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & -1 & 2 \end{pmatrix}$$

O comando

$\gg [VD] =$eig(A)

produz as matrizes

$$V = \begin{pmatrix} 1.0000 & 0 & 0.0000 \\ 0 & -0.7071 & -0.7071 \\ 0 & -0.7071 & -0.7071 \end{pmatrix} \qquad D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Podemos então concluir que a matriz A tem um valor próprio simples igual a 2 e um valor próprio duplo igual a 1. No entanto, o espaço próprio associado a este valor próprio tem apenas dimensão 1. (Porquê?)

Por definição, λ é valor próprio de uma matriz A, associado ao vector próprio x, se

$$Ax = \lambda x.$$

Teoricamente, este problema pode sempre resolver-se, procurando as raízes do polinómio característico de A, i.e, resolvendo a equação polinomial

$$\det(A - \lambda I) = 0.$$

No MATLAB, a função **poly** calcula o polinómio característico de uma matriz A e a função **roots** calcula as raízes de um polinómio. A aplicação destas duas funções para obter os valores próprios da matriz A definida anteriormente, ilustra a desvantagem do uso deste procedimento para a resolução de um problema de valores próprios e justifica de alguma maneira a razão de se estudarem métodos alternativos para resolver o problema.

```
≫ roots(poly(A))
 ans =

        2.00000000000000
        1.00000000000000  +  0.00000002775143i
        1.00000000000000  -  0.00000002775143i
```

A.3 Normas vectoriais e matriciais

As normas vectoriais e matriciais que se usam com mais frequência são:

Normas vectoriais

$$\begin{aligned} \| \, . \, \|_1 &: \mathbb{C}^n \to \mathbb{R} \\ & x = (x_i) \hookrightarrow \sum_{i=1}^{n} |x_i| \\ \| \, . \, \|_2 &: \mathbb{C}^n \to \mathbb{R} \\ & x = (x_i) \hookrightarrow (\sum_{i=1}^{n} x_i^2)^{1/2} \\ \| \, . \, \|_\infty &: \mathbb{C}^n \to \mathbb{R} \\ & x = (x_i) \hookrightarrow \max_{1 \le i \le n} |x_i| \end{aligned}$$

Normas matriciais (subordinadas às normas vectoriais)

$$\| \, A \, \|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$$

(máximo das somas por colunas).

$$\|A\|_2 = \sqrt{\rho \, (AA^*)}$$

($\rho \, (A) =$ *raio espectral de* $A = \max_i \{|\lambda_i| : \lambda_i$ é valor próprio de $A\}$).

$$\| \, A \, \|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$

(máximo das somas por linhas).

Estas (e outras) normas estão disponíveis no MATLAB, através da função **norm**,

cujo modo de utilização se indica de seguida. Saliente-se que a definição dada para a norma 2 de uma matriz, não sendo igual à indicada no *help* do MATLAB é perfeitamente equivalente (veja Secção 6.1, Exercício 8b).

```
≫ help norm
```

```
NORM   Matrix or vector norm.
       For matrices..
       NORM(X) is the largest singular value of X, max(svd(X)).
       NORM(X,2) is the same as NORM(X).
       NORM(X,1) is the 1-norm of X, the largest column sum,
       = max(sum(abs((X)))).
       NORM(X,inf) is the infinity norm of X, the largest row sum,
       = max(sum(abs((X'))))).
       NORM(X,'inf') is same as NORM(X,inf).
       NORM(X,'fro') is the F-norm, sqrt(sum(diag(X'*X))).
       NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.
       For vectors..
       NORM(V,P) = sum(abs(V)^P)^(1/P).
       NORM(V) = norm(V,2).
       NORM(V,inf) = max(abs(V)).
       NORM(V,-inf) = min(abs(V)).

       See also COND, RCOND, CONDEST, NORMEST.
```

Para a matriz A e o vector x definidos por

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ -2 \\ 4+3i \end{pmatrix},$$

as declarações

```
≫ norm1x=norm(x,1)

≫ norm2x=norm(x,1)

≫ normix=norm(x,inf)

≫ norm1A=norm(A,1)

≫ norm2A=norm(A,1)

≫ normiA=norm(A,inf)
```

resultam em

$$\text{norm1x} = 8,\ \text{norm2x} = 5.4772,\ \text{normix} = 5,$$

$$\text{norm1A} = 18,\ \text{norm2A} = 16.8481,\ \text{normiA} = 24$$

B Manual

B1 General purpose commands

help general

General information	
help	On-line help, display text at command line.
helpwin	Online help, separate window for navigation.
helpdesk	Comprehensive hypertext documentation and troubleshooting.
demo	Run demonstrations.
ver	MATLAB, SIMULINK, and toolbox version information.
whatsnew	Display Readme files.
Readme	What's new in MATLAB.

Managing the workspace	
who	List current variables.
whos	List current variables, long form.
workspace	Display Workspace Browser, a GUI for managing the workspace.
clear	Clear variables and functions from memory.
pack	Consolidate workspace memory.
load	Load workspace variables from disk.
save	Save workspace variables to disk.
quit	Quit MATLAB session.

Managing commands and functions	
what	List MATLAB-specific files in directory.
type	List M-file.
edit	Edit M-file.
open	Open files by extension.
lookfor	Search all M-files for keyword.
which	Locate functions and files.
pcode	Create pre-parsed pseudo-code file (P-file).
inmem	List functions in memory.
mex	Compile MEX-function.

Managing the search path	
path	Get/set search path.
addpath	Add directory to search path.
rmpath	Remove directory from search path.
pathtool	Modify search path.

Controlling the command window	
echo	Echo commands in M-files.
more	Control paged output in command window.
diary	Save text of MATLAB session.
format	Set output format.

Operating system commands	
cd	Change current working directory.
copyfile	Copy a file.
pwd	Show (print) current working directory.
dir	List directory.
delete	Delete file.
getenv	Get environment variable.
mkdir	Make directory.
!	Execute operating system command (see PUNCT).
dos	Execute DOS command and return result.
unix	Execute UNIX command and return result.
vms	Execute VMS DCL command and return result.
web	Open Web browser on site or files.
computer	Computer type.

Debugging M-files	
debug	List debugging commands.
dbstop	Set breakpoint.
dbclear	Remove breakpoint.
dbcont	Continue execution.
dbdown	Change local workspace context.
dbstack	Display function call stack.
dbstatus	List all breakpoints.
dbstep	Execute one or more lines.
dbtype	List M-file with line numbers.
dbup	Change local workspace context.
dbquit	Quit debug mode.
dbmex	Debug MEX-files (UNIX only).

B2 Operators and special characters

help ops

Arithmetic operators	
plus	Plus +
uplus	Unary plus +
minus	Minus -
uminus	Unary minus -
mtimes	Matrix multiply *
times	Array multiply .*
mpower	Matrix power ^
power	Array power .^
mldivide	Backslash or left matrix divide \
mrdivide	Slash or right matrix divide /
ldivide	Left array divide .\
rdivide	Right array divide ./
kron	Kronecker tensor product kron

Relational operators	
eq	Equal ==
ne	Not equal =
lt	Less than <
gt	Greater than >
le	Less than or equal <=
ge	Greater than or equal >=

Logical operators	
and	Logical AND &
or	Logical OR |
not	Logical NOT
xor	Logical EXCLUSIVE OR
any	True if any element of vector is nonzero
all	True if all elements of vector are nonzero

Special characters	
colon	Colon :
paren	Parentheses and subscripting ( )
paren	Brackets [ ]
paren	Braces and subscripting
punct	Decimal point .
punct	Structure field access .
punct	Parent directory ..
punct	Continuation ...
punct	Separator ,
punct	Semicolon ;
punct	Comment %
punct	Invoke operating system command !
punct	Assignment =
punct	Quote '
transpose	Transpose .'
ctranspose	Complex conjugate transpose '
horzcat	Horizontal concatenation [,]
vertcat	Vertical concatenation [;]
subsasgn	Subscripted assignment ( ), ,.
subsref	Subscripted reference ( ), ,.
subsindex	Subscript index

Bitwise operators	
bitand	Bit-wise AND.
bitcmp	Complement bits.
bitor	Bit-wise OR.
bitmax	Maximum floating point integer.
bitxor	Bit-wise XOR.
bitset	Set bit.
bitget	Get bit.
bitshift	Bit-wise shift.

Set operators	
union	Set union.
unique	Set unique.
intersect	Set intersection.
setdiff	Set difference.
setxor	Set exclusive-or.
ismember	True for set member.

B3 Programming language constructs

help matlab/lang

Control flow	
if	Conditionally execute statements.
else	IF statement condition.
elseif	IF statement condition.
end	Terminate scope of FOR, WHILE, SWITCH, TRY and IF statements.
for	Repeat statements a specific number of times.
while	Repeat statements an indefinite number of times.
break	Terminate execution of WHILE or FOR loop.
switch	Switch among several cases based on expression.
case	SWITCH statement case.
otherwise	Default SWITCH statement case.
try	Begin TRY block.
catch	Begin CATCH block.
return	Return to invoking function.

Evaluation and execution	
eval	Execute string with MATLAB expression.
evalc	Evaluate MATLAB expression with capture.
feval	Execute function specified by string.
evalin	Evaluate expression in workspace.
builtin	Execute built-in function from overloaded method.
assignin	Assign variable in workspace.
run	Run script.

Scripts, functions, and variables	
script	About MATLAB scripts and M-files.
function	Add new function.
global	Define global variable.
persistent	Define persistent variable.
mfilename	Name of currently executing M-file.
lists	Comma separated lists.
exist	Check if variables or functions are defined.
isglobal	True for global variables.
mlock	Prevent M-file from being cleared.
munlock	Allow M-file to be cleared.
mislocked	True if M-file cannot be cleared.
precedence	Operator Precedence in MATLAB.

Argument handling	
nargchk	Validate number of input arguments.
nargin	Number of function input arguments.
nargout	Number of function output arguments.
varargin	Variable length input argument list.
varargout	Variable length output argument list.
inputname	Input argument name.

Message display	
error	Display error message and abort function.
warning	Display warning message.
lasterr	Last error message.
lastwarn	Last warning message.
errortrap	Skip error during testing.
disp	Display an array.
display	Overloaded function to display an array.
fprintf	Display formatted message.
sprintf	Write formatted data to a string.

Interactive input	
input	Prompt for user input.
keyboard	Invoke keyboard from M-file.
pause	Wait for user response.
uimenu	Create user interface menu.
uicontrol	Create user interface control.

B4 Elementary matrices and matrix manipulation.

help elmat

Elementary matrices	
zeros	Zeros array.
ones	Ones array.
eye	Identity matrix.
repmat	Replicate and tile array.
rand	Uniformly distributed random numbers.
randn	Normally distributed random numbers.
linspace	Linearly spaced vector.
logspace	Logarithmically spaced vector.
meshgrid	X and Y arrays for 3-D plots.
:	Regularly spaced vector and index into matrix.

Basic array information	
size	Size of matrix.
length	Length of vector.
ndims	Number of dimensions.
disp	Display matrix or text.
isempty	True for empty matrix.
isequal	True if arrays are identical.
isnumeric	True for numeric arrays.
islogical	True for logical array.
logical	Convert numeric values to logical.

Matrix manipulation	
reshape	Change size.
diag	Diagonal matrices and diagonals of matrix.
blkdiag	Block diagonal concatenation.
tril	Extract lower triangular part.
triu	Extract upper triangular part.
fliplr	Flip matrix in left/right direction.
flipud	Flip matrix in up/down direction.
flipdim	Flip matrix along specified dimension.
rot90	Rotate matrix 90 degrees.
:	Regularly spaced vector and index into matrix.
find	Find indices of nonzero elements.
end	Last index.
sub2ind	Linear index from multiple subscripts.
ind2sub	Multiple subscripts from linear index.

Special variables and constants	
ans	Most recent answer.
eps	Floating point relative accuracy.
realmax	Largest positive floating point number.
realmin	Smallest positive floating point number.
pi	3.1415926535897....
i, j	Imaginary unit.
inf	Infinity.
NaN	Not-a-Number.
isnan	True for Not-a-Number.
isinf	True for infinite elements.
isfinite	True for finite elements.
flops	Floating point operation count.
why	Succinct answer.

Specialized matrices	
compan	Companion matrix.
gallery	Higham test matrices.
hadamard	Hadamard matrix.
hankel	Hankel matrix.
hilb	Hilbert matrix.
invhilb	Inverse Hilbert matrix.
magic	Magic square.
pascal	Pascal matrix.
rosser	Classic symmetric eigenvalue test problem.
toeplitz	Toeplitz matrix.
vander	Vandermonde matrix.
wilkinson	Wilkinson's eigenvalue test matrix.

B5 Elementary math functions

help elfun

Trigonometric	
sin	Sine.
sinh	Hyperbolic sine.
asin	Inverse sine.
asinh	Inverse hyperbolic sine.
cos	Cosine.
cosh	Hyperbolic cosine.
acos	Inverse cosine.
acosh	Inverse hyperbolic cosine.
tan	Tangent.
tanh	Hyperbolic tangent.
atan	Inverse tangent.
atan2	Four quadrant inverse tangent.
atanh	Inverse hyperbolic tangent.
sec	Secant.
sech	Hyperbolic secant.
asec	Inverse secant.
asech	Inverse hyperbolic secant.
csc	Cosecant.
csch	Hyperbolic cosecant.
acsc	Inverse cosecant.
acsch	Inverse hyperbolic cosecant.
cot	Cotangent.
coth	Hyperbolic cotangent.
acot	Inverse cotangent.
acoth	Inverse hyperbolic cotangent.

Exponential	
exp	Exponential.
log	Natural logarithm.
log10	Common (base 10) logarithm.
log2	Base 2 logarithm and dissect floating point number.
pow2	Base 2 power and scale floating point number.
sqrt	Square root.
nextpow2	Next higher power of 2.

Complex	
abs	Absolute value.
angle	Phase angle.
complex	Construct complex data from real and imaginary parts.
conj	Complex conjugate.
imag	Complex imaginary part.
real	Complex real part.
unwrap	Unwrap phase angle.
isreal	True for real array.
cplxpair	Sort numbers into complex conjugate pairs.

Rounding and remainder	
fix	Round towards zero.
floor	Round towards minus infinity.
ceil	Round towards plus infinity.
round	Round towards nearest integer.
mod	Modulus (signed remainder after division).
rem	Remainder after division.
sign	Signum.

B6 Specialized math functions

help specfun

Specialized math functions	
airy	Airy functions.
besselj	Bessel function of the first kind
bessely	Bessel function of the second kind
besselh	Bessel functions of the third kind (Hankel function)
besseli	Modified Bessel function of the first kind
besselk	Modified Bessel function of the second kind
beta	Beta function
betainc	Incomplete beta function
betaln	Logarithm of beta function
ellipj	Jacobi elliptic functions
ellipke	Complete elliptic integral
erf	Error function
erfc	Complementary error function
erfcx	Scaled complementary error function
erfinv	Inverse error function
expint	Exponential integral function
gamma	Gamma function
gammainc	Incomplete gamma function
gammaln	Logarithm of gamma function
legendre	Associated Legendre function
cross	Vector cross product

Number theoretic functions	
factor	Prime factors
isprime	True for prime numbers
primes	Generate list of prime numbers
gcd	Greatest common divisor
lcm	Least common multiple
rat	Rational approximation
rats	Rational output
perms	All possible permutations
nchoosek	All combinations of N elements taken K at a time
factorial	Factorial function

Coordinate transforms	
cart2sph	Transform Cartesian to spherical coordinates
cart2pol	Transform Cartesian to polar coordinates
pol2cart	Transform polar to Cartesian coordinates
sph2cart	Transform spherical to Cartesian coordinates
hsv2rgb	Convert hue-saturation-value colors to red-green-blue
rgb2hsv	Convert red-green-blue colors to hue-saturation-value

B7 Matrix functions - numerical linear algebra.

help matfun

Matrix analysis	
norm	Matrix or vector norm
normest	Estimate the matrix-norm
rank	Matrix rank
det	Determinant
trace	Sum of diagonal elements
null	Null space
orth	Orthogonalization
rref	Reduced row echelon form
subspace	Angle between two subspaces

Linear equations	
\ and /	Linear equation solution; use "help slash"
inv	Matrix inverse
cond	Condition number with respect to inversion
condest	1-norm condition number estimate
chol	Cholesky factorization
cholinc	Incomplete Cholesky factorization
lu	LU factorization
luinc	Incomplete LU factorization
qr	Orthogonal-triangular decomposition
lsqnonneg	Linear least squares with nonnegativity constraints
pinv	Pseudoinverse
lscov	Least squares with known covariance.

Eigenvalues and singular values	
eig	Eigenvalues and eigenvectors
svd	Singular value decomposition
gsvd	Generalized singular value decomposition
eigs	A few eigenvalues
svds	A few singular values
poly	Characteristic polynomial
polyeig	Polynomial eigenvalue problem
condeig	Condition number with respect to eigenvalues
hess	Hessenberg form
qz	QZ factorization for generalized eigenvalues
schur	Schur decomposition

Matrix functions	
expm	Matrix exponential
logm	Matrix logarithm
sqrtm	Matrix square root
funm	Evaluate general matrix function

Factorization utilities	
qrdelete	Delete column from QR factorization
qrinsert	Insert column in QR factorization
rsf2csf	Real block diagonal form to complex diagonal form
cdf2rdf	Complex diagonal form to real block diagonal form
balance	Diagonal scaling to improve eigenvalue accuracy
planerot	Given's plane rotation
cholupdate	rank 1 update to Cholesky factorization
qrupdate	rank 1 update to QR factorization

B8 Data analysis and Fourier transforms

help datafun

Finite differences	
diff	Difference and approximate derivative
gradient	Approximate gradient
del2	Discrete Laplacian

Correlations	
corrcoef	Correlation coefficients
cov	Covariance matrix
subspace	Angle between subspaces

Basic operations	
max	Largest component
min	Smallest component
mean	Average or mean value
median	Median value
std	Standard deviation
var	Variance
sort	Sort in ascending order
sortrows	Sort rows in ascending order
sum	Sum of elements
prod	Product of elements
hist	Histogram
histc	Histogram count
trapz	Trapezoidal numerical integration
cumsum	Cumulative sum of elements
cumprod	Cumulative product of elements
cumtrapz	Cumulative trapezoidal numerical integration

Filtering and convolution	
filter	One-dimensional digital filter
filter2	Two-dimensional digital filter
conv	Convolution and polynomial multiplication
conv2	Two-dimensional convolution
convn	N-dimensional convolution
deconv	Deconvolution and polynomial division
detrend	Linear trend removal

Fourier transforms	
fft	Discrete Fourier transform
fft2	Two-dimensional discrete Fourier t.
fftn	N-dimensional discrete Fourier t.
ifft	Inverse discrete Fourier t.
ifft2	Two-dimensional inverse discrete Fourier t.
ifftn	N-dimensional inverse discrete Fourier t.
fftshift	Shift DC component to center of spectrum
ifftshift	Inverse FFTSHIFT

Sound and audio	
sound	Play vector as sound
soundsc	Autoscale and play vector as sound
speak	Convert input string to speech (Macintosh only)
recordsound	Record sound (Macintosh only)
soundcap	Sound capabilities (Macintosh)
mu2lin	Convert mu-law encoding to linear signal
lin2mu	Convert linear signal to mu-law encoding

Audio file inport/export	
auwrite	Write NeXT/SUN (".au") sound file
auread	Read NeXT/SUN (".au") sound file
wavwrite	Write Microsoft WAVE (".wav") sound file
wavread	Read Microsoft WAVE (".wav") sound file
readsnd	Read SND resources and files (Macintosh only)
writesnd	Write SND resources and files (Macintosh only)

B9 Interpolation and polynomials

help polyfun

Data interpolation	
interp1	1-D interpolation (table lookup)
interp1q	Quick 1-D linear interpolation
interpft	1-D interpolation using FFT method
interp2	2-D interpolation (table lookup)
interp3	3-D interpolation (table lookup)
interpn	N-D interpolation (table lookup)
griddata	Data gridding and surface fitting

Spline interpolation	
spline	Cubic spline interpolation
ppval	Evaluate piecewise polynomial

Geometric analysis	
delaunay	Delaunay triangulation
dsearch	Search Delaunay triagulation for nearest point
tsearch	Closest triangle search
convhull	Convex hull
voronoi	Voronoi diagram
inpolygon	True for points inside polygonal region
rectint	Rectangle intersection area
polyarea	Area of polygon

Polynomials	
roots	Find polynomial roots
poly	Convert roots to polynomial
polyval	Evaluate polynomial
polyvalm	Evaluate polynomial with matrix argument
residue	Partial-fraction expansion (residues)
polyfit	Fit polynomial to data
polyder	Differentiate polynomial
conv	Multiply polynomials
deconv	Divide polynomials

B10 Function functions and ODE solvers

help funfun

Optimization and root finding	
fminbnd	Scalar bounded nonlinear function minimization
fminsearch	Multidimensional unconstrained nonlinear minimization, by Nelder-Mead direct search method
fzero	Scalar nonlinear zero finding

Optimization Option handling	
optimset	Create or alter optimization OPTIONS structure
optimget	Get optimization parameters from OPTIONS structure

Numerical integration (quadrature)	
quad	Numerically evaluate integral, low order method
quad8	Numerically evaluate integral, higher order method
dblquad	Numerically evaluate double integral

Plotting	
ezplot	Easy to use function plotter
fplot	Plot function

Inline function object	
inline	Construct INLINE function object
argnames	Argument names
formula	Function formula
char	Convert INLINE object to character array

Utilities	
vectorize	Vectorize string expression or INLINE function object

Ordinary differential equation solvers (If unsure about stiffness, try ODE45 first, then ODE15S)	
ode45	Solve non-stiff differential equations, medium order method
ode23	Solve non-stiff differential equations, low order method
ode113	Solve non-stiff differential equations, variable order method
ode23t	Solve moderately stiff differential equations, trapezoidal rule
ode15s	Solve stiff differential equations, variable order method
ode23s	Solve stiff differential equations, low order method
ode23tb	Solve stiff differential equations, low order method
odefile	ODE file syntax

ODE Option handling	
odeset	Create/alter ODE OPTIONS structure
odeget	Get ODE OPTIONS parameters

ODE output functions	
odeplot	Time series ODE output function
odephas2	2-D phase plane ODE output function
odephas3	3-D phase plane ODE output function
odeprint	Command window printing ODE output function

B11 Sparse matrices

help sparfun

Elementary sparse matrices	
speye	Sparse identity matrix
sprand	Sparse uniformly distributed random matrix
sprandn	Sparse normally distributed random matrix
sprandsym	Sparse random symmetric matrix
spdiags	Sparse matrix formed from diagonals

Full to sparse conversion	
sparse	Create sparse matrix
full	Convert sparse matrix to full matrix
find	Find indices of nonzero elements
spconvert	Import from sparse matrix external format

Working with sparse matrices	
nnz	Number of nonzero matrix elements
nonzeros	Nonzero matrix elements
nzmax	Amount of storage allocated for nonzero matrix elements
spones	Replace nonzero sparse matrix elements with ones
spalloc	Allocate space for sparse matrix
issparse	True for sparse matrix
spfun	Apply function to nonzero matrix elements
spy	Visualize sparsity pattern

Reordering algorithms	
colmmd	Column minimum degree permutation
symmmd	Symmetric minimum degree permutation
symrcm	Symmetric reverse Cuthill-McKee permutation
colperm	Column permutation
randperm	Random permutation
dmperm	Dulmage-Mendelsohn permutation

Linear algebra	
eigs	A few eigenvalues
svds	A few singular values
luinc	Incomplete LU factorization
cholinc	Incomplete Cholesky factorization
normest	Estimate the matrix 2-norm
condest	1-norm condition number estimate
sprank	Structural rank

Linear Equations (iterative methods)	
pcg	Preconditioned Conjugate Gradients Method
bicg	BiConjugate Gradients Method
bicgstab	BiConjugate Gradients Stabilized Method
cgs	Conjugate Gradients Squared Method
gmres	Generalized Minimum Residual Method
qmr	Quasi-Minimal Residual Method

Operations on graphs (trees)	
treelayout	Lay out tree or forest
treeplot	Plot picture of tree
etree	Elimination tree
etreeplot	Plot elimination tree
gplot	Plot graph, as in "graph theory"

Miscellaneous	
symbfact	Symbolic factorization analysis
spparms	Set parameters for sparse matrix routines
spaugment	Form least squares augmented system

B12 Two dimensional graphs

help graph2D

Elementary X-Y graphs	
plot	Linear plot
loglog	Log-log scale plot
semilogx	Semi-log scale plot
semilogy	Semi-log scale plot
polar	Polar coordinate plot
plotyy	Graphs with y tick labels on the left and right

Axis control	
axis	Control axis scaling and appearance
zoom	Zoom in and out on a 2-D plot
grid	Grid lines
box	Axis box
hold	Hold current graph
axes	Create axes in arbitrary positions
subplot	Create axes in tiled positions

Graph annotation	
plotedit	Tools for editing and annotating plots
legend	Graph legend
title	Graph title
xlabel	X-axis label
ylabel	Y-axis label
texlabel	Produces TeX format from a character string
text	Text annotation
gtext	Place text with mouse

Hardcopy and printing	
print	Print graph or SIMULINK system; or save graph to M-file
printopt	Printer defaults
orient	Set paper orientation

B13 Three dimensional graphs

help graph3D

Elementary 3-D plots	
plot3	Plot lines and points in 3-D space
mesh	3-D mesh surface
surf	3-D colored surface
fill3	Filled 3-D polygons.

Color control	
colormap	Color look-up table
caxis	Pseudocolor axis scaling
shading	Color shading mode
hidden	Mesh hidden line removal mode
brighten	Brighten or darken color map
colordef	Set color defaults
graymon	Set graphics defaults for gray-scale monitors

Lighting	
surfl	3-D shaded surface with lighting
lighting	Lighting mode
material	Material reflectance mode
specular	Specular reflectance
diffuse	Diffuse reflectance
surfnorm	Surface normals

Color maps	
hsv	Hue-saturation-value color map
hot	Black-red-yellow-white color map
gray	Linear gray-scale color map
bone	Gray-scale with tinge of blue color map
copper	Linear copper-tone color map
pink	Pastel shades of pink color map
white	All white color map
flag	Alternating red, white, blue, and black color map
lines	Color map with the line colors
colorcube	Enhanced color-cube color map
vga	Windows colormap for 16 colors
jet	Variant of HSV
prism	Prism color map
cool	Shades of cyan and magenta color map
autumn	Shades of red and yellow color map
spring	Shades of magenta and yellow color map
winter	Shades of blue and green color map
summer	Shades of green and yellow color map

Axis control	
axis	Control axis scaling and appearance
zoom	Zoom in and out on a 2-D plot
grid	Grid lines
box	Axis box
hold	Hold current graph
axes	Create axes in arbitrary positions
subplot	Create axes in tiled positions
daspect	Data aspect ratio
pbaspect	Plot box aspect ratio
xlim	X limits
ylim	Y limits
zlim	Z limits

Viewpoint control	
view	3-D graph viewpoint specification
viewmtx	View transformation matrix
rotate3d	Interactively rotate view of 3-D plot

Camera control	
campos	Camera position
camtarget	Camera target
camva	Camera view angle
camup	Camera up vector
camproj	Camera projection

High level camera control	
camorbit	Orbit camera
campan	Pan camera
camdolly	Dolly camera
camzoom	Zoom camera
camroll	Roll camera
camlookat	Move camera and target to view specified objects
cameramenu	Interactively manipulate camera

High level light control	
camlight	Creates or sets position of a light
lightangle	Spherical position of a light

Graph annotation	
title	Graph title
xlabel	X-axis label
ylabel	Y-axis label
zlabel	Z-axis label
colorbar	Display color bar (color scale)
text	Text annotation
gtext	Mouse placement of text
plotedit	Experimental graph editing and annotation tools

Hardcopy and printing	
print	Print graph or SIMULINK system; or save graph to M-file
printopt	Printer defaults
orient	Set paper orientation
vrml	Save graphics to VRML 2.0 file

B14 Specialized graphs

help specgraph

Specialized 2-D graphs	
area	Filled area plot
bar	Bar graph
barh	Horizontal bar graph
bar3	3-D bar graph
bar3h	Horizontal 3-D bar graph
comet	Comet-like trajectory
errorbar	Error bar plot
ezplot	Easy to use function plotter
ezpolar	Easy to use polar coordinate plotter
feather	Feather plot
fill	Filled 2-D polygons
fplot	Plot function
hist	Histogram
pareto	Pareto chart
pie	Pie chart
pie3	3-D pie chart
plotmatrix	Scatter plot matrix
ribbon	Draw 2-D lines as ribbons in 3-D
scatter	Scatter plot
stem	Discrete sequence or "stem"plot
stairs	Stairstep plot

Contour and 2-1/2 D graphs	
contour	Contour plot
contourf	Filled contour plot
contour3	3-D Contour plot
clabel	Contour plot elevation labels
ezcontour	Easy to use contour plotter
ezcontourf	Easy to use filled contour plotter
pcolor	Pseudocolor (checkerboard) plot
voronoi	Voronoi diagram

Specialized 3-D graphs	
comet3	3-D comet-like trajectories
ezgraph3	General purpose surface plotter
ezmesh	Easy to use 3-D mesh plotter
ezmeshc	Easy to use combination mesh/contour plotter
ezplot3	Easy to use 3-D parametric curve plotter
ezsurf	Easy to use 3-D colored surface plotter
ezsurfc	Easy to use combination surf/contour plotter
meshc	Combination mesh/contour plot
meshz	3-D mesh with curtain
scatter3	3-D scatter plot
stem3	3-D stem plot
surfc	Combination surf/contour plot
trisurf	Triangular surface plot
trimesh	Triangular mesh plot
waterfall	Waterfall plot

Volume and vector visualization	
vissuite	Visualization suite
isosurface	Isosurface extractor
isonormals	Isosurface normals
isocaps	Isosurface end caps
contourslice	Contours in slice planes
slice	Volumetric slice plot
streamline	Stream lines from 2D or 3D vector data
stream3	3D stream lines
stream2	2D stream lines
quiver3	3D quiver plot
quiver	2D quiver plot
coneplot	3D cone plot
subvolume	Extract subset of volume dataset
reducevolume	Reduce volume dataset
smooth3	Smooth 3D data
reducepatch	Reduce number of patch faces
shrinkfaces	Reduce size of patch faces

Images display and file I/O	
image	Display image
imagesc	Scale data and display as image
colormap	Color look-up table
gray	Linear gray-scale color map
contrast	Gray scale color map to enhance image contrast
brighten	Brighten or darken color map
colorbar	Display color bar (color scale)
imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Information about graphics file

Movies and animation	
capture	Screen capture of current figure
moviein	Initialize movie frame memory
getframe	Get movie frame
movie	Play recorded movie frames
qtwrite	Translate movie into QuickTime format (Macintosh)
rotate	Rotate object about specified orgin and direction
frame2im	Convert movie frame to indexed image
im2frame	Convert index image into movie format

Color related functions	
spinmap	Spin color map
rgbplot	Plot color map
colstyle	Parse color and style from string
ind2rgb	Convert indexed image to RGB image

Solid modeling	
cylinder	Generate cylinder
sphere	Generate sphere
patch	Create patch
surf2patch	Convert surface data to patch data

B15 Handle Graphics

help graphics

Figure window creation and control	
figure	Create figure window
gcf	Get handle to current figure
clf	Clear current figure
shg	Show graph window
close	Close figure
refresh	Refresh figure

Axis creation and control	
subplot	Create axes in tiled positions
axes	Create axes in arbitrary positions
gca	Get handle to current axes
cla	Clear current axes
axis	Control axis scaling and appearance
box	Axis box
caxis	Control pseudocolor axis scaling
hold	Hold current graph
ishold	Return hold state

Handle Graphics objects	
figure	Create figure window
axes	Create axes
line	Create line
text	Create text
patch	Create patch
rectangle	Create rectangle, rounded-rectangle, or ellipse
surface	Create surface
image	Create image
light	Create light
uicontrol	Create user interface control
uimenu	Create user interface menu
uicontextmenu	Create user interface context menu

Handle Graphics operations	
set	Set object properties
get	Get object properties
reset	Reset object properties
delete	Delete object
gco	Get handle to current object
gcbo	Get handle to current callback object
gcbf	Get handle to current callback figure
drawnow	Flush pending graphics events
findobj	Find objects with specified property values
copyobj	Make copy of graphics object and its children
isappdata	Check if application-defined data exists
getappdata	Get value of application-defined data
setappdata	Set application-defined data
rmappdata	Remove application-defined data

Hardcopy and printing	
print	Print graph or SIMULINK system; or save graph to M-file
printopt	Printer defaults
orient	Set paper orientation

Utilities	
closereq	Figure close request function
newplot	M-file preamble for NextPlot property
ishandle	True for graphics handles

ActiveX Client Functions (PC Only)	
actxcontrol	Create an ActiveX control
actxserver	Create an ActiveX server

GUI functions	
uicontrol	Create user interface control
uimenu	Create user interface menu
ginput	Graphical input from mouse
dragrect	Drag XOR rectangles with mouse
rbbox	Rubberband box
selectmoveresize	Interactively select, move, resize, or copy objects
waitforbuttonpress	Wait for key/buttonpress over figure
waitfor	Block execution and wait for event
uiwait	Block execution and wait for resume
uiresume	Resume execution of blocked M-file
uistack	Control stacking order of objects
uisuspend	Suspend the interactive state of a figure
uirestore	Restore the interactive state of a figure

GUI design tools	
guide	Design GUI
align	Align uicontrols and axes
cbedit	Edit callback
menuedit	Edit menu
propedit	Edit property

Dialog boxes	
dialog	Create dialog figure
axlimdlg	Axes limits dialog box
errordlg	Error dialog box
helpdlg	Help dialog box
inputdlg	Input dialog box
listdlg	List selection dialog box
menu	Generate menu of choices for user input
msgbox	Message box
questdlg	Question dialog box
warndlg	Warning dialog box
uigetfile	Standard open file dialog box
uiputfile	Standard save file dialog box
uisetcolor	Color selection dialog box
uisetfont	Font selection dialog box
pagedlg	Page position dialog box
pagesetupdlg	Page setup dialog
printdlg	Print dialog box
waitbar	Display wait bar
printpreview	Display preview of figure to be printed

Menu utilities	
makemenu	Create menu structure
menubar	Computer dependent default setting for MenuBar property
umtoggle	Toggle "checked"status of uimenu object
winmenu	Create submenu for "Window"menu item

Toolbar button group utilities	
btngroup	Create toolbar button group
btnstate	Query state of toolbar button group
btnpress	Button press manager for toolbar button group
btndown	Depress button in toolbar button group
btnup	Raise button in toolbar button group

Preferences	
addpref	Add preference
getpref	Get preference
rmpref	Remove preference
setpref	Set preference

Miscellaneous utilities	
allchild	Get all object children
findall	Find all objects
hidegui	Hide/unhide GUI
edtext	Interactive editing of axes text objects
findfigs	Find figures positioned off screen
getstatus	Get status text string in figure
setstatus	Set status text string in figure
popupstr	Get popup menu selection string
remapfig	Transform figure objects' positions
setptr	Set figure pointer
getptr	Get figure pointer
overobj	Get handle of object the pointer is over
uiclearmode	Clears the currently active interactive mode

B17 Character strings

help strfun

General	
char	Create character array (string)
double	Convert string to numeric character codes
cellstr	Create cell array of strings from character array
blanks	String of blanks
deblank	Remove trailing blanks
eval	Execute string with MATLAB expression

String tests	
ischar	True for character array (string)
iscellstr	True for cell array of strings
isletter	True for letters of the alphabet
isspace	True for white space characters

String operations	
strcat	Concatenate strings
strvcat	Vertically concatenate strings
strcmp	Compare strings
strncmp	Compare first N characters of strings
strcmpi	Compare strings ignoring case
strncmpi	Compare first N characters of strings ignoring case
findstr	Find one string within another
strjust	Justify character array
strmatch	Find possible matches for string
strrep	Replace string with another
strtok	Find token in string
upper	Convert string to uppercase
lower	Convert string to lowercase

String to number conversion	
num2str	Convert number to string
int2str	Convert integer to string
mat2str	Convert matrix to eval'able string
str2double	Convert string to double precision value
str2num	Convert string matrix to numeric array
sprintf	Write formatted data to string
sscanf	Read string under format control

Base number conversion	
hex2num	Convert IEEE hexadecimal to double precision number
hex2dec	Convert hexadecimal string to decimal integer
dec2hex	Convert decimal integer to hexadecimal string
bin2dec	Convert binary string to decimal integer
dec2bin	Convert decimal integer to binary string
base2dec	Convert base B string to decimal integer
dec2base	Convert decimal integer to base B string

B18 File input/output

help iofun

File opening and closing	
fopen	Open file
fclose	Close file

Binary file I/O	
fread	Read binary data from file
fwrite	Write binary data to file

Formatted file I/O	
textread	Read formatted data from text file
fscanf	Read formatted data from file
fprintf	Write formatted data to file
fgetl	Read line from file, discard newline character
fgets	Read line from file, keep newline character
input	Prompt for user input

String conversion	
sprintf	Write formatted data to string
sscanf	Read string under format control

File positioning	
ferror	Inquire file error status
feof	Test for end-of-file
fseek	Set file position indicator
ftell	Get file position indicator
frewind	Rewind file

File name handling	
matlabroot	Root directory of MATLAB installation
filesep	Directory separator for this platform
pathsep	Path separator for this platform
mexext	MEX filename extension for this platform
fullfile	Build full filename from parts
fileparts	Filename parts
partialpath	Partial pathnames
tempdir	Get temporary directory
tempname	Get temporary file
prefdir	Preference directory name

File import/export functions	
load	Load workspace from MAT-file
dlmread	Read ASCII delimited file
dlmwrite	Write ASCII delimited file
wk1read	Read spreadsheet (WK1) file
wk1write	Write spreadsheet (WK1) file

HDF library interface help	
hdf	MEX-file interface to the HDF library
hdfan	MATLAB Gateway to HDF multifile annotation interface
hdfdf24	MATLAB Gateway to HDF raster image interface
hdfdfr8	MATLAB Gateway to HDF 8-bit raster image interface
hdfh	MATLAB Gateway to HDF H interface
hdfhd	MATLAB Gateway to HDF HD interface
hdfhe	MATLAB Gateway to HDF HE interface
hdfml	MATLAB-HDF gateway utilities
hdfsd	MATLAB Gateway to HDF multifile scientific dataset interface
hdfv	MATLAB Gateway to HDF V (Vgroup) interface
hdfvf	MATLAB Gateway to HDF VF (Vdata) interface
hdfvh	MATLAB Gateway to HDF VH (Vdata) interface
hdfvs	MATLAB Gateway to HDF VS (Vdata) interface

HDF-EOS library interface help	
hdfgd	MATLAB Gateway to HDF-EOS grid interface
hdfpt	MATLAB Gateway to HDF-EOS point interface
hdfsw	MATLAB Gateway to HDF-EOS swath interface

Image file import/export	
imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Return information about graphics file

Audio file import/export	
auwrite	Write NeXT/SUN (".au") sound file
auread	Read NeXT/SUN (".au") sound file
wavwrite	Write Microsoft WAVE (".wav") sound file
wavread	Read Microsoft WAVE (".wav") sound file

Command window I/O	
clc	Clear command window
home	Send cursor home
disp	Display array
input	Prompt for user input
pause	Wait for user response

FIG file support for plotedit and printframes	
hgload	Loads a Handle Graphics object from a file
hgsave	Saves an HG object heirarchy to a file

Utilites	
str2rng	Convert spreadsheet range string to numeric array
wk1const	WK1 record type definitions
wk1wrec	Write a WK1 record header

Obsolete functions	
csvread	Read a comma separated value file
csvwrite	Write a comma separated value file

B19 Time and dates

help timefun

Current date and time	
now	Current date and time as date number
date	Current date as date string
clock	Current date and time as date vector

Basic functions	
datenum	Serial date number
datestr	String representation of date
datevec	Date components

Date functions	
calendar	Calendar
weekday	Day of week
eomday	End of month
datetick	Date formatted tick labels

Timing functions	
cputime	CPU time in seconds
tic	Start stopwatch timer
toc	Stop stopwatch timer
etime	Elapsed time
pause	Wait in seconds

B20 Data types and structures

help datatypes

Data types (classes)	
double	Convert to double precision
sparse	Create sparse matrix
char	Create character array (string)
cell	Create cell array
struct	Create or convert to structure array
single	Convert to single precision
uint8	Convert to unsigned 8-bit integer
uint16	Convert to unsigned 16-bit integer
uint32	Convert to unsigned 32-bit integer
int8	Convert to signed 8-bit integer
int16	Convert to signed 16-bit integer
int32	Convert to signed 32-bit integer
inline	Construct INLINE object

Multi-dimensional array functions	
cat	Concatenate arrays
ndims	Number of dimensions
ndgrid	Generate arrays for N-D functions and interpolation
permute	Permute array dimensions
ipermute	Inverse permute array dimensions
shiftdim	Shift dimensions
squeeze	Remove singleton dimensions

Cell array functions	
cell	Create cell array
cellfun	Functions on cell array contents
celldisp	Display cell array contents
cellplot	Display graphical depiction of cell array
num2cell	Convert numeric array into cell array
deal	Deal inputs to outputs
cell2struct	Convert cell array into structure array
struct2cell	Convert structure array into cell array
iscell	True for cell array

Structure functions	
struct	Create or convert to structure array
fieldnames	Get structure field names
getfield	Get structure field contents
setfield	Set structure field contents
rmfield	Remove structure field
isfield	True if field is in structure array
isstruct	True for structures

Object oriented programming functions	
class	Create object or return object class
struct	Convert object to structure array
methods	Display class method names
isa	True if object is a given class
isobject	True for objects
inferiorto	Inferior class relationship
superiorto	Superior class relationship
substruct	Create structure argument for SUBSREF/SUBSASGN

Overloadable operators	
minus	Overloadable method for a-b
plus	Overloadable method for a+b
times	Overloadable method for a.*b
mtimes	Overloadable method for a*b
mldivide	Overloadable method for a\b
mrdivide	Overloadable method for a/b
rdivide	Overloadable method for a./b
ldivide	Overloadable method for a.\b
power	Overloadable method for a.^ b
mpower	Overloadable method for a^b
uminus	Overloadable method for -a
uplus	Overloadable method for +a
horzcat	Overloadable method for [a b]
vertcat	Overloadable method for [a;b]
le	Overloadable method for a<=b
lt	Overloadable method for a<b
gt	Overloadable method for a>b
ge	Overloadable method for a>=b
eq	Overloadable method for a==b
ne	Overloadable method for a~=b
not	Overloadable method for ~a
and	Overloadable method for a&b
or	Overloadable method for a—b
subsasgn	Overloadable method for a(i)=b, ai=b, and a.field=b
subsref	Overloadable method for a(i), ai, and a.field
colon	Overloadable method for a:b
end	Overloadable method for a(end
transpose	Overloadable method for a.'
ctranspose	Overloadable method for a'
subsindex	Overloadable method for x(a)
loadobj	Called when loading an object from a .MAT file
saveobj	Called with saving an object to a .MAT file

B21 Windows Operating System Interface Files (DDE/ActiveX) *help winfun*

ActiveX Client Functions	
actxcontrol	Create an ActiveX control
actxserver	Create an ActiveX server
winfun\activex	ActiveX class

Overloadable operators	
ActiveX Demos mwsamp	Sample activex control creation
sampev	Sample event handler for ActiveX server

Overloadable operators	
DDE Client Functions ddeadv	Set up advisory link
ddeexec	Send string for execution
ddeinit	Initiate DDE conversation
ddepoke	Send data to application
ddereq	Request data from application
ddeterm	Terminate DDE conversation
ddeunadv	Release advisory link