Multimédia I
Introdução ao Matlab

Manuela. Pereira
mpereira@di.ubi.pt

October 27, 2016

Chapter 1

Comandos Básicos

O MATLAB é um programa cálculo numérico que pode ser usado interactivamente. A sua estrutura de dados fundamental é a matriz, que pode ter elementos reais ou complexos. Embora na sua versão base o MATLAB já possua um vasto conjunto de funções de carácter genérico, existem várias bibliotecas de funções adicionais (designadas por toolboxes) que expandem as suas capacidades em domínios de aplicação mais específicos.

O Matlab apresenta uma interface gráfica com os vários painéis devidamente separados. A interface está dividida em:

Current Folder Pasta actualmente a ser utilizada.

Command Window Linha de comandos onde é executado o código.

Workspace Lista com as variáveis em memória (com respectivo nome, tipo, entre outras informações).

Command History Histórico dos comandos usados.

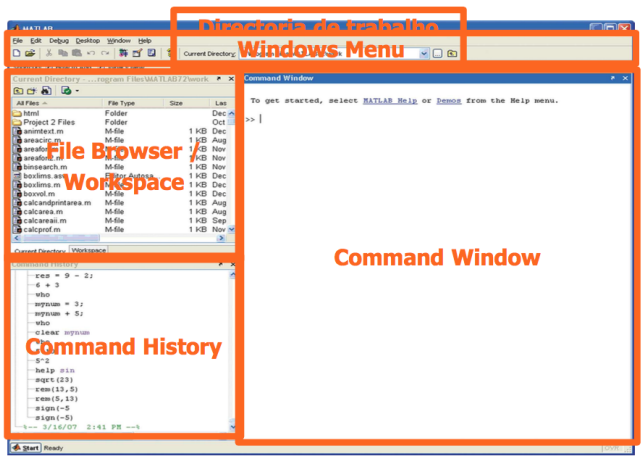

Figure 1.1: Interface do Matlab.

1.1 Comandos de ajuda

Há vários comandos de ajuda no Matlab. Para além da abertura de uma janela de ajuda quando F1 é pressionado, podem ser utilizados outros comandos.

help Mostra os tópicos de ajuda principais na Command Window.

lookfor Procura na ajuda a palavra-chave introduzida.

helpwin Abre uma janela com os comentários de ajuda de todas as funções ou da especificada.

helpdesk Abre janela de ajuda do Matlab.

demos Abre a janela de ajuda nas demonstrações e exemplos.

doc Abre uma janela de documentação.

1.2 Comandos para o sistema de ficheiros

dir apresenta o conteúdo da directoria.

which fich devolve a path para o ficheiro indicado.

what dire devolve os ficheiros existentes na directoria indicada.

cd dire acede à directoria indicada.

type fich mostra o conteúdo de ficheiro.

delete fich apaga o ficheiro indicado.

Chapter 2

Variáveis e Constantes

2.1 Variáveis

Variáveis são objectos utilizados para armazenar valores na memória do computador. No MATLAB todas as variáveis são identificadas por um nome que é formado por um ou mais símbolos alfanuméricos, não podendo o primeiro ser um dígito. É também possível utilizar o símbolo _ como parte do identificador de uma variável. As variáveis são caracterizadas pelo par:

- Identificador: um nome dado à variável para possibilitar sua manipulação
- Conteúdo: o valor a reter, que poderá ser numérico ou alfanumérico

Exemplo:

```
>>  Temperatura = 45; Dia = 12;
```

Algumas notas:

- As variáveis só são válidas durante a execução da aplicação. O conteúdo destas perde-se quando a aplicação é encerrada, podem no entanto ser gravadas em suporte físico permanente
- A cada variável existente na aplicação corresponde uma zona exclusiva na memória principal do computador, sendo esta ligação gerida pelo conjunto aplicação & SO.
- Se nada for dito em contrário o MATLAB atribui o resultado da última operação à variável **ans**.

2.2 Nomes de Variáveis no MATLAB (regras)

- O nome da variável começa sempre por uma letra, podendo ser seguido por uma cadeia de caracteres alfanuméricos
- O comprimento máximo para o nome depende do SO, sendo dado pela instrução namelengthmax

- O MATLAB é case-sensitive (Exemplo: Dia $\neq$ dia $\neq$ DIA)
- Existem palavras reservadas
- Embora possam ser definidas variáveis com o nome de funções (pré-definidas ou definidas pelo utilizador), tal gera confusão e resulta num mau estilo de programação

2.3 Comandos para manipular variáveis

who apresenta as variáveis existentes no ambiente de trabalho

whos informação detalhada das variáveis existentes no ambiente de trabalho.

clear clear all; apaga todas as variáveis existentes no ambiente de trabalho.

clear vars apaga as variáveis enunciadas.

2.4 Utilização de Variáveis e Atribuição

2.4.1 Inicialização, incremento, e decremento

nome_variável = expressão

Exemplos para efetuar no Matlab:

```
>> meu_numero = 6

meu_numero =

     6

>> 6 = meu_numero
 6 = meu_numero
   |
Error: The expression to the left of the equals sign is not a valid target for an
assignment.

>> res = 9 - 2

res =

     7

>> 9 - 2

ans =

     7

>> ans
```

```
ans =

     7

>> meu_numero = 0

meu_numero =

     0

>> meu_numero = meu_numero + 1

meu_numero =

     1

>> meu_numero = meu_numero - 1

meu_numero =

     0
```

2.5 Constantes em MATLAB

Aplica-se a generalidade das características enunciadas para as variáveis. A diferença para estas é que o valor da constante é fixo e atribuído no momento da sua definição.

Exemplos

pi constante Pi ($\pi = 3.141592654...$).

i representação de número imaginário.

j representação de número imaginário.

inf infinito.

NaN not a number (0 / 0).

2.6 Operadores matemáticos

```
help matlab\ops
help ops
```

Exemplos:

+ adição

- subtracção

***** produto

/ divisão

rem resto da divisão inteira (11 / 5 = 2, resto = 1)

$\wedge$ exponenciação

2.7 Funções pré-definidas (exemplos)

```
help matlab\elfun
help elfun
```

Exemplos:

sin, cos, tan, atan funções trigonométricas

sqrt raiz quadrada

log, log10, exp funções de logaritmo e exponenciação

abs valor absoluto

fix, floor, ceil, round funções de arredondamento

2.8 Expressões

As expressões podem ser criadas utilizando:

- Valores numéricos e/ou variáveis (desde que já criadas!)
- Operadores
- Funções pré-definidas e/ou criadas pelo utilizador
- Parêntesis
- Operador Ellipsis (...)

Quando a expressão é terminada com o carácter ';' o MATLAB não mostra o seu resultado. Esta possibilidade é particularmente útil quando estamos a efectuar cálculos intermédios envolvendo grande quantidade de dados e não necessitamos de saber quais os seus valores.

Exemplo para efetuar no Matlab:

```
>> resultado = ( 3 + 5 ) + sin(1.5708) ...
*2

resultado =

   10.0000
```

2.9 Regras de precedência nas expressões

Precedência dos operadores:

() parêntisis;

$\wedge$ expoente;

- sinal de negação;

***, /** produto e divisão;

+,- adição e subtracção.

Exemplos para efetuar no Matlab:

```
>> - ( 3 + 5 ) * 2

ans =

   -16

>>  - 3 + 5 * 2

ans =

     7
```

2.10 Tipos de dados

No MATLAB os tipos são genericamente definidos como classes, as quais são constituídas por, tipo de dados e operações que podem ser executadas sobre o tipo de dados.

- Reais (números fraccionários)
- Inteiros (com ou sem negativos)
- Caracteres (letras, dígitos, símbolos, sinais de pontuação)
- Lógicos (valores lógicos true e false)

```
>>help datatypes
```

Exemplos de tipos de dados no Matlab:

single, double números reais (by default)

uint8, uint16, uint32, uint64 números inteiros sem signal

int8, int16, int32, int64 números inteiros com signal

char cadeias de caracteres

false tipo lógico

Exemplos para efetuar no Matlab:

```
>> val = 6 + 3;
>> whos
  Name      Size            Bytes  Class     Attributes

  val       1x1                 8  double

>> 2 * sin(1.4)

ans =

    1.9709

>> format long
>> 2 * sin(1.4)

ans =

   1.970899459976920

>> format short
>> 2 * sin(1.4)

ans =

    1.9709
```

2.10.1 Cadeias de caracteres

A atribuição do conteúdo de caracteres no MATLAB é feita entre pelicas.

Exemplos para efetuar no Matlab:

```
 >> a='a'

a =

a

>> b= 'x'

b =

x
```

2.10.2 Conversão de tipos

A representação interna no MATLAB é feita com valores numéricos, que podem ser manipulados para fazer a conversão do tipo:

Exemplos para efetuar no Matlab:

```
>> val = 6 + 3;
>> whos
  Name      Size            Bytes  Class     Attributes

  val       1x1                 8  double

>> vali = int32(val);
>> whos
  Name      Size            Bytes  Class     Attributes

  val       1x1                 8  double
  vali      1x1                 4  int32

>> int32('a')

ans =

          97

>> char(97)

ans =

a
```

2.11 Utilidades Diversas

Quando se sai do matlab, perdem-se todas as variáveis. O comando **save** permite gravar todas as variáveis no ficheiro matlab.mat (por omissão) O comando **load** permite restaurar as variáveis a partir desse ficheiro. Pode-se gravar apenas algumas variáveis.

Um exemplo:

```
>> x=2;
>> A=[1 3 6;4 2 6;6 8 9];
>> B=[1 34 45;4 5 6;100 200 300];
>> save backup_aula A B x
>> clear
>> load backup_aula
```

O matlab pode gravar todos os comandos introduzidos na janela de comando. Para esse efeito usa-se o comando **diary**:

Para suspender o comando **diary** faz-se o comando **diary off** e para voltar a habilitá-lo faz-se **diary on**.

```
>> diary 'teste.txt' % grava todos os comandos (excepto gráficos)
>> x=2;
>> A=[1 3 6;4 2 6;6 8 9];
```

```
>> B=[1 34 45;4 5 6;100 200 300];
>> diary off
```

Chapter 3

Vectores e Matrizes

3.1 Vectores

As variáveis podem ser utilizadas para armazenar conjuntos de dados do mesmo tipo na forma de:

- vector coluna com dimensão [n x 1].
  Exemplo de um vetor A de dimensões 3×1:

$$A = \begin{bmatrix} 3 \\ 7 \\ 4 \end{bmatrix}$$

- vector linha com dimensão [1 x m]
  Exemplo de um vetor B de dimensões 1×4:

$$B = \begin{bmatrix} 5 & 88 & 3 & 11 \end{bmatrix}$$

- matrizes com dimensão [n x m]
  Exemplo de uma matriz C de dimensões 3×4:

$$C = \begin{bmatrix} 5 & 88 & 3 & 11 \\ 9 & 6 & 3 & 10 \\ 15 & 34 & 22 & 1 \end{bmatrix}$$

3.2 Criação de vectores linha

A forma mais simples de criar vectores é simplesmente listar os seus elementos. Para tal é apenas necessário usar os caracteres [e] para sinalizar o início e o fim do vector.

Exemplos para efetuar no Matlab:

```
>>  vlin_1 = [1 2 3 4 5 6]
```

```
vlin_1 =

     1     2     3     4     5     6

>> vlin_1 = [1,2,3,4,5,6]

vlin_1 =

     1     2     3     4     5     6

>> vin_1 = 1:6

vin_1 =

     1     2     3     4     5     6

>> vlin_1 = 1:1:6

vlin_1 =

     1     2     3     4     5     6

>> vlin_2 = 1:2:6

vlin_2 =

     1     3     5

>> vlin_3 = 6:-2:1

vlin_3 =

     6     4     2

>> vlin_4 = [vlin_2 vlin_3]

vlin_4 =

     1     3     5     6     4     2
```

3.3 Criação de vectores coluna

Exemplos para efetuar no Matlab:

```
>>  vcol_1 = [1; 2]

vcol_1 =
```

```
     1
     2

>>  vcol_2 = 5:6;
>> vcol_2 = vcol_2'

vcol_2 =

     5
     6

>> vcol = [vcol_1 vcol_2]

vcol =

     1     5
     2     6
```

3.4 Acesso / modificação de vectores

A(i,j) elemento situado na linha i e coluna j

A(imin : imax , jmin : jmax) Para aceder a um subconjunto de elementos é usado o operador dois pontos (:). O resultado obtido são todos os elementos da matriz A definidos pelas linhas imin até imax e pelas colunas jmin até jmax.

Exemplos para efetuar no Matlab:

```
>> vlin = [6 7 8 9]

vlin =

     6     7     8     9

>>  vlin(3)

ans =

     8

>> vlin(3:4)

ans =

     8     9

>> vlin(2) = 5

vlin =
```

```
     6     5     8     9
```

3.5 Funções sobre vetores

Para determinar o valor de uma função em todos os elementos de um vector basta indicar o vector como argumento da função. Assim se pretendermos calcular a função coseno nos pontos correspondentes aos elementos do vector v, basta efectuar:

```
>> v=0:2:10

v =

     0     2     4     6     8    10

>> cos(v)

ans =

    1.0000   -0.4161   -0.6536    0.9602   -0.1455   -0.8391
```

Outras funções que poderão ser úteis:

length Dimensões do vetor.

```
>> vlin = [6 7 8 9]

vlin =

     6     7     8     9

>> length(vlin)

ans =

     4
```

linspace Gera um vetor de valores linearmente espaçados.

```
>>  x=linspace(1, 10, 20)

x =

  Columns 1 through 8

    1.0000   1.4737   1.9474   2.4211   2.8947   3.3684   3.8421   4.3158

  Columns 9 through 16
```

```
    4.7895   5.2632   5.7368   6.2105   6.6842   7.1579   7.6316   8.1053

  Columns 17 through 20

    8.5789   9.0526   9.5263  10.0000
```

3.6 Criação de matrizes

O modo mais simples de definir uma matriz é indicando os seus elementos. Estes deverão estar compreendidos entre os caracteres [e] e são listados linha a linha; cada linha da matriz é terminada com o carácter ';' ou com ENTER.

Exemplos para efetuar no Matlab:

```
>> mat = [4 3 1; 2 5 6]

mat =

     4     3     1
     2     5     6

>> mat = [3 5 7; 1 2]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.

>>  mat = [2:4; 3:5]

mat =

     2     3     4
     3     4     5
```

Também podem ser usadas funções para criação de matrizes especiais, nomeadamente:

A = rand(m,n) Gera matriz com elementos aleatórios.

A = eye(n) Gera matriz identidade

A= ones(m,n) Gera matriz com todos elementos iguais a 1

A= zeros(m,n) Gera matriz com todos elementos iguais a 0

Exemplos para efetuar no Matlab:

```
>> randi([10,30],2,3)

ans =
```

```
    27    12    23
    29    29    12

>> rand(2)

ans =

    0.2785    0.9575
    0.5469    0.9649

>> zeros(2)

ans =

     0     0
     0     0
```

3.7 Acesso aos elementos da matrizes

Exemplos para efetuar no Matlab:

```
>> mat = [4 3 1; 2 5 6]

mat =

     4     3     1
     2     5     6

>>  mat(2,3)

ans =

     6

>> mat(2,:)

ans =

     2     5     6

>> mat(2,2:end)

ans =

     5     6
```

3.8 Modificação de matrizes

Exemplos para efetuar no Matlab:

```
>> mat = [4 3 1; 2 5 6]

mat =

     4     3     1
     2     5     6

>> mat(2,3) = 9

mat =

     4     3     1
     2     5     9

>> mat(2,:) = 11:2:15

mat =

     4     3     1
    11    13    15
```

3.9 O operador dois pontos (:)

O operador **:** é um dos mais úteis no Matlab. Pode ser usado para criar vetores ou matrizes. Para criar vetores cujos valores são igualmente espaçados pode proceder de uma das seguintes formas:

início:fim O resultado é um vector cujo primeiro valor é igual a inicio e os restantes serão incrementados de uma unidade até ao último valor que será igual a fim. Esta sintaxe retorna uma matriz vazia se o valor de início for superior ao valor de fim

início:incremento:fim O resultado é um vector cujo primeiro valor é igual a inicio e os restantes serão incrementados com um valor igual a incremento até ao último valor que será igual a fim. Caso do valor de incremento seja positivo, esta sintaxe retorna uma matriz vazia se o valor de início for superior ao valor de fim. Caso do valor de incremento seja negativo esta sintaxe retorna uma matriz vazia se o valor de início for inferior ao valor de fim.

Também pode usar os dois pontos para criar um vetor de índices para selecionar linhas, colunas ou elementos de matrizes, em que:

A(:,j) é a j-ésima coluna de A.

A(i,:) é a i-ésima linha de A.

A(:,:) é a matriz bidimensional equivalente. Para matrizes este é o mesmo que A.

A(j:k) é A(j), A(j+1),...,A(k).

A(:,j:k) é A(:,j), A(:,j+1),...,A(:,k).

A(:,:,k) é a k-ésima página de uma matriz tri-dimensioanl A.

A(:) são todos os elementos de A, representados numa única coluna . No lado esquerdo de um comando de atribuição, A (:) preenche A, preservando a sua forma de antes. Neste caso, o lado direito tem de conter o mesmo número de elementos de A.

3.10 Informação dimensional

size dimensões da matriz

length tamanho de um vector maior dimensão de uma matriz

numel número de elementos de uma matriz

Exemplos para efetuar no Matlab:

```
>> mat = [4 3 1; 2 5 6];
>> size(mat)

ans =

     2     3

>> length(mat)

ans =

     3

>> numel(mat)

ans =

     6
```

3.11 Re-arranjo de matrizes

reshape modifica as dimensões da matriz rearranjando os elementos desta

fliplr rearranja os elementos da matriz pela troca da esquerda para a direita

flipud rearranja os elementos da matriz pela troca de baixo para cima

rot90 roda os elementos da matriz 90ř no sentido contrário aos ponteiros do relógio

Exemplos para efetuar no Matlab:

```
>> mat = [4 3 1; 2 5 6]

mat =

     4     3     1
     2     5     6

>> fliplr(mat)

ans =

     1     3     4
     6     5     2

>> flipud(mat)

ans =

     2     5     6
     4     3     1

>> reshape(mat,3,2)

ans =

     4     5
     2     1
     3     6

>> rot90(mat)

ans =

     1     6
     3     5
     4     2
```

3.12 Mais exemplos

```
>> x2(5)

ans =

   54
```

```
>> x2(10:end)

ans =

  121  134  148  161  174  188  201  215  228  242  255

>> x2(:)

ans =

    0
   13
   27
   40
   54
   67
   81
   94
  107
  121
  134
  148
  161
  174
  188
  201
  215
  228
  242
  255

>> x2(1:end)

ans =

  Columns 1 through 19

    0  13  27  40  54  67  81  94 107 121 134 148 161 174 188 201 215 228 242

  Column 20

  255

>> x2(end:-1:10)

ans =

  255  242  228  215  201  188  174  161  148  134  121
```

```
>> x2(end:-2:10)

ans =

  255  228  201  174  148  121

>> x2(10:2:end)

ans =

  121  148  174  201  228  255

>> x2([2 7 9 11])

ans =

   13   81  107  134

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

A =

     1     2     3
     4     5     6
     7     8     9

>> A(2,3)

ans =

     6

>> A(:,3)

ans =

     3
     6
     9

>> A(2,:)

ans =

     4     5     6

>> A(1:2,2:3)

ans =
```

```
     2     3
     5     6

>> B=A;
>> B(:,3)=0

B =

     1     2     0
     4     5     0
     7     8     0

>> A(end,end)

ans =

     9

>> A(2:end,end:-2:1)

ans =

     6     4
     9     7

>> A(:)

ans =

     1
     4
     7
     2
     5
     8
     3
     6
     9

>> S=sum(A(:))

S =

    45

>> sum(sum(A))

ans =

    45
```

```
>> sum(A)

ans =

    12    15    18

>>B=A'

B =

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

Chapter 4

Gráficos

Para desenhar um gráfico de uma função real de variável real, teremos de definir dois vectores, um com os valores da variável independente e outro com os valores da função calculados em cada um desses pontos. Assim se quisermos desenhar o gráfico da função seno no intervalo [0,10] teremos de definir:

```
>> x=0:0.05:10;
>> y=sin(x);
>> plot(x,y)
```

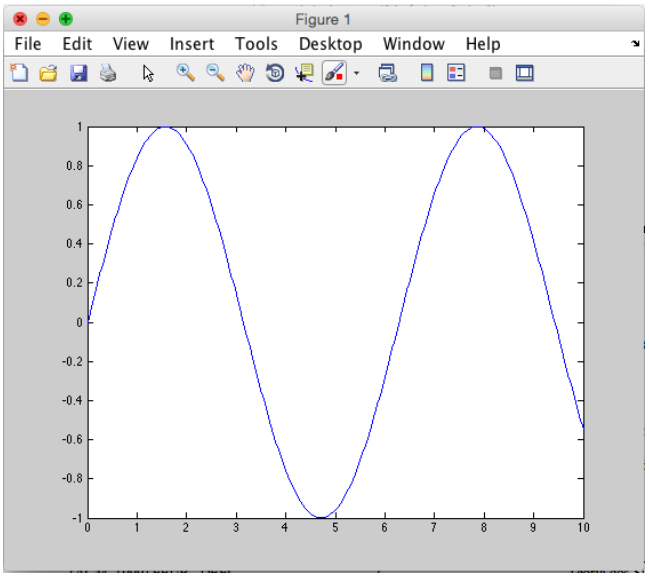

Figure 4.1: Resultado da função plot acima.

O comando help graphics permite listar as funções gráficas disponíveis. O comando help plot permite ver as opções de desenho de gráficos. Alguns exemplos:

```
>>  x=linspace(0,2*pi,30);
```

```
y1 = cos(x);
y2 = sin(x);
hold on
plot(x,y1, 'r-s');
plot(x,y2, 'g-*');
grid
xlabel('eixo x'); % legenda no eixo horizontal
ylabel('eixo y'); % legenda no eixo vertical
title('Grafico do seno e do cosseno'); % título do gráfico >> legend ('sen(x) ', 'co
hold off
```

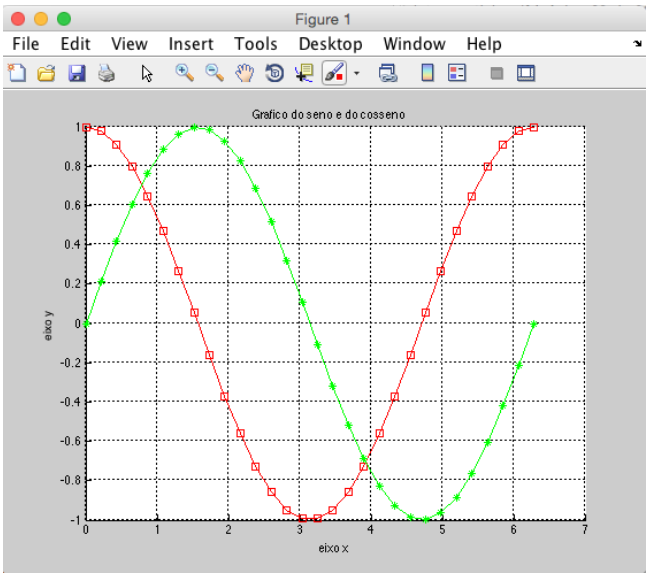

Figure 4.2: Resultado da função plot acima.

Chapter 5

Manipulações Básicas de Sinais de Áudio

Para o Matlab o áudio é representado por vetores. Tantas dimensões quantos os canais de áudio.

audioinfo Informação sobre o ficheiro de áudio.

```
>> audioinfo('e.wav')

ans =

             Filename: '/Users/manuelapereira/Dropbox/_Docencia/1415/M1/Audio_Matlab/dsp1/
    CompressionMethod: 'Uncompressed'
          NumChannels: 2
           SampleRate: 44100
         TotalSamples: 105216
             Duration: 2.3859
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 16
```

audioread('nome do ficheiro') Abre um ficheiro de áudio. Devolve as amostras de áudio, e o número de amostras por unidade de tempo.

```
>> [y,Fs] = audioread('handel.wav');
```

audiowrite(filename,y,Fs) Escreve num ficheiro de áudio.

sound(y) sends audio signal y to the speaker at the default sample rate of 8192 hertz.

```
>> sound(y,Fs)
```

Chapter 6

Manipulação de imagens

Para o Matlab as imagens são matrizes com 2 ou 3 dimensões.

6.1 Funções sobre matrizes

Nesta secção serão apresentadas algumas funções úteis para manipular matrizes que se pretende que representem imagens.

mat2gray Converte os valores em valores no intervalo $[0, 1]$.

```
>> x=linspace(1, 10, 20)

x =

  Columns 1 through 9

    1.0000  1.4737  1.9474  2.4211  2.8947  3.3684  3.8421  4.3158  4.7895

  Columns 10 through 18

    5.2632  5.7368  6.2105  6.6842  7.1579  7.6316  8.1053  8.5789  9.0526

  Columns 19 through 20

    9.5263   10.0000
>> x1 = mat2gray(x)

x1 =

  Columns 1 through 9

         0  0.0526  0.1053  0.1579  0.2105  0.2632  0.3158  0.3684  0.4211

  Columns 10 through 18

    0.4737  0.5263  0.5789  0.6316  0.6842  0.7368  0.7895   0.8421  0.8947
```

```
  Columns 19 through 20

    0.9474    1.0000
```

im2uint8 Converte os valores em unsigned int de 8 bits (0..255).

```
>> x2 = im2uint8(x1)

x2 =

  Columns 1 through 19

    0  13  27  40  54  67  81  94 107 121 134 148 161 174 188 201 215 228 242

  Column 20

  255
```

im2bw Converte os valores em valores 0 e 1.

```
>> x3=im2bw(x2)

x3 =

  Columns 1 through 16

     0     0     0     0     0     0     0     0     0     0     1     1     1     1     1     1

  Columns 17 through 20

     1     1     1     1
```

6.1.1 Abrir imagens

imgetfile Open Image dialog box.

uigetfile Open standard dialog box for retrieving files.

imread ('nome do ficheiro') Abre a imagem para uma variável.
Exemplos: f = imread('chestxray.jpg');
f = imread('D:\myimages \chestxray.jpg'); .

```
>> I = imread('monarch.pgm');
```

imfinfo Detalhes sobre o ficheiro.
Exemplo: iminfo bubbles25.jpg
K = iminfo('bubbles25.jpg');

```
>> imfinfo('monarch.pgm')
```

```
ans =

           Filename: [1x97 char]
        FileModDate: '26-Nov-2008 10:08:02'
           FileSize: 393263
             Format: 'PGM'
      FormatVersion: 'P5'
              Width: 768
             Height: 512
           BitDepth: 8
          ColorType: 'grayscale'
    FormatSignature: 'P5'
           Encoding: 'rawbits'
           MaxValue: 255
    ImageDataOffset: 46
```

6.1.2 Guardar imagens

imwrite Escreve a imagem em ficheiro.
Exemplos: imwrite(f,'nome','tif');
imwrite(f,'nome.tif');
imwrite(f,'nome.jpg', 'quality',25);

```
>> imwrite(f2,'monarch.jpg', 'quality',25);
>> imfinfo('monarch.jpg')

ans =

           Filename: [1x97 char]
        FileModDate: '08-Oct-2014 22:03:14'
           FileSize: 21901
             Format: 'jpg'
      FormatVersion: ''
              Width: 768
             Height: 512
           BitDepth: 8
          ColorType: 'grayscale'
    FormatSignature: ''
    NumberOfSamples: 1
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {}
```

Mostrar imagens

imshow Representação da image.
Exemplos: imshow(f);
imshow(f,[low, high]);

imshow(f,[]);
figure; imshow(f).

image Display image object.

imagesc Scale data and display image object.

impixelinfo Mostra o cursor sobre a última imagem mostrada. O botão X desliga.

6.2 Operações sobre imagens

size Dá as dimensões da imagem.
Exemplo: [M,N] = size(f).

rgb2gray Converte de RGB para níveis de cinza.

imcrop Crop image.

```
>> I = imread('airplane.ppm');
I2 = imcrop(I,[75 68 130 112]);
imshow(I), figure, imshow(I2)

>> I = imread('Rosa1024.pgm');
    I2 = imcrop(I,[250 250 500 500]);
    figure, imshow(I);
    figure, imshow(I2);
> In imagesci/private/readpnm at 92
  In imread at 435
Warning: Image is too big to fit on screen; displaying at 50%
> In imuitools/private/initSize at 72
  In imshow at 259
```

imresize Altera as dimensões de uma imagem.

```
>> I3 = imresize(I,0.5);
>> figure, imshow(I)
    figure, imshow(I3)

>> I4 = imresize(I2,2);
>> figure, imshow(I4)
```

imrotate Efetua uma rotação da imagem.

```
>> I5 = imrotate(I3,90); figure; imshow(I5)
>> I6 = imrotate(I3,-90); figure; imshow(I6)
>> I7 = imrotate(I3,45); figure; imshow(I7)
Warning: Image is too big to fit on screen; displaying at 67%
> In imuitools/private/initSize at 72
  In imshow at 259
>> I8 = imrotate(I3,45,'crop'); figure; imshow(I8)
```