# Matlab num Instante

Versão 1.4

José Manuel Neto Vieira
Departamento de Electrónica e Telecomunicações
da
Universidade de Aveiro

1 de Setembro de 2004

# Conteúdo

## Agradecimentos

O autor agradece aos seus colegas Prof. Ana Maria Tomé, Prof. António Joaquim Teixeira, Prof. Paulo Jorge Ferreira e Prof. José Luís Azevedo, pelas sugestões e correcções realizadas neste documento

# Matlab num Instante

## Introdução

### O que é o Matlab?

O Matlab é um sistema para cálculo científico que proporciona um ambiente de fácil utilização com uma notação intuitiva mas poderosa. Permite a realização de algoritmos numéricos sobre matrizes com o mínimo de programação. Além disso, no ambiente Matlab é possível a criação e manipulação de matrizes sem a necessidade de dimensionamento prévio e a manipulação das variáveis pode ser realizada de forma interactiva. O termo"Matlab" tem origem na conjugação dos termos"MATrix" e"LABoratory".

Este documento pretende ser apenas uma introdução muito breve ao Matlab e permitir a um utilizador não"iniciado" começar a dominar os aspectos mais básicos em pouco tempo. A leitura deste documento deve ser realizada ao lado de um PC com o Matlab, para o utilizador poder testar os exemplos e fazer os diferentes exercícios.

## Ajuda

### O comando HELP

Para esclarecer a maior parte das dúvidas acerca da utilização de uma dada função do Matlab o comando `help` é de grande utilidade. Se se pretender, por exemplo, informação sobre a função `sin`, basta fazer

```
≫help sin
```

obtendo-se a seguinte descrição

```
SIN Sine.
SIN(X) is the sine of the elements of X.
```

O Matlab possui todas as funções organizadas em grupos e a própria estrutura de directórios onde o Matlab é armazenado em disco reflecte esse facto. Por exemplo, todas as funções de álgebra linear estão armazenadas no directório `matfun`. Para obter uma lista completa deste tipo de funções basta fazer

```
≫help matfun
```

Como não é fácil decorar os nomes de todas as categorias de funções, existe uma janela de ajuda mais organizada, bastando para tal escrever o comando

```
≫helpwin
```

### O comando LOOKFOR

Quando se pretende encontrar uma função para resolver um problema, mas desconhece-se se existirá alguma adequada no Matlab, o comando `lookfor` permite pesquisar as primeiras linhas do"help" de todas funções da instalação do Matlab. Esta pesquisa é adequada para resolver a maior parte das situações uma vez que a primeira linha do"help" de uma função contém sempre uma descrição sumária da sua funcionalidade. O seguinte exemplo procura pela palavra"inverse".

```
≫lookfor inverse
```

Se se pretender que a função `lookfor` pesquise todas as linhas do"help", pode-se utilizar a opção `-all`, tal como o seguinte exemplo ilustra.

```
≫lookfor -all inverse
```

### O "Help Desk"

A partir da versão 5.0, o Matlab vem acompanhado com uma ajuda em formato"html" e todos os manuais em formato"pdf". Essa informação pode ser encontrada no endereço http://www.mathworks.com. Dentro do campus da Universidade de Aveiro, também é possível ter acesso directo a essa informação no endereço http://acs.det.ua.pt/matlab.

### O sistema Matlab

O sistema Matlab é constituído pelas seguintes partes:

- A linguagem

  Permite a manipulação e criação de matrizes de forma rápida e intuitiva. Diferentes soluções para um problema podem ser testadas numa fracção do tempo que levaria com outras linguagens (C ou Fortran por ex.). Possui um conjunto muito vasto de funções que permitem resolver problemas complexos de forma eficiente.

- O ambiente de trabalho

  O Matlab proporciona um ambiente de trabalho que permite a gestão e visualização das variáveis, ler e gravar variáveis em disco e gerar programas em linguagem Matlab, possibilitando assim a automatização de cálculos complexos.

- Gráficos

  As funções de criação, visualização e manipulação de gráficos são muito fáceis de usar e permitem a criação de gráficos 2D e 3D. O ajuste de escala é automático e o utilizador pode começar a utilizar as funções de geração de gráficos pouco tempo depois do primeiro contacto com o ambiente do Matlab.

- "Toolboxes"

  O Matlab disponibiliza um conjunto de pacotes de funções para as mais variadas áreas de cálculo científico, sendo estes denominados "toolboxes". Existem"toolboxes" para estatística, processamento de sinal, processamento de imagem, controlo, cálculo simbólico, etc.

## Matrizes

### Criação de matrizes

Uma matriz é uma estrutura de dados bidimensional que permite guardar números de uma forma ordenada e indexável. Para criar, por exemplo, a seguinte matriz com duas linhas e três colunas

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

basta introduzir o comando:

```
≫A= [1 2 3;4 5 6]
```

ou em alternativa

```
≫A= [1,2,3;4,5,6]
```

Os vectores coluna e linha são casos particulares de matrizes e são criados utilizando a mesma notação. Para criar uma variável com apenas um elemento basta fazer

```
≫a= 10
```

Este é um caso particular de uma matriz de dimensão $1 \times 1$, e pode-se igualmente utilizar a notação mais geral

```
≫a= [10]
```

## Índices

O elemento da linha $i$ e da coluna $j$ de uma matriz $A$ é designado por $A(i,j)$. Por exemplo o elemento da linha 1 e coluna 3 da matriz $A$ é designado por $A(1,3)$. Em notação Matlab, para obter o elemento $A(1,3)$ definida anteriormente, pode-se escrever

```
≫A(1,3)
```

e obtém-se

```
ans= 3
```

Para alterar o valor do elemento $A(1,3)$ para 7 basta fazer

```
≫A(1,3)= 7
```

Os índices das matrizes são listas de números inteiros positivos que podem ser armazenadas em vectores declarados previamente. Se pretendermos por exemplo, extrair a segunda linha da matriz $A$ podemos fazer

```
≫v= A(2,[1 2 3])
v=
    4    5    6
```

ou declarando primeiro um vector para os índices das colunas

```
≫k= [1 2 3]
≫v= A(2,k)
v=
    4    5    6
```

## O operador ":"

A criação de vectores elemento a elemento é bastante morosa e para matrizes de grandes dimensões quase irrealizável. O Matlab permite gerar sequências de números de forma rápida se fizermos uso do operador":". Se quisermos gerar o vector $a = [1, 2, 3, \ldots, 10]$ podemos fazer

```
≫a= 1:10
```

A notação geral para o operador ":" é a seguinte

`Número_inical` : `incremento` : `Número_final`

e permite a geração de sequências de números inteiros como no exemplo anterior ou mesmo de números reais. Eis alguns exemplos:

```
≫e= 0:pi/20:2*pi
≫f= 10:-1:-10
```

O operador":" pode ser utilizado na geração de vectores de índices obtendo-se uma notação muito compacta. Se quisermos obter as colunas ímpares da matriz

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

podemos fazer

```
≫B=A(1:3,1:2:3)
```

No caso anterior são indexadas todas as linhas da matriz. Para simplificar a notação, quando não se conhece exactamente o número de linhas de uma matriz, pode-se utilizar a notação

```
≫B=A(:,1:2:3)
```

Se quiséssemos obter a primeira linha da matriz A podíamos fazer

```
≫A(1,:)
```

**Exercício 1** *Gere uma sequência de números pares com início em 4 e a terminar no número 100.*

**Exercício 2** *Gere uma sequência numérica decrescente com início em 5 e a terminar em -5.*

**Exercício 3** *Gere uma sequência numérica com 100 elementos pertencentes ao intervalo* $[0 \ldots 1]$.

**Exercício 4** *Considere uma matriz A com 20 linhas e 30 colunas. Construa um comando que permita extrair para uma matriz B uma sub-matriz de A constituída pelas linhas de 10 a 15 e as colunas de 9 a 12.*

**Exercício 5** *Gere uma sequência de números múltiplos de 3 compreendidos entre 100 e 132, dispostos num vector por ordem decrescente.*

**Exercício 6** *Gere uma sequência a começar em* $\pi$ *e a acabar em* $-\pi$ *com um passo de* $-\pi/15$.

# Expressões

O Matlab permite a construção de expressões matemáticas sem qualquer declaração do formato numérico ou dimensão das matrizes. Existem quatro constituintes básicos nas expressões do Matlab:

- Variáveis
- Números
- Operadores
- Funções

## Variáveis

Todas as variáveis do Matlab são do tipo matriz e a sua criação é automática. Por exemplo, o comando

```
≫Custo_total= 1000
```

resulta na criação em memória de uma matriz de $1 \times 1$ com o valor 1000. O Matlab distingue as letras maiúsculas das minúsculas nos nomes das variáveis e só toma em consideração os primeiros 31 caracteres. Para visualizar o valor de uma variável basta escrever o seu nome.

Existe uma variável especial que é utilizada pelo Matlab quando não se atribui o resultado de uma expressão a qualquer variável. Esta variável designa-se por"`ans`" (do termo"answer", resposta) e pode ser utilizada numa sessão interactiva para continuação dos cálculos, tal como o exemplo seguinte demonstra

```
≫2*sin(2)
ans=
    1.8186
≫ans/102
ans =
    0.017829
```

## Números

O Matlab utiliza uma notação standard para a representação dos números, admitindo notação científica e números complexos. A unidade imaginária $j = sqrt(-1)$ é representada no Matlab pelas letras $i$ ou $j$. Como estas letras podem ser utilizadas no nome de outras variáveis convém ter o cuidado de confirmar o seu valor antes de as usar. A seguir ilustram-se alguns exemplos:

| | | |
|---|---|---|
| 10 | -11 | 0.312 |
| 1+i | 10j | 3e2 |
| 10e6j | 10e-2 | -2.3e-4 |

## Operadores

As operações nas expressões do Matlab seguem as regras habituais de precedência e podem ser aplicadas quer a matrizes quer a números.

| | |
|---|---|
| `+` | Adição |
| `-` | Subtracção |
| `*` | Multiplicação |
| `/` | Divisão |
| `^` | Potenciação |
| `'` | Transposta |
| `()` | Parêntesis |

## Funções

O Matlab possui um conjunto muito grande de funções matemáticas que permitem resolver grande parte dos problemas de cálculo encontrados, tais como: `sin`, `cos`, `tan`, `sqrt`, `log`, etc. Estas aceitam números complexos como argumento e podem também devolver resultados do tipo complexo. Se, por exemplo, se calcular a raíz quadrada de um número negativo não ocorre um erro mas o correspondente número complexo é automaticamente devolvido. Quando as funções são aplicadas sobre matrizes a função é aplicada a cada um dos elementos. Eis uma lista de algumas das funções matemáticas mais comuns e disponéis no Matlab:

| | | | |
|---|---|---|---|
| `cos` | coseno (radianos) | `log` | logaritmo nepriano (base $e$) |
| `sin` | seno | `log10` | logaritmo base 10 |
| `tan` | tangente | `rem` | resto da divisão inteira |
| `acos` | arco-cosseno | `abs` | valor absoluto |
| `asin` | arcoseno | `sign` | sinal |
| `atan` | arco-tangente | `round` | arredonda para o mais próximo |
| `sqrt` | raiz quadrada | `floor` | arredonda para baixo |
| `exp` | exponencial | `ceil` | arredonda para cima |
| `max` | máximo | `min` | mínimo |
| `real` | parte real de um complexo | `abs` | módulo |
| `imag` | parte imaginária de um complexo | `angle` | argumento de um complexo |

Para obter ajuda sobre a utilização de qualquer função basta fazer

≫`help func`

em que `func` é o nome da função.

As funções max e min permitem obter, para além do valor máximo e mínimo de um vector, o índice onde ele ocorre. Vejamos um exemplo:

```
≫a= [1 4 9 2];
≫[m,idx]= max(a)
≫ m = 9
≫ idx = 3
```

Existem algumas funções especiais que devolvem o valor das constantes matemáticas mais utilizadas:

| | |
|---|---|
| `pi` | 3.14159265 |
| `i` | $\sqrt{-1}$ |
| `j` | $\sqrt{-1}$ |
| `eps` | Precisão relativa do formato double $2^{-52}$ |
| `realmin` | O menor número real $2^{-1022}$ |
| `realmax` | O maior número real $(2 - eps)2^{1023}$ |
| `Inf` | Infinito |
| `NaN` | "Not-a-Number" |

## Expressões

Na construção de expressões podemos utilizar operadores, funções e variáveis. Eis alguns exemplos:

```
≫r= (pi+1)/(pi-1)
r =
    1.9339
≫a= abs(4+3i)
a=
    5
≫t= angle(4+3i)
t=
    0.6435
≫a= 0/0
Warning:  Divide by zero.
a =
    NaN
≫x= sqrt(-2)
x =
    0    +    1.4142i
≫x= log(-1)
x=
    0 +    3.1416i
```

## Polinómios

Existe no ambiente Matlab um conjunto de funções para manipular polinómios. A definição de polinómios é feita criando vectores cujos elementos são os coeficientes do polinómio ordenados por potência decrescente tal como o exemplo seguinte ilustra. Para representar o polinómio $p(x) = x^2 - 3x + 2$, cria-se o vector

```
p=[1 -3 2];
```

e para calcular as suas raízes existe a função

```
r= roots(p)
r=
    2
    1
```

Para obter os coeficientes de um polinómio a partir das raízes pode-se utilizar a função

```
poly(r)
ans=
    1    -3    2
```

Também é possível calcular o valor de um polinómio num conjunto de pontos utilizando a função `polyval` tal como o exemplo ilustra

```
polyval(p,r)
ans=
    0
```

```
    0
```

e que, como se pode ver, calculou o valor do polinómio nos seus zeros.

O produto entre dois polinómios é obtido através da função `conv` (convolução). O exemplo seguinte ilustra o produto entre os polinómios $x^2+1$ e $x^3+x-1$ dado por $x^5+2x^3-x^2+x-1$

```
p1= [1 0 1]
p2= [1 0 1 -1]
conv(p1,p2)
ans=
    1 0 2 -1 1 -1
```

**Exercício 7** *Calcule o seno, o coseno, a tangente, a raíz quadrada e a raíz cúbica de* $\pi/2$.

**Exercício 8** *Calcule o logaritmo e a raíz quadrada de -1.*

**Exercício 9** *Calcule o valor da função* $e^x$ *em 100 pontos do intervalo* $[-1\ldots1]$.

**Exercício 10** *Calcule o valor da função* $\sin(x+pi/10)*\cos(x)$ *em 100 pontos do intevalo* $[-\pi\ldots\pi]$.

**Exercício 11** *Calcule o produto dos polinómios* $x^6+10$ *e* $x^2-2x+3$.

**Exercício 12** *Obtenha o polinómio cujas raízes são os números inteiros 1, 2 e 3.*

**Exercício 13** *Calcule os zeros do seguinte polinómio* $p(x)=x^3+4x^2-3x+1$. *Calcule o valor do polinómio em 100 pontos da forma* $x=e^{j\omega}$ *com* $\omega\in[0\ldots2\pi]$.

**Exercício 14** *Calcule o valor do módulo do quociente* $\frac{1}{x^3+1}$ *em 100 pontos da forma* $x=(-2)^w$ *com* $w\in[0\ldots3]$.

# Manipulação de matrizes

Esta secção introduz alguns conceitos elementares sobre a criação e manipulação de matrizes.

## Criação de matrizes

As cinco funções seguintes permitem a criação de algumas matrizes elementares:

| | |
|---|---|
| • `zeros` | Cria uma matriz preenchida com zeros |
| • `ones` | Cria uma matriz preenchida com uns |
| • `eye` | Cria a matriz identidade |
| • `rand` | Cria uma matriz de números aleatórios com distribuição uniforme $[0\ldots1]$ |
| • `randn` | Cria uma matriz de números aleatórios com distribuição normal de média nula e desvio padrão 1. |

Vejamos alguns exemplo de utilização destas funções:

```
≫Z= zeros(2,5)
Z =
    0    0    0    0    0
    0    0    0    0    0
≫a= ones(2,3)*3
a =
    3    3    3
    3    3    3
≫n= round(3*rand(1,10))
```

```
n =
    3    1    2    1    3    2    1    0    2    1
≫A= randn(1,3)+j*randn(1,3)
A =
   -0.43256    +    0.28768i    -1.6656    -    1.1465i    0.12533    +    1.1909i
```

## Concatenação de matrizes

Concatenar matrizes consiste em formar matrizes a partir de outras mais pequenas. A notação é idêntica à utilizada para formar matrizes com números. Os seguintes exemplos ilustram a concatenação de matrizes.

```
≫a= [1 2; 3 4]
a =
    1    2
    3    4
≫A= [a a; a a]
A =
    1    2    1    2
    3    4    3    4
    1    2    1    2
    3    4    3    4
≫b= (1:4)'
b =
    1
    2
    3
    4
≫B= [b b [a;a]]
B =
    1    1    1    2
    2    2    3    4
    3    3    1    2
    4    4    3    4
```

**Exercício 15** *Crie uma matriz* $3 \times 3$ *em que todos os elementos são iguais a 3.*

**Exercício 16** *Crie um vector coluna com 100 elementos aleatórios com uma distribuição uniforme.*

**Exercício 17** *Crie uma matriz* $4 \times 4$ *em que todos os elementos são iguais a* $1 + j2$.

**Exercício 18** *Com a função* `eye` *crie uma matriz diagonal* $4 \times 4$ *em que todos os elementos da diagonal são iguais a 3.*

**Exercício 19** *Construa um vector com 128 elementos com a sequinte sequência:*

$$\begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & \cdots & 0 & -1 \end{bmatrix}.$$

**Exercício 20** *Crie uma matriz com dimensão* $4 \times 4$ *em que cada coluna seja o vector* $v = [1, 2, 3, 4]$.

**Exercício 21** *Forme a matriz quadrada com dimensão* $6 \times 6$ *constituída por quatro blocos em que os elementos de cada bloco sejam "um" ou zero, tal como é ilustrado na seguinte matriz:*

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

### Remoção de colunas e linhas

É possível remover de uma dada matriz qualquer conjunto de linhas e colunas. Para tal, basta atribuir o valor de uma matriz vazia definida por"[]" às linhas e colunas que se pretende remover. No exemplo que se segue, elimina-se a $2^a$ coluna da matriz $A$.

```
≫a= [1 2; 3 4];
≫A= [a a; a a]
A =
    1    2    1    2
    3    4    3    4
    1    2    1    2
    3    4    3    4
≫A(:,2)= []
A =
    1    1    2
    3    3    4
    1    1    2
    3    3    4
```

A remoção de um elemento isolado de uma matriz não é possível uma vez que esta deixaria de respeitar as propriedades de uma matriz

```
≫A(1,2)=[]
???  Indexed empty matrix assignment is not allowed.
```

**Exercício 22** *Gere uma matriz com dimensão* $8 \times 8$ *constituída por números aleatórios obtidos com a função* `randn`*, e obtenha uma sub-matriz constituída pelas colunas de índice ímpar utilizando a técnica de remoção de colunas.*

**Exercício 23** *Resolva o exercício anterior utilizando uma solução sem "[]".*

**Exercício 24** *Gere um vector com todos os números inteiros de 1 a 101 e elinine os elementos pares.*

## Gráficos

A geração de gráficos no Matlab representa um dos seus aspectos mais úteis. Nesta secção serão apresentados alguns dos comandos mais importantes para a criação de gráficos a partir dos valores armazenados em matrizes.

### Criação de gráficos

A função `plot` é a mais utilizada no Matlab para gerar gráficos variando o seu comportamento consoante os parâmetros de entrada. A sua forma mais simples consiste em passar como entrada apenas um vector:

```
≫plot(y)
```

O gráfico gerado apresenta em abcissas os índices $i$ dos elementos do vector e em ordenadas o valor de cada um dos elementos do vector. Também é possível utilizar um segundo vector para o eixo das abcissas tal como no exemplo seguinte:

```
≫w=0:pi/100:2*pi;
≫x= sin(w);
≫plot(w,x)
```

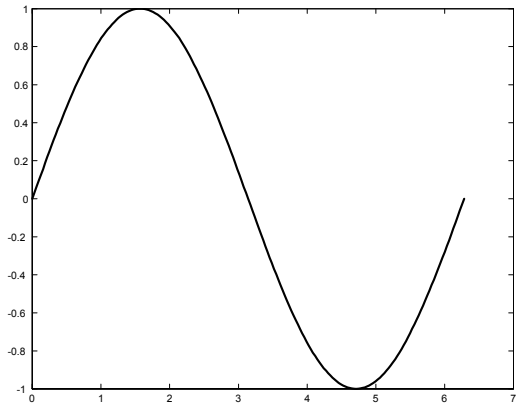


Figura 1:

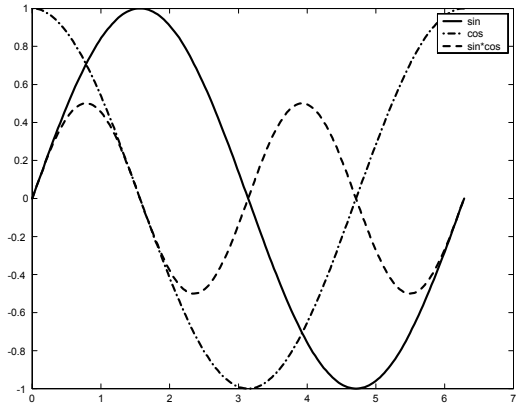



Figura 2:

A função `plot` admite a representação simultânea de várias curvas, acrescentando mais argumentos de entrada, devendo os vectores possuir o mesmo número de amostras. Vejamos um exemplo:

```
≫w=0:pi/100:2*pi;
≫x1= sin(w);
≫x2= sin(w+pi/2);
≫x3= x1.*x2;
≫plot(w,x1,w,x2,w,x3)
≫legend('sin','cos','asin*cos')
```

A função `plot` permite escolher o tipo de linha, a cor, etc; e existem ainda funções para por exemplo, acrescentar etiquetas aos eixos, criar uma grelha, etc, (ver o "help" da função `plot` para referência). Destas funções destacam-se as seguintes:

| | |
|---|---|
| `xlabel` | Etiqueta do eixo das abcissas |
| `ylabel` | Etiqueta do eixo das ordenadas |
| `title` | Nome da figura |
| `legend` | legenda com o significado de cada linha |
| `figure` | Cria uma janela nova |
| `figure(gcf)` | Coloca a janela de gráfica corrente à frente |
| `zoom` | Para aumentar zonas de um gráfico |
| `grid` | Grelha |
| `axis` | Define os limites dos eixos do gráfico |

O Matlab disponibiliza uma pequena função que permite gerar um certo número de valores num dado intervalo. Se se pretender por exemplo, gerar 100 pontos no intervalo de $-\pi$ a $\pi$:

$\gg$`x= linspace(-pi,pi,100);`

Para além da função plot, o Matlab tem ainda várias funções de desenho de gráficos, que se utilizam de forma semelhante mas cujo resultado é diferente. Na tabela em baixo podemos ver alguns exemplos:

| | |
|---|---|
| `semilogx` | Gráfico xy com a escala do eixo horizontal logarítmica |
| `semilogy` | Gráfico xy com a escala do eixo vertical logarítmica |
| `loglog` | Gráfico xy com a escala dos dois eixos logarítimicas |
| `bar` | Gráfico de barras verticais |
| `barh` | Gráfico de barras horizontais |

Para gerar os vectores para os gráficos com escalas logarítmicas convém gerar vectores cujo espaçamento entre pontos seja igualmente logarítmico. A função logspace facilita esta tarefa e tem a seguinte sintaxe: os dois primeiros argumentos são a potência de base 10 dos pontos inicial e final respectivamente, o terceiro argumento é o número de pontos. Vejamos um exemplo. Se se pretender gerar 15 pontos entre 10 e 100 com espaçamento logarítmico, pode-se utilizar o comando

$\gg$`x= logspace(1,2,15);`

Note-se que $10^1 = 10$ e que $10^2 = 100$.

**Exercício 25** *Faça o plot da função seno para dois períodos.*

**Exercício 26** *Compare o gráfico da função seno com o da função coseno para ângulos entre 0 e* $2\pi$ *e um passo de pi/150. O seno deve ser representado por círculos azuis e o coseno por uma linha verde a cheio.*

**Exercício 27** *Desenhe o gráfico da função* $\sin(\theta)/\theta$ *com* $\theta$ *a variar de* $-2\pi$ *a* $2\pi$ *e passos de* $\pi/10$.

**Exercício 28** *Visualize o gráfico da função* $e^x$ *para 100 valores de* $x \in [0 \ldots 5]$.

**Exercício 29** *Visualize o gráfico da função* $e^{-x}$ *para 100 valores de* $x \in [0 \ldots 5]$.

**Exercício 30** *Visualize o gráfico da função* $\log_e x$ *para valores de* $x \in [1 \ldots 5]$.

**Exercício 31** *Construa uma matriz* `X` *em que a primeira coluna é constituída com 100 valores da função* $e^x$ *e a segunda coluna por 100 valores da função* $\log_e x$ *para o intervalo* $x \in [1 \ldots 2]$. *Execute o comando* `plot(X)`.

**Exercício 32** *Determine graficamente uma solução aproximada da equação* $\log_e x = e^{-x}$. *Utilize para a análise o intervalo* $[0.5 \ldots 2]$.

**Exercício 33** *Desenhe no mesmo gráfico a função* $e^{\alpha x}$ *em que a variável* $\alpha$ *toma os seguintes valores* $\alpha \in [1, 1.1, 1.2, 1.3, 1.4, 1.5]$ *e a variável* $x \in [0.1 \ldots 1]$.

**Exercício 34** *Visualize o gráfico da função* $\log_{10} x$ *utilizando uma escala logarítmica para o eixo horizontal. Utilize 100 pontos na gama* $\left[10^{-5} : 10^5\right]$.

## Gráficos de variáveis complexas

Quando o argumento ds função `plot` é complexo, em abcissa é colocada a parte real do vector e em ordenadas a parte imaginária. Pode-se pensar neste processo como uma forma compacta de representação. O comando que se segue

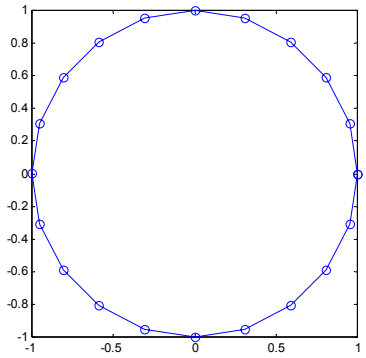
```
≫plot(z)
```

é equivalente a fazer

```
≫plot(real(z),imag(z))
```

Vejamos um exemplo em que se utiliza a função `axis square`, para que a figura obtida seja uma circunferência.

```
≫w= 0:pi/10:2*pi;
≫plot(exp(i*w),'-o')
≫axis square
```

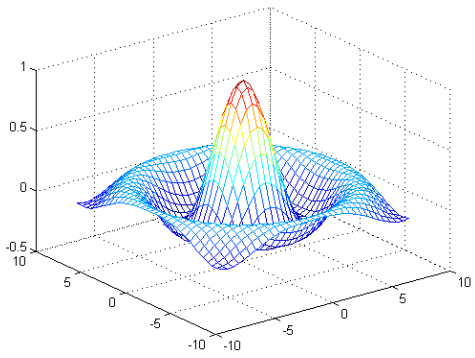



**Exercício 35** *Visualize em gráfico a parte real e a parte imaginária da função* $e^{j\omega}$*, para valores de* $\omega \in [0 \ldots 2\pi]$ *e passo de* $\pi/10$*.*

## Visualização de funções de duas variáveis

Para visualizar uma função de duas variáveis da forma $z = f(x, y)$ o Matlab possui a função `mesh`. Para utilizar esta função é necessário gerar duas matrizes $X$ e $Y$ em que a primeira possui colunas idênticas e a segunda linhas idênticas, com os valores de $x$ e $y$ em que se pretende calcular a função $f$. A função `meshgrid` gera as matrizes $X$ e $Y$ a partir dos valores iniciais, finais e respectivo incremento. O seguinte exemplo ilustra a forma de gerar um gráfico 3D da função $\sin(r)/r$, com $r = \sqrt{x^2 + y^2}$.

```
≫[X,Y] = meshgrid(-8:.5:8);
≫R = sqrt(X.^2 + Y.^2) + eps;
≫Z = sin(R)./R;
≫mesh(X,Y,Z)
```




# O ambiente Matlab

A janela de comando do Matlab possui algumas potencialidades na apresentação dos resultados, gestão da variáveis, formatação numérica dos resultados, edição da linha de comandos, etc. Nesta secção vamos descrever algumas das suas potencialidades.

## Gestão de variáveis

Todas as variáveis do Matlab são guardadas na "área de trabalho" ("workspace"). Para visualizar as variáveis que estão na "área de trabalho" num dado momento podemos utilizar os comados `who` e `whos`. O primeiro apresenta no ecran uma lista abreviada das variáveis indicando apenas o seu nome, enquanto que o segundo apresenta também o espaço ocupado em memória, dimensão, etc.

Para apagar variáveis da memória pode-se utilizar o comando `clear`. Para apagar por exemplo as variáveis `x` e `b` faz-se

```
≫clear x b
```

O comando `clear` sem argumentos apaga todas as variáveis armazenadas em memória.

## O comando `format`

O comando `format` controla a forma como os valores numéricos são apresentados. Este comando controla apenas a forma com os valores são apresentados e não a forma como estes são representados internamente. Os seguintes exemplos ilustram os formatos mais comuns.

```
≫a= [1/5 pi]
format short
       0.2000    3.1416
format short e
       2.0000e-001    3.1416e+000
format long
       0.20000000000000    3.14159265358979
format long e
       2.000000000000000e-001    3.141592653589793e+000
```

Uma opção útil do comando `format` é

```
≫format compact
```

que elimina as linhas em branco extra introduzidas entre a apresentação das variáveis permitindo uma apresentação dos resultados mais compacta.

## Supressão de resultados

Quando as matrizes são de grande dimensão torna-se bastante incómodo para o utilizador a apresentação do resultado no ecran de todos os cálculos efectuados. Para evitar a apresentação dos resultados basta colocar no final da linha de comando um ponto e vírgula tal como o seguinte exemplo demonstra:

```
≫a= [1/5 pi];
```

## Edição de comandos

A janela de comando do Matlab permite a edição de comandos escritos anteriormente. Se por exemplo se entrou o seguinte comando:

```
≫a= sqrt(1:10]
              |
Improper function reference.  A ’’,’’ or ’’)’’ is expected.
```

o Matlab devolve, como se pode constatar, um erro de sintaxe. Para alterar a linha e substituir o parêntesis recto por um curvo, basta carregar na tecla ↑ que a linha aparecerá de novo. Com as teclas ⟵ e ⟶ desloca-se o cursor ao longo da linha. O Matlab guarda uma lista de todos os comandos introduzidos numa sessão. Se quiser obter um comando introduzido anteriormente, basta escrever os primeiros caracteres desse comando e carregar em ↑ para que o comando apareça. A lista seguinte mostra as teclas de edição disponíveis

| | |
|---|---|
| ⟵ | Movimenta o cursor para a esquerda |
| ⟶ | Movimenta o cursor para a direita |
| ↑ | Salta para o comando anterior |
| ↓ | Salta para o comando seguinte |
| `home` | Salta para o início da linha |
| `end` | Salta para o fim da linha |
| `esc` | Apaga a linha |
| `del` | Apaga o carácter da direita |
| `backspace` | Apaga o carácter da esquerda |

Existe ainda a possibilidade de guardar num ficheiro todos os comandos introduzidos durante uma sessão com o comando `diary`.

## Directórios e ficheiros

Quando se arranca o Matlab, o directório corrente será o `c:\matlabR12\work`. No entanto, se pretendermos guardar num ficheiro variáveis, funções e programas Matlab elaborados por nós, é conveniente mudar a directoria corrente para uma directoria pessoal. Para visualizar qual o directório corrente pode-se utilizar o comando

```
≫pwd
ans =
C:\matlabR12\work
```

Para mudar de directório pode-se utilizar o comando `cd`. Para mudar para a “drive” `H:` por exemplo, o seguinte comando deve ser utilizado

```
≫cd h:
```

Quando se escreve um comando na janela do Matlab esse nome será pesquisado em primeiro lugar no directório corrente. Se não for encontrado, o Matlab pesquisa nos directórios que constarem duma lista interna (comando `path`).

### Os comandos LOAD e SAVE

Com estes comandos é possível guardar num ficheiro variáveis do Matlab que estejam no "workspace". O comando `save` permite guardar as variáveis para disco e o comando `load` permite carregá-las para o "workspace". O exemplo seguinte ilustra a funcionalidade destes comandos.

```
≫A= rand(2)
A =
    0.61543    0.92181
    0.79194    0.73821
≫b= rand(1,3)
b=
    0.9501     0.2311     0.6068
≫save ficheiro A b
≫load ficheiro
```

Os comandos `load` e `save` permitem igualmente guardar ficheiros em formato ASCII.

```
≫save ficheiro A b -ascii
```

sendo mais fácil para o utilizador verificar o conteúdo com um simples editor de texto.

**Exercício 36** *Execute a sequência anterior de comandos e guarde as variáveis A e b em formato ASCII para um ficheiro com o nome* `teste.txt`*. Edite o conteúdo desse ficheiro com o* `Notepad`*. Repita a operação para o ficheiro gerado com* `save ficheiro A b`*.*

**Exercício 37** *Após ter criado as variáveis* `A` *e* `b` *tal como é indicado no texto, execute o comando* `whos` *e compare a memória ocupada por cada uma delas. Visualize a variável* `b` *nos seguintes formatos numéricos:* `long`*,* `short e` *e* `long e`*. Finalmente, apague as variáveis e verifique com o comando* `whos`*.*

### Ficheiros de comandos

Quando se pretende executar repetidamente um conjunto de comandos muito longo, ter de os escrever torna-se muito moroso. Para resolver este problema, é possível colocar num ficheiro de texto com a extensão ".m" o conjunto de comandos que se pretende executar. Para executar os comandos guardados no ficheiro, basta escrever na janela de comando do Matlab o nome do ficheiro sem a extensão. Para criar e começar a editar um ficheiro basta escrever o comando `edit`, para que arranque um pequeno editor de texto com "debugging" integrado.

## Tópicos sobre matrizes

Nesta secção iremos descobrir mais algumas das possibilidades que o Matlab oferece para operar com matrizes.

### Álgebra linear

É possível resolver com o Matlab diversos problemas da álgebra linear, sendo fácil realizar cálculos elaborados com matrizes, como por exemplo, o produto de duas matrizes, inversão de matrizes, cálculo dos valores próprios, etc. Dadas as matrizes $A$ e $B$, por exemplo, a sua soma é calculada da seguinte forma:

```
≫A= rand(4);
≫B= rand(4);
≫C= A+B;
```

Para calcular o produto entre as duas matrizes anteriores basta fazer

```
≫C= A*B
```

que efectua produtos internos entre as linhas de $A$ e as colunas de $B$. Se quisermos calcular o produto interno entre dois vectores linha $x$ e $y$

```
≫x= 1:4
≫y= 3:6
≫x*y'
ans=
    50
```

enquanto que o produto externo entre estes vectores é dado por

```
≫ A = x'*y
A =
     3     4     5     6
     6     8    10    12
     9    12    15    18
    12    16    20    24
```

O determinante da matriz $A$ pode ser calculado fazendo apenas

```
≫d= det(A)
```

e se quisermos calcular os valores próprios da matriz $A$, pode-se utilizar o comando

```
≫eig(A)
```

e a inversa da matriz $A$ obtém-se fazendo

```
≫inv(A)
```

É igualmente possível calcular a $n$ potência de uma matriz quadrada fazendo simplesmente

```
≫A^n
```

e se se pretender calcular o polinómio característico de uma matriz temos

```
≫p= poly(A)
p=
    1     2.1     1.1     -5     2
```

o que indica que o polinómio característico

$$\det\left(A - \lambda I\right)$$

é

$$\lambda^4 + 2.1\lambda^3 + 1.1\lambda^2 - 5\lambda^1 + 2.$$

O Matlab permite resolver sistemas de equações numericamente de forma muito eficiente. Considere-se o seguinte sistema de equações:

$$Ax = b$$

em que

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 4 & 1 & 2 \\ 6 & 3 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}.$$

Para resolver o sistema de equações podemos utilizar a equação

$$x = A^{-1}b$$

e o correspondente comando Matlab

```
≫x= inv(A)*b
x=
 0.0588
 0.7059
-0.4706
```

Uma forma alternativa de resolver o sistema de equações e que possui uma maior estabilidade numérica consiste em utilizar a divisão à esquerda do Matlab

```
≫x= A\b;
```

O Matlab utiliza o algoritmo de eliminação Gaussiana para resolver o sistema de equações.

**Exercício 38** *Construa um vector linha $x$ constituído pelos números inteiros pares pertencentes ao intervalo $[1 \ldots 8]$. Calcule os valores próprios da matriz resultante do produto $x^T x$.*

**Exercício 39** *Resolva o seguinte sistema de equações com o Matlab*

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

*Verifique com o Matlab a solução obtida.*

**Exercício 40** *Considere o seguinte sistema de equações*

$$\begin{cases} x + 2y - z = 0 \\ 2x + 4y - 2z = 2 \\ -x + 2z = 1 \end{cases}$$

*Verifique que este sistema não tem solução.*

**Exercício 41** *Verifique que fazendo o produto matricial entre um vector coluna $v$ e um vector linha preenchido com um número $n$ de uns, se obtém uma matriz com $n$ colunas iguais a $v$.*

**Exercício 42** *Calcule os valores próprios da matriz gerada com o seguinte comando Matlab* `A=rand(4)+1`*, e verifique que pelo menos um deles é superior à unidade. Calcule $A^{20}$ e verifique que os elementos da matriz resultante aumentaram de valor. Calcule agora $\left(\frac{A}{20}\right)^{20}$ e verifique que os elementos da matriz resultante diminuiram de amplitude e que os valores próprios são todos inferiores à unidade.*

## Operações elemento-a-elemento

Em muitas situações pretende-se efectuar operações aritméticas com matrizes aplicando a operação entre os elementos das matrizes. Para transformar um operador matricial na forma elemento-a-elemento, basta acrescentar o carácter "`.`" antes do operador. Note-se que no caso da adição "`+`" e da subtração "`-`" não existe qualquer diferença para os operadores "`.+`" e "`.-`" respectivamente.

O exemplo seguinte ilustra o produto elemento-a-elemento de duas matrizes com a mesma dimensão:

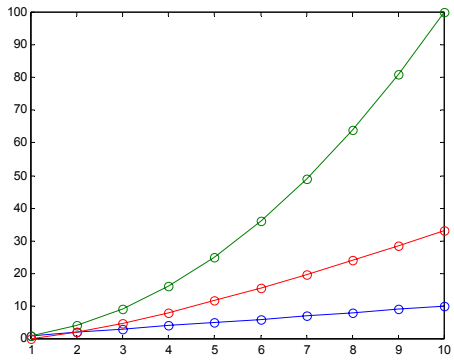
```
≫A= [1 2 3; 4 5 6]
≫B= [2 3 4; 9 8 7]
≫A.*B
ans=
       2     6    12
      36    40    42
```

Vejamos agora uma lista de operadores elemento-a-elemento

| | |
|---|---|
| `+` | Adição |
| `-` | Subtração |
| `.*` | Multiplicação elemento-a-elemento |
| `./` | Divisão elemento-a-elemento |
| `.^` | Potenciação elemento-a-elemento |
| `.'` | Transposta não conjugada |

Este tipo de operador é muito útil para, por exemplo, gerar matrizes com valores de funções:

```
≫n= (1:10)';
≫M= [n n.^2 n.*log2(n)];
plot(M,'-o')
```




**Exercício 43** *Crie uma matriz com três colunas e coloque em cada uma delas as seguintes funções: sin(x), cos(x) e sin(x)*cos(x). O argumento $x$ deverá ser um vector a começar em 0 e a acabar em $2 \times \pi$, com um passo de $\pi/100$.*

**Exercício 44** *Desenhe o gráfico da função $\sin\left(x^2\right)$ utilizando o mesmo argumento que no exercício anterior.*

**Exercício 45** *A função* **`abs`** *dá como resultado o módulo de um número complexo e a função* **`angle`** *o seu argumento. Construa um gráfico do módulo e argumento da seguinte função*

$$y\left(\omega\right) = \omega \times e^{\mathrm{j}\omega}$$

*em que $\omega$ varia de 0 a $8\pi$ com passos de $\pi/40$. Desenhe igualmente um gráfico da parte real versus a parte imaginária.*

## Funções estatísticas

O Matlab possui uma grande variedade de funções estatísticas para analisar um sinal. Quando estas funções recebem uma matriz como entrada, a operação é aplicada sobre cada coluna da matriz. Para calcular a média de cada uma das colunas de uma matriz podemos fazer:

```
≫M= randn(100,3);
≫mean(M)
```

para calcular o desvio padrão pode-se fazer

```
≫std(M)
```

e para obter um histograma de cada coluna de $M$

```
≫hist(M)
```

Se se pretender um histograma com mais classes basta acrescentar um segundo argumento à função com o número pretendido.

**Exercício 46** *Gere um vector com um milhão de amostras obtidas com a função* **`rand`**. *Calcule a sua média, desvio padrão e visualize o seu histograma com 100 classes.*

**Exercício 47** *Gere agora um vector com um milhão de amostras com a função* **`randn`**. *Calcule a sua média, desvio padrão e visualize o seu histograma com 100 classes. Quais as diferenças em relação ao exemplo anterior?*

**Exercício 48** *Gere uma sequência com 100 elementos iguais à unidade e calcule a sua média e desvio padrão.*

## Expansão escalar

A expansão escalar consiste na repetição automática de um escalar de modo a que seja possível realizar a operação pretendida com cada elemento de uma matriz. Se pretendermos somar o número 5 a uma matriz $A$ de dimensão $4 \times 4$ basta fazer

```
≫A+5
```

em vez de

```
≫A+5*ones(4)
```

Esta notação simplifica as expressões tornando-as de leitura mais fácil. Também é possível realizar atribuições fazendo por exemplo:

```
≫A= magic(4);
≫A(1:3,1:3)= 1
     1   1   1  13
     1   1   1   8
     1   1   1  12
     4  14  15   1
```

## Índices lógicos

O Matlab permite a indexação de elementos de uma matriz através de índices lógicos que podem ser obtidos a partir de equações lógicas. Considere-se o seguinte vector $v$

```
≫v= [0.1 0.4 0.3 -0.3 -0.9 0.4 0.1 -0.5 -0.1 -0.3 0.1 0.5]
```

e vamos supor que se pretendia obter um vector apenas com as amostras com amplitude superior a 0.3; então, fazendo

```
≫v>0.3
ans =
    0    1    0    0    0    1    0    0    0    0    0    1
```

obtém-se um vector de zeros e uns com o valor 1 nas amostras onde a condição era verdadeira. Para obter os elementos do vector que satisfazem a condição basta fazer

```
≫v(v>0.3)
    0.4    0.4    0.5
```

Se se pretender um vector com os índices das amostras que satisfazem a condição $v > 0.3$ pode-se fazer

```
≫n= 1:length(v);
≫n(v>0.3)
ans=
    2    6    12
```

ou, em alternativa, utilizar a função `find`

```
≫ find(v>0.3)
ans=
    2    6    12
```

Na tabela seguinte são apresentados os operadores lógicos mais comuns da linguagem Matlab

| | |
|---|---|
| == | igual |
| ~= | diferente |
| < | menor |
| > | maior |
| <= | menor ou igual |
| >= | maior ou igual |
| & | "e" lógico |
| \| | "ou" lógico |
| ~ | negação |

**Exercício 49** *Gere a sequência* $\{1, 2, 3, \ldots, 9, 10\}$ *e extraia os números maiores que 4.*

**Exercício 50** *Gere uma sequência aleatória com distribuição normal e elimine os elementos negativos.*

**Exercício 51** *Gere uma sequência aleatória com distribuição normal e calcule o número de elementos negativos.*

**Exercício 52** *Gere um sinal aleatório com distribuição uniforme e com 1000 amostras. Obtenha o número de amostras com uma amplitude superior a 0.9. Calcule em seguida o número de amostras cujo módulo possui uma amplitude entre 0.5 e 0.7.*

# Programação em Matlab

O Matlab possui um conjunto de instruções que permitem a construção de programas. Nesta introdução apenas faremos referência às instruções `if` e `for`.

## A instrução IF

Esta instrução permite a execução condicional de código tendo em conta uma dada condição lógica. A estrutura é a seguinte

```
if cond,
    instrução 1
    instrução 2
    ...
else
    instrução 3
    instrução 4
    ...
end
```

Se a condição `cond` for verdadeira são executadas as instruções 1 e 2, senão serão executadas as instruções 3 e 4. A directiva `end` no final indica o fim da instrução `if`.

Na forma mais simples a directiva `else` pode ser omitida.

## A instrução FOR

Esta instrução permite a repetição de um conjunto de instruções. A sintaxe de utilização é a seguinte:

```
for variável= início:passo:fim,
    Instrução 1
    Instrução 2
    ...
end
```

A seguir podemos ver alguns exemplos de utilização

```
≫r= 10;
≫for n= 1:10,
≫    r= r+10
≫end
```

Não é difícil verificar que a expressão à direita do sinal de igual na instrução `for` apenas cria um vector e que se pode utilizar a instrução `for` da seguinte forma

```
≫r=10;
≫v= 1:10;
≫for n= v,
≫    r= r+10
```

```
≫end
```

A instrução `for` só deve ser utilizada quando não se encontra outra forma de implementar o algoritmo pois como veremos na secção sobre "Vectorização de algoritmos" (pag.27), torna a execução do programa muito lenta uma vez que o Matlab utiliza uma linguagem interpretada.

**Exercício 53** *Gere um vector $x$ com 1000 elementos utilizando a função rand. Extraia para um outro vector todos os elementos de $x$ menores que 0.3. Utilize as instruções* `for` *e* `if`*.*

**Exercício 54** *Neste exercício pretende-se visualizar a função $e^{\alpha x}$ quando se varia o parâmentro $\alpha$ segundo a expressão $\alpha = 1 + 0.1 \times n$, com $n \in [1, 2, \ldots, 10]$. Armazene cada uma das variantes da função numa coluna de uma matriz $A$ $(100 \times 10)$ considerando 100 pontos para a variável $x \in [0 \ldots 2]$. Para visulizar as curvas simultâneamente faça apenas* `plot(A)` *e experimente* `mesh(A)`*.*

**Exercício 55** *A expansão em série de Taylor da função seno é dada pela expressão*

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{x^n}{n!} \sin\left(n\frac{\pi}{2}\right).$$

*Calcule a aproximação para 5 termos da série em 100 pontos do intervalo $[0 \ldots 2\pi]$ e faça um gráfico comparando a aproximação com a função seno.*

**Exercício 56** *Usando a instrução* `for`*, construa uma matriz de $10 \times 10$ em que cada coluna é um vector $v$ cujos elementos são os números inteiros de 1 a 10*[1]*.*

## Som e Imagem no Matlab

O Matlab permite a manipulação directa de ficheiros de som e imagem, sendo possível experimentar algoritmos de processamento sobre sinais reais e verificar os resultados de imediato.

## Sinais de Áudio

O Matlab possui uma série de funções que permitem manipular ficheiros de som em formato "riff"[2] e, se o computador tiver uma placa de som, reproduzir o sinal armazenado num vector. A instrução `sound` permite fazer esta reprodução utilizando uma dada frequência de amostragem. A frequência de amostragem indica o número de amostras que são convertidas de digital para analógico em cada segundo. Se quisermos reproduzir um vector preenchido com um sinal aleatório a uma taxa de 8000Hz e com uma duração de 2 segundos utilizamos a seguinte sequência de comandos:

```
≫r= 2*(rand(1,16000)-0.5);
≫sound(r,8000);
```

A amplitude dos elementos do vector de entrada da função `sound` deve estar compreendida entre -1 e 1, sendo as amplitudes superiores convertidas para -1 ou 1. Se quisermos reproduzir uma sinusóide de 1000Hz com a duração de 0.5 segundos, utilizando uma frequência de amostragem de 8000Hz procede-se da seguinte forma

```
≫fs= 8000;        % Frequência de amostragem
≫Ts= 1/fs;        % Período de amostragem
≫t= 0:Ts:0.5;    % Instantes de amostragem
≫x= sin(2*pi*1000*t);
≫sound(x,fs);
```

Para carregar para um vector um sinal armazenado num ficheiro de som com o formato "riff", pode-se utilizar o comando

[1] Este exemplo da instrução `for`, pode ser realizado de forma mais eficiente com o comando: `v*ones(1,length(v))`, em que `v` é um vector coluna.

[2] No ambiente windows os ficheiros de som com este formato têm a extensão "`wav`".

```
≫[y,fs]= wavread('ficheiro.wav');
```

e se se pretender gravar num ficheiro com a extensão "`wav`" um vector pode-se utilizar o comando

```
≫wavwrite(y,fs,'ficheiro.wav');
```

Apesar de no Matlab se poder escolher qualquer frequência de amostragem a maior parte das placas de som não permitem a reprodução de sinais amostrados a não ser nas seguintes frequências em Hz:

| | | |
|---|---|---|
| 48000 | 24000 | 12000 |
| 44100 | 22050 | 11025 |
| 32000 | 16000 | 8000 |

.

**Exercício 57** *Gere um vector com 32000 elementos utilizando a função* ***randn****. Ouça o sinal criado utilizando uma frequência de amostragem de 32000Hz.*

**Exercício 58** *Gere a sinusóide de 1000Hz do exemplo dado e ouça-a com a frequência de amostragem de 8000Hz. Reproduza agora a sinusóide com as seguintes frequências de amostragem: 4000, 16000, 44100.*

**Exercício 59** *Gere um sinal constituído por 5000 elementos 1 e -1 dispostos alternadamente. Reproduza esse sinal com a função* ***sound*** *utilizando uma frequência de amostragem de 8000Hz.*

### Imagens

O Matlab possui um conjunto de funções para manipulação e visualização de imagens a cores. As imagens são representadas sob a forma de matrizes. No caso de imagens só com níveis de cinzento, estas podem ser armazenadas numa matriz bidimensional em que cada elemento representa o nível de cinzento. Os seguintes comandos mostram como visualizar uma imagem armazenada num ficheiro em disco

```
≫I=imread('banner.jpg');
≫image(I)
axis image
```

O comando `imread` carrega a imagem `banner.jpg` para a matriz I, e o comando `image` mostra a figura no ecran.

Podemos facilmente criar imagens sintéticas com o Matlab e visualizá-las com os comandos anteriores. Vejamos um exemplo de uma imagem com a metade inferior a branco e a superior a preto:

```
≫M= zeros(100);
≫M(1:50,:)= 1;
≫colormap gray
≫imagesc(M);
```

## Construção de código eficiente com o Matlab

O Matlab permite o desenvolvimento rápido de algoritmos de processamento, mas quando os dados a tratar possuem uma dimensão elevada é necessário ter alguns cuidados para que a execução do código não se torne demasiado lenta. Existem duas formas principais de optimizar a execução do código: a reserva de memória para as variáveis e a vectorização de algoritmos.

## Reserva de espaço para variáveis

A reserva de espaço em memória para as variáveis faz-se simplesmente preeenchendo a variável a ser utilizada com zeros, de forma a que a sua dimensão não volte a ser alterada. No código que se segue o vector `x` vê a sua dimensão ser alterada, o que leva a um novo pedido de reserva de memória para armazenar o vector

```
≫x= 1:5;
≫x= [x x];
```

o que não acontece nesta versão que apesar de mais eficiente é menos legível

```
≫x= zeros(1,10);
≫x(1:5)= 1:5;
≫x(6:10)= x(1:5);
```

Mas vejamos que diferença pode ocorrer nos tempos de processamento com um pequeno programa, em que se testam duas versões de um algoritmo muito simples, em que na primeira não se reserva memória para a variável.

```
N= 1000;
x= rand(1,N);
% Versão sem reserva de memória
t= cputime;
y= 0;
for n= 1:N,
    y= [y x(n)];    % Instrução crítica
end
tsem= cputime-t
clear y
%
% Versão com reserva de memória
t= cputime;
y= zeros(1,N+1);
for n= 1:N,
    y(n+1)= x(n);
end
tcom= cputime-t
% Velocidade relativa
% Num Pentium II a 266MHz a versão com reserva de memória
% foi cerca de 8 vezes mais rápida
rel= tsem/tcom
```

Eis mais alguns conselhos úteis sobre a gestão de memória no Matlab

- Evitar criar variáveis intermédias dos dados quando não são necessárias utilizando antes variáveis da mesma dimensão já existentes.
- Caso tal não seja possível utilizar a função `clear var1 var2 ...`l
- Deve-se ter em conta que o Matlab utiliza 8 bytes para armazenar cada elemento de uma matriz. Como no caso das variáveis complexas esse número duplica, deve-se evitar ter este tipo de variáveis sempre que a parte real ou a imaginária sejam nulas. Nestes casos deve utilizar-se as funções `real` e `imag`.
- No caso de ser necessário processar dados com muitas amostras deve-se realizar o processamento por blocos com as seguintes etapas:

  1. Ler bloco do ficheiro de entrada
  2. Processar bloco

3. Escrever o bloco com o resultado no ficheiro de saída

Esta técnica apesar de permitir um aumento da rapidez de execução, resulta normalmente num algoritmo mais complexo.

## Vectorização de algoritmos

Todos os operadores e funções do Matlab podem ser aplicados sobre vectores e matrizes. Se, por exemplo, quisermos calcular o seno de cada um dos elementos do vector $x$ com 100 elementos basta fazer

```
≫v= sin(x)
```

para se obter o resultado pretendido. O Matlab permite assim utilizar como argumento da função `sin` não apenas um número mas um vector, aplicando a função a cada um dos seus elementos. No entanto, existe uma outra forma de realizar os cálculos que consiste em recorrer ao ciclo `for` aplicando a função `sin` a um elemento de `x` de cada vez

```
≫for n= 1:100
≫    v(n)= sin(x(n));
≫end
```

Contudo, o Matlab utiliza uma linguagem interpretada, o que significa que antes de ser executada cada linha de código é lida e descodificada. Este processo é repetido para cada linha e pode em certos casos, ser da ordem de grandeza do próprio processamento ou mesmo superior. Um caso ilustrativo é o do cálculo do produto interno entre dois vectores linha $a$ e $b$, definido pela expressão

$$a \cdot b = \sum_{n=1}^{N} a(n)\, b(n)\,,$$

e que em linguagem do Matlab se escreve simplesmente `a*b'`.

O programa que se segue pretende demonstrar a diferença de velocidade de processamento entre a versão que calcula o produto interno fazendo uso de um ciclo `for`, e outra que o calcula na forma vectorizada.

```
% Comparação do tempo de CPU no cálculo de um produto interno.
N= 10000;
a= rand(1,N);
b= rand(1,N);
nciclos= 100;
% Cálculo utilizando um ciclo for
t= cputime;
for c= 1:nciclos,
    y= 0;
    for n= 1:N,
        y= y+a(n)*b(n);
    end
end
tfor= cputime-t

% Cálculo de forma vectorizada
t= cputime;
for c= 1:nciclos,
    y= a*b';
end
tmat= cputime-t
% Velociade de cálculo relativa
% Num Pentium II a 266MHz foi cerca de 180 vezes mais rápido
```

```
rel= tfor/tmat
```

Pelo exemplo anterior pode-se concluir que a "vectorização" dos algoritmos é a única forma de utilizar o Matlab de forma eficiente.

**Exercício 60** *Obtenha uma matriz com 50 colunas em que cada coluna contém um período da função seno amostrada em 50 pontos.*

**Exercício 61** *Gere uma sequência com 10000 valores aleatórios com distribuição uniforme e guarde num vector apenas os que possuem amplitude superior a 0.9. Coloque noutro vector os índices das amostras anteriores.*

**Exercício 62** *Obtenha o gráfico da função* $\sin(x)/x$ *variando o argumento entre -100 e 100 com passos de 0.1.*