



Sistemas Multimédia

2024/2025

Breve Introdução ao MatLab

Universidade de Aveiro



Sumário

- Introdução
- Variáveis no MatLab
- Vetores e Matrizes
- Operação elemento-a-elemento
- Indexação lógica
- Gráficos 2D e 3D
- Programação em MatLab
- Funções em MatLab



Ambientes de Cálculo Numérico

Matlab



O que é o Matlab ?

- Aplicação informática vocacionada para o cálculo numérico
- MATLAB = MATrix LABoratory
 - Os elementos são sempre matrizes
 - Um número é uma matriz com apenas um elemento
- Aplicações
 - Análise de dados
 - Visualização científica
 - Simulação de sistemas



Inatalação

- Empresa que comercializa o Matlab é a Mathworks
 - <https://www.mathworks.com/>
- Instalação MATLAB
 - <https://www.mathworks.com/academia/tah-portal/universidade-de-aveiro-40766421.html>
 - Use as suas credenciais de Utilizador Universal
- Ajuda
 - https://www.mathworks.com/support/contact_us.html?s_tid=tah_po_helpbutton_ua.pt
- Aprenda MATLAB em duas horas: Curso online
 - <https://matlabacademy.mathworks.com/>



Demonstração

- O Matlab tem um conjunto de demonstrações que ilustram as suas possíveis aplicações. Para aceder à demonstração basta entrar o comando:
 - »demo.
 - Gráficos de funções
 - Visualização de volumes
 - Animações
 - Tutoriais sobre o Matlab



O Ambiente Gráfico

Ajuda

The image shows the MATLAB R2024b - academic use interface. The top menu bar includes HOME, PLOTS, and APPS. The ribbon contains various toolbars: FILE (New Script, New Live Script, New, Open, Find Files, Compare), VARIABLE (Import Data, Clean Data, Save Workspace, Clear Workspace), CODE (Run and Time, Analyze Code, Clear Commands), ENVIRONMENT (Layout, Set Path, Add-Ons), and RESOURCES (Help, Community, Request Support, Learn MATLAB). The main workspace is divided into several panes: Current Folder (showing Examples and startup.m), Command Window (with a prompt fx >> |), Workspace (showing Name and Value columns), and Details. Annotations with arrows point to specific features: 'Ajuda' points to the Help icon in the RESOURCES toolbar; 'Para mudar a pasta de trabalho' points to the Current Folder pane; 'Conteúdo da pasta de trabalho' points to the file list in the Current Folder pane; 'Janela de comandos (CW)' points to the Command Window pane; and 'Espaço de trabalho com as variáveis' points to the Workspace pane.



Ajuda

- `help` : Informação na janela de comando sobre um comando ou operador

```
Command Window

>> help sin
sin - Sine of argument in radians
This MATLAB function returns the sine of the elements of X.

Syntax
    Y = sin(X)

Input Arguments
    X - Input angle in radians
        scalar | vector | matrix | multidimensional array | table |
        timetable

Output Arguments
    Y - Sine of input angle
        scalar | vector | matrix | multidimensional array | table |
        timetable

Examples
    Plot Sine Function
    Sine of Vector of Complex Angles

See also sind, asin, asind, sinh, sinpi

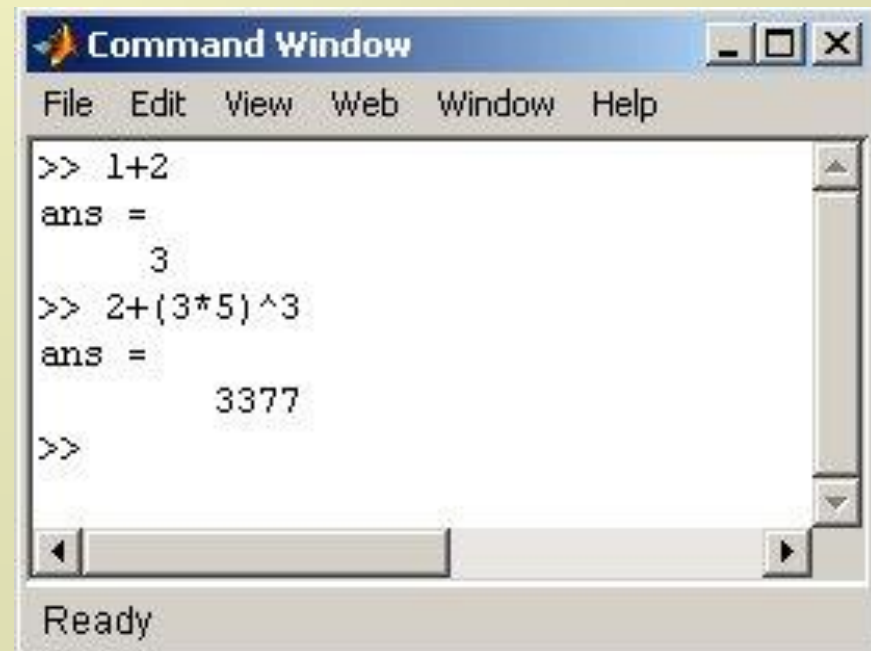
Introduced in MATLAB before R2006a
Documentation for sin
Other uses of sin

fx >>
```




Variáveis no Matlab

- O Matlab como calculadora
 - O Matlab permite o cálculo numérico directo a partir da janela de comando.
- Operações matemáticas
 - + soma
 - subtracção
 - * multiplicação
 - / divisão
 - ^ potenciação





Variáveis no Matlab

- Variáveis
 - No Matlab é possível guardar em variáveis conjuntos de números, exemplo:
»x= 2
 - Os nomes das variáveis distinguem as letras maiúsculas das minúsculas. Exemplo: $\pi \neq Pi$
 - As variáveis são guardadas no espaço de trabalho “workspace”
 - As variáveis podem ser utilizadas nas operações da mesma forma que os números.
 - Se não for atribuída nenhuma variável o resultado é armazenado numa variável temporária “**ans**”



Variáveis no Matlab

- Números complexos
 - O Matlab permite a representação de números complexos.
Para criar o número complexo

$$1 + 2i \quad \text{ou} \quad 1 + 2j$$

basta introduzir na janela de comandos:

```
>> 1 + 2i
```

ou

```
>> 1 + 2 * i
```



Variáveis no Matlab

- Regras para atribuição de nomes de variáveis
 - Aceitam-se no máximo 31 caracteres (letras, números , “_”)
 - O primeiro caracter tem de ser uma letra.

Exemplos

a1=2 , soma1=10

a_1=2, soma_2=20

Alguns erros frequentes

1a=2, 1_a=2

a 1=2, aula_nº1



Funções matemáticas

- O Matlab dispõe dum vasto conjunto de funções matemáticas. Alguns exemplos:

cos	co-seno (radianos)	log	logaritmo neperiano (base e)
sin	seno	log10	logaritmo base 10
tan	tangente	rem	resto da divisão inteira
acos	arco co-seno	abs	valor absoluto
asin	arco seno	sign	sinal
atan	arco tangente	round	arredondamento para o mais próximo
sqrt	raiz quadrada	floor	arredondamento para baixo
exp	exponencial	ceil	arredondamento para cima



Vectores

- No Matlab para criar um vector “**v**” basta fazer por exemplo:

» **v = [4 , 5 , 4 , 2 , 1 , 7]**

1 2 3 4 5 6

Índices

- Os elementos são separados por espaços ou vírgulas
 - Num vector coluna os elementos são separados por “ ; ”
- Os índices são números inteiros e começam sempre pelo número 1.



Vectores

- Para obter um elemento do vector escreve-se no CW

```
>> V(3) ; ans=4
```

- Para substituir um elemento do vector escreve-se no CW

```
>> V(2) =10
```

$V = [4, 10, 4, 2, 1, 7]$

- **Atenção**

Novo elemento

```
>> V(0)
```

- ??? Subscript indices must either be real positive integers or logicals.

Matrizes

- Definição

- Organização bidimensional de dados
- Estrutura de dados primária em MATLAB
- Tabela de valores com m linhas e n colunas
- Extensão do conceito de vector

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

m vectores linha

n vectores coluna

Dimensão

$M \times N$



Matrizes

- Temperaturas registadas durante uma semana

$$T = \begin{bmatrix} 10 & 11 & 10 & 9 & 10 & 11 & 10 \\ 7 & 8 & 8 & 6 & 7 & 9 & 7 \\ 22 & 24 & 22 & 18 & 22 & 18 & 24 \\ 18 & 19 & 18 & 16 & 17 & 16 & 19 \end{bmatrix}$$

↓ Medidas / dia

→ Dia da semana



Matrizes

No Matlab para criar uma matriz “**A**” basta fazer por exemplo:

```
» A= [4 , 5 , 4 ; 2 , 1 , 7]
```

Separação
entre linhas

Os elementos em cada linha são separados por espaços ou vírgulas e a separação entre linhas por “ ; ”

Matrizes

Exemplos:

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

4x4

Matriz

$$v = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix}$$

4x1

Vector
coluna

$$v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

1x4

Vector
linha

$$n = \begin{bmatrix} 7 \end{bmatrix}$$

1x1



Matrizes

Índices das
linhas

Índices:

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \end{bmatrix}$$

(3×4)

Índices das
colunas

1 2 3 4

Matrizes

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \end{bmatrix}$$

The matrix A is a 3x4 matrix. The element 10 is circled in red. A green arrow points from the label $A_{2,3}$ to the element 10. The columns are indexed 1 to 4, and the rows are indexed 1 to 3.

16	2	3	13	1
5	11	10	8	2
9	7	6	12	3
1	2	3	4	

$A_{2,3}$

Matrizes

- Para obter um elemento da matriz escreve-se no CW

```
>> A(2,3) ; ans=10
```

- Para substituir um elemento da matriz escreve-se no CW

```
>> A(2,2) = -20
```

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & -20 & 10 & 8 \\ 9 & 7 & 6 & 12 \end{bmatrix}$$

- **Atenção**

```
>> A(2,5)
```

•??? Index exceeds matrix dimensions.

Novo elemento



Geração de uma sequência de números

- O operador “:”
 - O mais versátil operador do MATLAB
 - Permite definir de forma compacta um conjunto de valores (vector) em progressão aritmética.
- » `% x = valor inicial : passo : valor final;`
- » `% Nota: argumentos de “:” não podem ser complexos`
- » `x = 1:10;`
- » `x = -pi : 2*pi/359 : pi;`
- » `x = 100:-2:80;`
- O recurso ao “:” não obriga à delimitação por []



Geração de uma sequência de números

- Função Linspace

- » % Quando sabemos os limites numéricos da sequência
- » % xi e xf e o número de elementos N então devemos
- » % recorrer à função
- » **`x = linspace(xi,xf,N) ;`**
- » % Espaçamento linear (uniforme) entre os elementos
- » de x. Evita-se o cálculo do passo.
- » **`x = linspace(10,-10,5)`**

x =

10	5	0	-5	-10
----	---	---	----	-----



Indexação de matrizes

- O operador “:” revela-se um poderoso meio de indexação.

```
>> x = 1:2:50;
```

```
>> x(10:15)
```

```
ans =
```

```
19    21    23    25    27    29
```

- Vectores de índices

```
>> v1 = 10:15;
```

```
>> x(v1)
```

```
ans =
```

```
19    21    23    25    27    29
```



Indexação de matrizes

```
>> A = magic(4)
```

```
A =
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

```
>> B = A(:,1:2:end) %B é composta pelas colunas ímpares de A
```

```
B =
```

16	3
5	10
9	6
4	15

```
>> last = A(end,end); % Última linha, última coluna
```

```
last = 1
```



Exercícios

- Elabore um “script” Matlab que resolva os seguintes problemas:
 - Gere uma sequência de números ímpares entre 1 e 10
 - Gere uma sequência de 11 números inteiros entre -5 e 5.
(resolva de 2 formas)
 - Gere a seguinte matriz
$$A = \begin{bmatrix} 1 & 5 & 1-j \\ 4 & j & -1 \end{bmatrix}$$
 - Acrescente uma nova linha e coluna à matriz A
 - Apague as colunas ímpares.



Criação de Matrizes

- Quando se pretende criar uma matriz cujos elementos se podem relacionar facilmente, o Matlab possui as seguintes funções:

– **zeros(N,M)** gera uma matriz de zeros com N linha e M colunas

Exemplo : `>> A=zeros (2, 3)`

A =

0 0 0

0 0 0

– **ones(N,M)** gera uma matriz de uns com N linhas e M colunas



Criação de Matrizes

- **rand(N,M)** gera uma matriz de elementos aleatórios com N linha e M colunas (distribuição uniforme)
- **randn(N,M)** gera uma matriz de elementos aleatórios com N linha e M colunas (distribuição normal)
- **magic(N)** gera um quadrado mágico de dimensão N
- **eye(N)** gera uma matriz identidade de dimensão N

Exemplo:

```
>> A=eye (2)
```

A =

1 0

0 1



Criação de Matrizes

- O Matlab permite também gerar automaticamente algumas matrizes especiais

<code>compan</code>	- Companion matrix.
<code>hadamard</code>	- Hadamard matrix.
<code>hankel</code>	- Hankel matrix.
<code>hilb</code>	- Hilbert matrix.
<code>invhilb</code>	- Inverse Hilbert matrix.
<code>magic</code>	- Magic square.
<code>pascal</code>	- Pascal matrix.
<code>toeplitz</code>	- Toeplitz matrix.
<code>vander</code>	- Vandermonde matrix.



Criação de Matrizes

- Mais funções relacionadas com matrizes

<code>size</code>	- Size of array.
<code>length</code>	- Length of vector.
<code>numel</code>	- Number of elements.
<code>cat</code>	- Concatenate arrays.
<code>diag</code>	- Diagonal matrices and diagonals of matrix.
<code>blkdiag</code>	- Block diagonal concatenation.
<code>tril</code>	- Extract lower triangular part.
<code>triu</code>	- Extract upper triangular part.
<code>fliplr</code>	- Flip matrix in left/right direction.
<code>flipud</code>	- Flip matrix in up/down direction.
<code>flipdim</code>	- Flip matrix along specified dimension.
<code>rot90</code>	- Rotate matrix 90 degrees.
<code>find</code>	- Find indices of nonzero elements.
<code>end</code>	- Last index.

Verifique no Help o que cada uma faz



Operações com Matrizes

- Soma algébrica com entidades escalares é extensível a vectores e matrizes desde que as **dimensões sejam idênticas**.

```
>> A = rand(3);
```

```
>> B = magic(3);
```

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

```
>> C = A - B;
```

- Soma e multiplicação com valor escalar

```
>> D = 5 + B; E = 5*B;
```

```
>> F = 7 + 3*B - 12*A;
```

```
>> F = (2 + 2j)*ones(3);
```




Operações com Matrizes

- Cada elemento $Y_{i,j}$ da matriz resulta do produto interno entre a linha A_i e a coluna B_j
- Só podem realizar-se produtos de matrizes AB tal que o n° de colunas $A =$ n° de linhas de B

$$\underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}}_A \underbrace{\begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix}}_B = \underbrace{\begin{bmatrix} y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mk} \end{bmatrix}}_Y$$

$$Y_{ij} = \langle A_i, B_j \rangle \quad i = 1, \dots, m \quad j = 1, \dots, k$$

Operações com Matrizes

$$\begin{matrix} 2 \\ \left[\begin{array}{ccc} 1 & 3 & 2 \\ 3 & 4 & 5 \\ 5 & 2 & 6 \end{array} \right] \end{matrix} \times \begin{matrix} 1 \\ \left[\begin{array}{ccc} 1 & 1 & 2 \\ 7 & 3 & 9 \\ 3 & 1 & 7 \end{array} \right] \end{matrix} = \begin{matrix} \left[\begin{array}{ccc} 28 & 12 & 43 \\ 46 & 20 & 77 \\ 37 & 17 & 70 \end{array} \right] \end{matrix}$$

(2,1)

$$3 \times 1 + 4 \times 7 + 5 \times 3 = 46$$



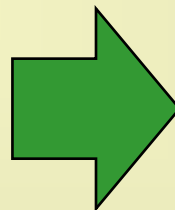
Operações com Matrizes

- Exemplos

```
>> A = magic(3)
```

A =

8	1	6
3	5	7
4	9	2



```
>> B = [3; 6; 7]
```

```
>> U = round(10*rand(3))
```

U =

9	4	4
7	9	9
2	9	1

```
>> Y = A*B
```

Y =

72

88

80

```
>> C = A*U
```

C =

91 95 47

76 120 64

103 115 99

```
>> D = Y*C
```

??? Error using==> mtimes
Inner matrix dimensions
must agree.



Operações com Matrizes

- Mais exemplos

```
>> A = magic(3)
```

A =

8	1	6
3	5	7
4	9	2

```
>> A2 = A^2
```

A2 =

91	67	67
67	91	67
67	67	91

Atenção

```
>> A = rand(3,2);
```

```
>> A^2
```

??? Error using ==> ^
Matrix must be square.



Operações com Matrizes

- Dimensões

- Número de elementos dum vector ou matriz

```
>> x = 1:10;
```

```
>> A = rand(3,2);
```

```
>> dim_x = size(x),    dim_y = size(A)
```

```
>> dim_x =          dim_y =  
          1      10          3      2
```

- Várias formas de calcular o número de elementos de uma matriz

```
>> nelementos = prod(size(A)); ou
```

```
>> nelementos = numel(A); ou
```

```
>> nelementos = length(A(:))
```



Manipulação Matrizes

- Transposta de uma matriz
 - A operação de transposição troca as linhas pelas colunas de uma matriz. Em notação matemática a transposta de uma matriz \mathbf{A} representa-se por \mathbf{A}^T . Em notação Matlab a transposta de uma matriz representa-se por $\mathbf{A}.'$
- Exemplo:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

- Um vector linha pode ser transformado em coluna através da transposta



Manipulação Matrizes

- Hermitiana de uma matriz
 - É a transposta conjugada de uma matriz \mathbf{A} e geralmente é representada por $(\mathbf{A}^*)^T = \mathbf{A}^H$
 - Em notação Matlab a hermitiana de uma matriz representa-se por \mathbf{A}' . Se \mathbf{A} é real então $\mathbf{A}^T = \mathbf{A}^H \implies \mathbf{A}' = \mathbf{A}$. '
- Exemplo:

$$\mathbf{A} = \begin{bmatrix} 1+j & 1 & 1 \\ 2 & 2+2j & 2 \\ 3 & 3 & 3-j \end{bmatrix} \quad \mathbf{A}^H = \begin{bmatrix} 1-j & 2 & 3 \\ 1 & 2-2j & 3 \\ 1 & 2 & 3+j \end{bmatrix}$$



Manipulação Matrizes

– `reshape (A,N,M)`

Cria uma nova matriz de dimensão $N \times M$, a partir dos elementos de A, retirados em coluna

Exemplos:

```
>> A=reshape ([1 2; 3 4],1,4)
```

A =

1 3 2 4

```
>> A=reshape ([1 2; 3 4],2,3)
```

??? Error using ==> reshape
To RESHAPE the number of elements
must not change

A nova matriz tem que
ter a mesma dimensão
da inicial



Manipulação Matrizes

- É possível criar matrizes maiores a partir de outras mais pequenas- Concatenação :

– **repmat (A,N,M)**
e

Repete a matriz A, N vezes em linha
e M vezes em coluna.

Exemplo:

```
>> A=repmat ([1 2; 3 4], 2, 2)
```

A =

1	2	1	2
3	4	3	4
1	2	1	2
3	4	3	4

– **Manualmente**

Exemplo:

```
>> A1=[1 2; 3 4]    A2=[-1 -3; 3 2]
```

```
>> A=[A1 A2] ou A=[A1; A2]
```

Exercício

- Crie um vector S com os primeiros 200 inteiros. De seguida crie uma matriz M com 3 colunas de acordo com as seguintes especificações: 1ª coluna deve ser igual a S , 2ª coluna deve conter os elementos de S por ordem decrescente e a 3ª coluna deverá conter a média das duas primeiras.
- Gere uma matriz, A , com 10 linhas e 10 colunas e com os valores seguintes (não utilize a definição exaustiva da matriz):

$$A = \begin{bmatrix} 1 & 2 & \dots & 10 \\ 11 & 12 & \dots & 20 \\ \dots & \dots & \dots & \dots \\ 91 & 92 & \dots & 100 \end{bmatrix}$$



Operações “elemento a elemento”

- Por vezes estamos interessados em operações “elemento-a-elemento” em vez de matriciais
- Para efetuar uma operação “elemento-a-elemento” usa-se o “.” antes de cada operação (multiplicação, divisão, potenciação, etc)

Multiplicação (x),
divisão (/), etc

$(X) \cdot \text{operação}(Y)$

Indica operação elemento a elemento

A dimensão de X tem que ser a mesma de Y



Operações “elemento a elemento”

- Exemplos

```
>> x = [1 2 3 4]; y = [2 2 10 10];
```

```
>> p = x .* y % Multiplicação elem. a elem.
```

```
p =
```

```
      2      4     30     40
```

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

```
>> p = A .* A
```

```
P =
```

```
      1      4      9     16
```

```
     25     36     49     64
```

```
>> p = A .* y
```

??? Error using ==> Matrix dimensions must agree.



Operações “elemento a elemento”

- Mais exemplos

```
>> A = [2 4 ; 6 9]; B = [1 2 ; 2 3];
```

```
>> C = A .^2 % Potenciação elem. a elem.
```

B =

4	16
36	81

```
>> D = A ./B % Divisão elem. a elem.
```

D =

2	2
3	3



Operações “elemento a elemento”

- Calculo de funções elaboradas
 - É possível decompor uma função num conjunto de funções elementares mais simples. Vejamos o seguinte exemplo

$$f(x) = \sin(x) \cos(x^2).$$

- Podemos decompor a função anterior no produto das funções $\sin(x)$ e $\cos(x)$. O produto entre estas duas funções deverá ser realizado **elemento-a-elemento**.



Operações “elemento a elemento”

- Decomposição das operações para calcular

As operações são realizadas elemento-a-elemento

x é um vector

$$f(x) = \sin(x) \cos(x^2).$$

$$\mathbf{f} = \sin(\mathbf{x}) .* \cos(\mathbf{x}.^2)$$

x	sin(x)	x.^2	cos(x.^2)	sin(x) .* cos(x.^2)
0	0	0	1	0
pi/10	0.3090	0.0987	0.9951	0.3075
2pi/10	0.5878	0.3948	0.9231	0.5426
3pi/10	0.8090	0.8883	0.6308	0.5103
pi	0	9.8696	-0.9027	0



Indexação lógica

- Em muitas situações pretende-se referenciar os elementos de uma matriz que satisfazem uma dada condição. Por exemplo, dado o vector

$$\mathbf{x} = [1 \ 2 \ -1 \ 3 \ -3]$$

como se pode gerar um outro que apenas contenha os elementos menores que zero?

- Se fizer $\mathbf{x} < 0$ obtêm-se o seguinte vector lógico

$$0 \ 0 \ 1 \ 0 \ 1$$

Este vector pode ser utilizado para indexar os elementos de \mathbf{x}

$$\begin{aligned} &\mathbf{x}(\mathbf{x} < 0) \\ &-1 \ -3 \end{aligned}$$

Indexação lógica

- Índices lógicos em Matlab

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



Indexação lógica

- O Matlab tem uma função designada por **FIND** que também permite indexação lógica.

Sintaxe

find(cond.logica)

% Encontra os índices
dos elementos do
vector que satisfazem
a condição lógica.

Exemplo

```
>> x= [1 2 -1 3 -3]
```

```
>> n=find(x<0) ;      y=x(n)
```

```
>> n =                y=
```

3

5

-1 -3



Gráficos elementares

- A sintaxe da função **PLOT** para fazer um gráfico de coordenadas (x,y) é

plot(**x**,**y**)

% coloca **x** nas abcissa e **y** nas ordenadas

Exemplos

```
>>x=linspace(0,2*pi,200);
```

```
>> y=sin(2*x);
```

```
>> plot(x,y)
```

```
>> z=rand(1,100);
```

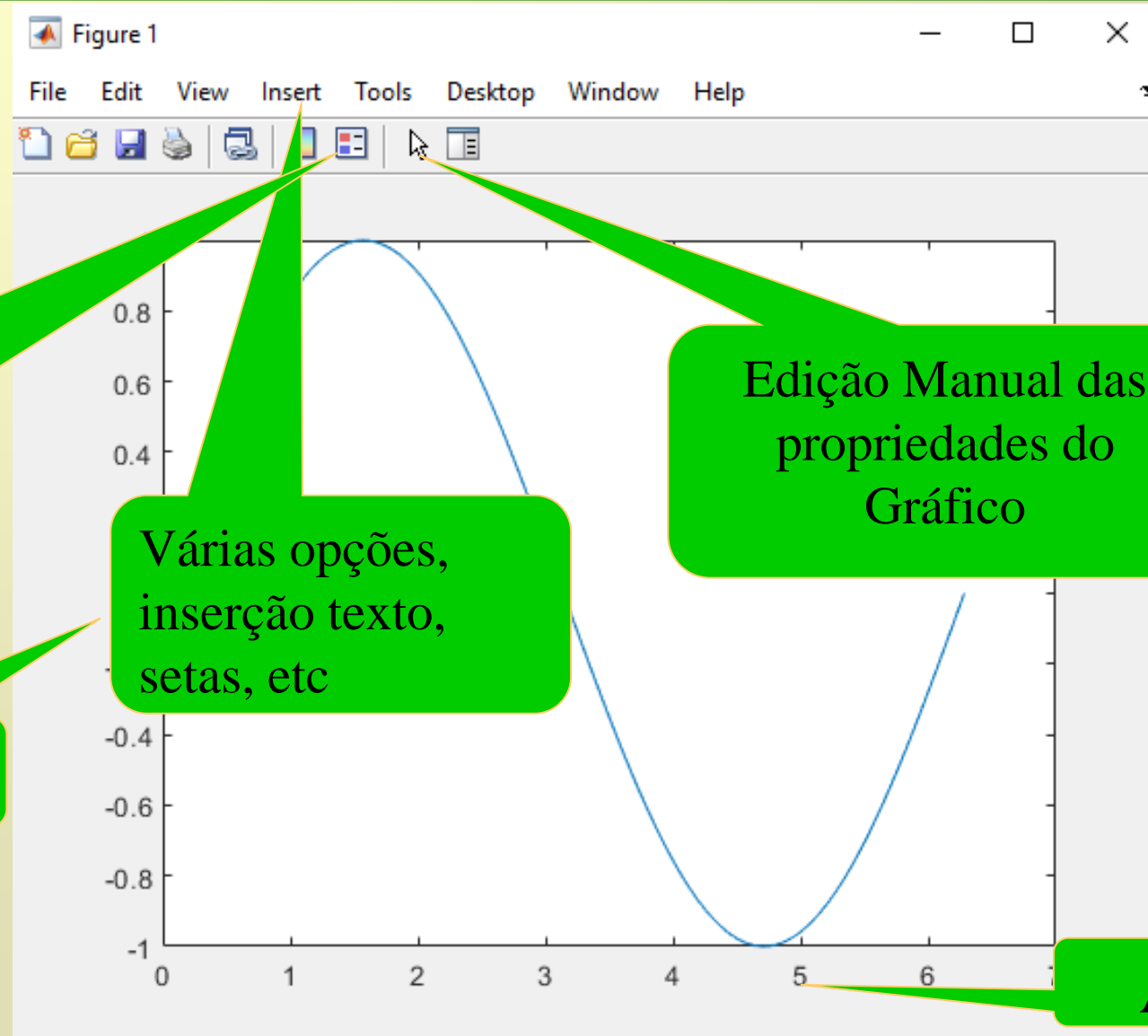
```
>>plot(x,z)
```

Atenção!

Os vectores **x** e **z** têm que ter o mesmo número de pontos

??? Error using ==> plot , Vectors must be the same lengths.

Figura do Matlab





Gráficos elementares

- Alteração do aspecto gráfico
 - Para além dos argumentos vectoriais a função **plot** permite ainda alterar o modo como as linhas são desenhadas. Essas indicações são codificadas na forma de uma “string” de texto colocada a seguir aos vectores dos pontos.

plot(x,y,'string')

- A “**string**” pode definir os seguintes atributos das linhas desenhadas
 - Marcadores dos pontos do gráfico
 - Cor das linhas e marcadores
 - Tipo de linha a desenhar

Gráficos elementares

- Caracteres definidores de atributos

Cor

y	amarelo
m	rosa
c	azul claro
r	encarnado
g	verde
b	azul
w	branco
k	preto

Marcadores

.	ponto
o	círculo
x	marca x
+	marca mais
*	estrela
s	quadrado
d	diamante
v	triângulo (cima)
^	triângulo (baixo)
<	triângulo (esquerda)
>	triângulo (direita)
p	pentagrama
h	"hexagram"

Linhas

-	linha a cheio
:	pontuada
-.	traço ponto
--	tracejada

Gráficos elementares

- Título, labels dos eixos, legenda

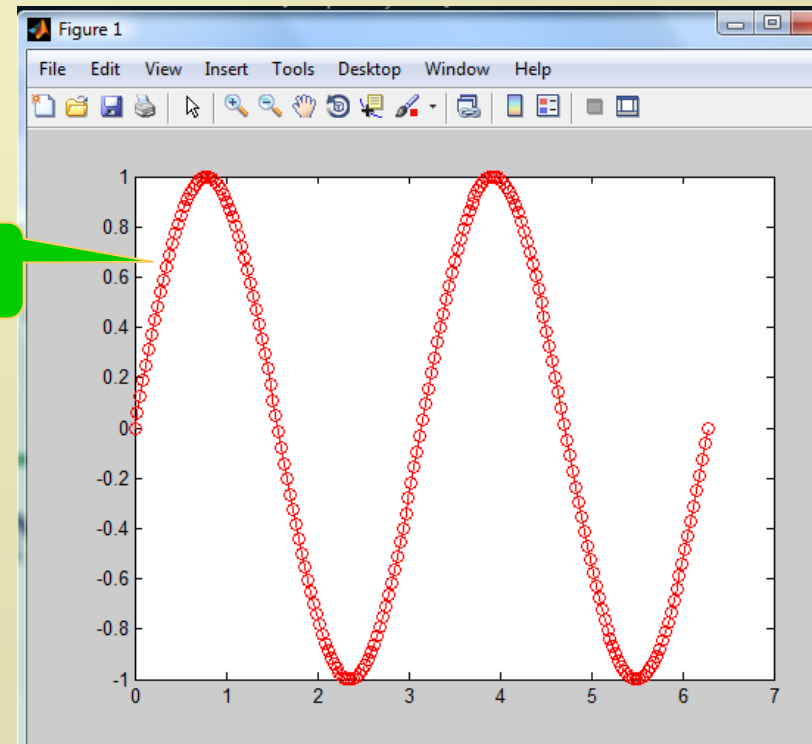
- Na sequência do plot anterior:

```
>>x=linspace(0,2*pi,200);
```

```
>> y=sin(2*x);
```

```
>> plot(x,y,'o-r');
```

plot(x,y,'o-r')



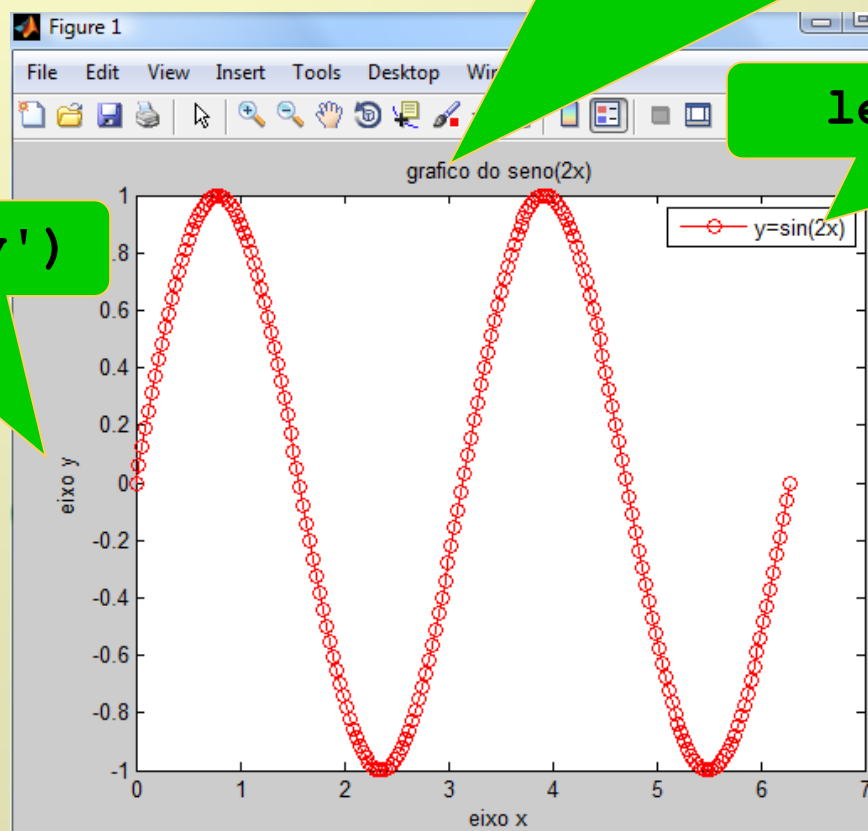
Gráficos elementares

- Título, labels dos eixos, legenda

`title('grafico do seno(2x)')`

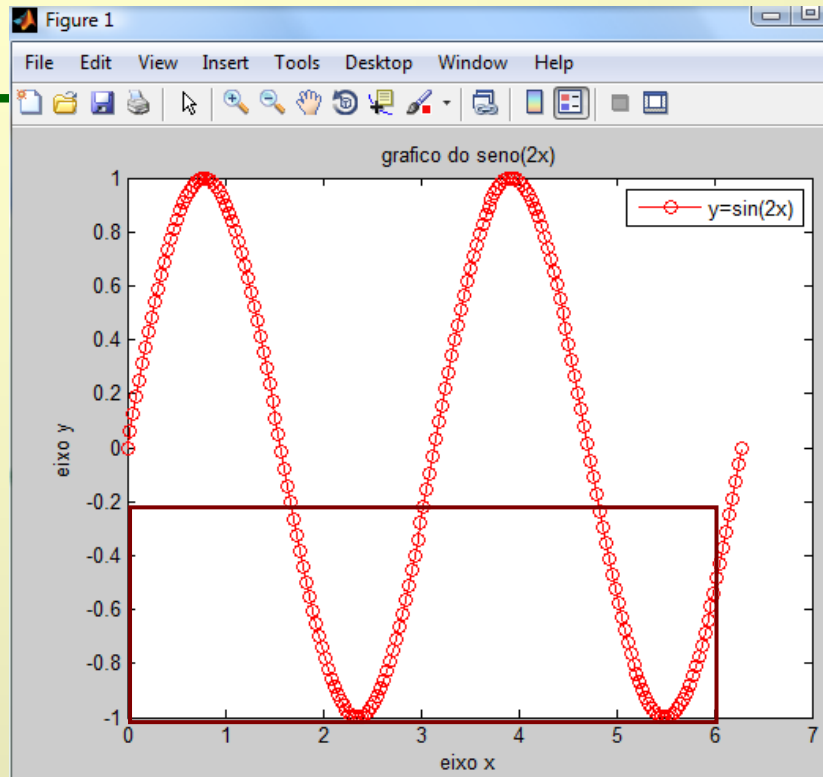
`legend('y=sin(2x)')`

`ylabel('eixo y')`

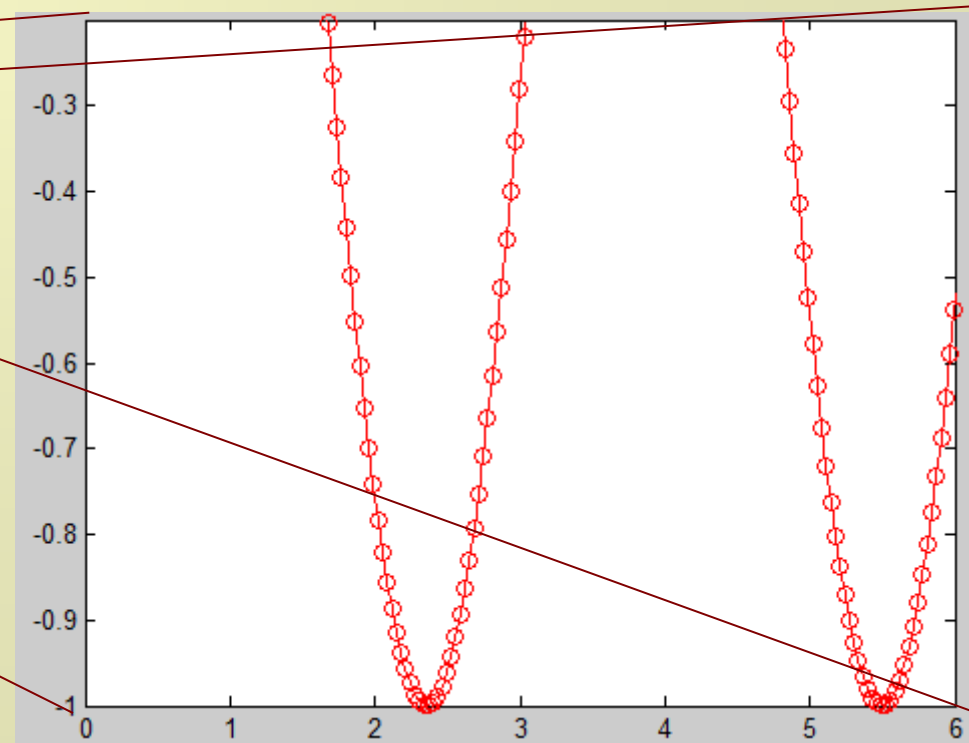


`xlabel('eixo x')`

Eixos – Função **axis**



`axis([0 6 -1 -0.2])`





Gráficos elementares

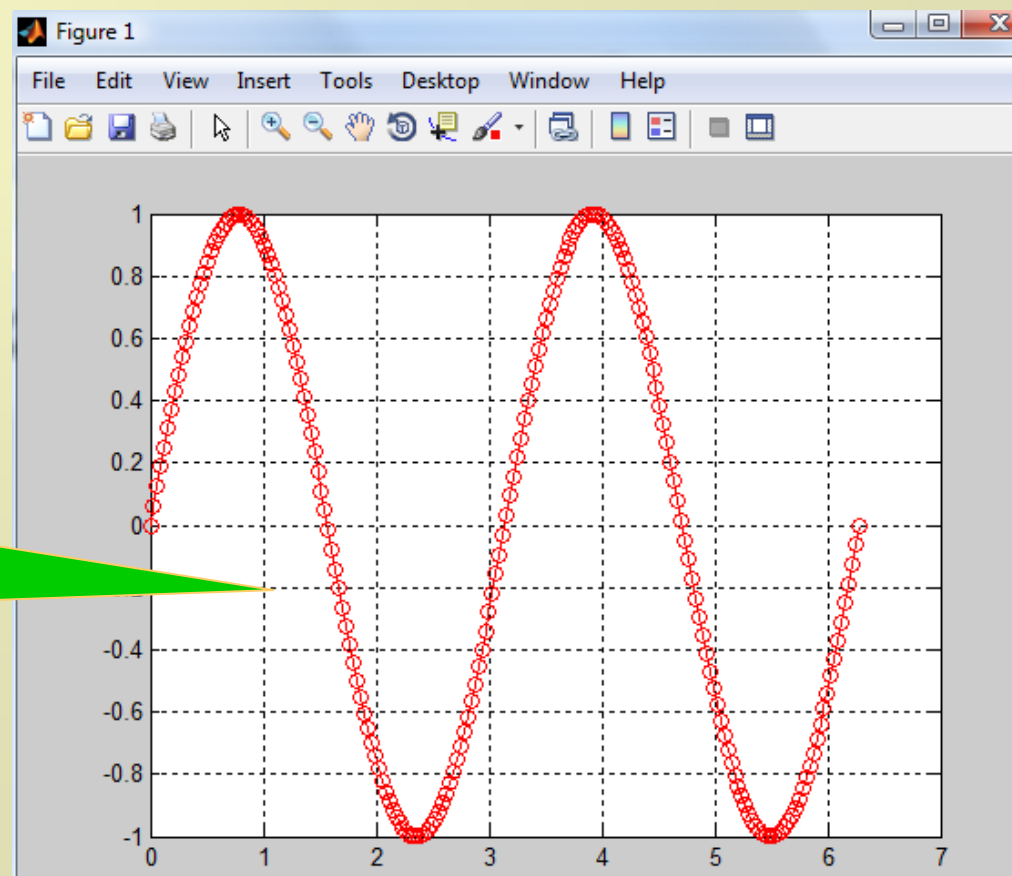
- Sintaxe da função **axis**

axis ([xmin xmax ymin ymax])

Gráficos elementares

- Também é possível colocar uma grelha nas figuras geradas pelo Matlab
 - Funções **grid on**, **grid off**

Grelhas
Vantagem: permitem
uma leitura mais
fácil





Exercício

- Suponha que pretendia visualizar o gráfico da seguinte função matemática

$$y = \sin(x)e^x$$

quando a variável x pertence a $[-2\pi, 0]$ (considere 200 elementos para o vetor x distribuídos uniformemente no intervalo indicado)

- Faça o gráfico da função
- a linha deve ser a ponteadado **vermelho**
- o eixo das abcissas deve ter a gama $[-8, 0]$ e o eixo das ordenadas $[-0.5, 0.8]$
- documente o gráfico (título, labels nos eixos)



Sobreposição de funções no mesmo gráfico

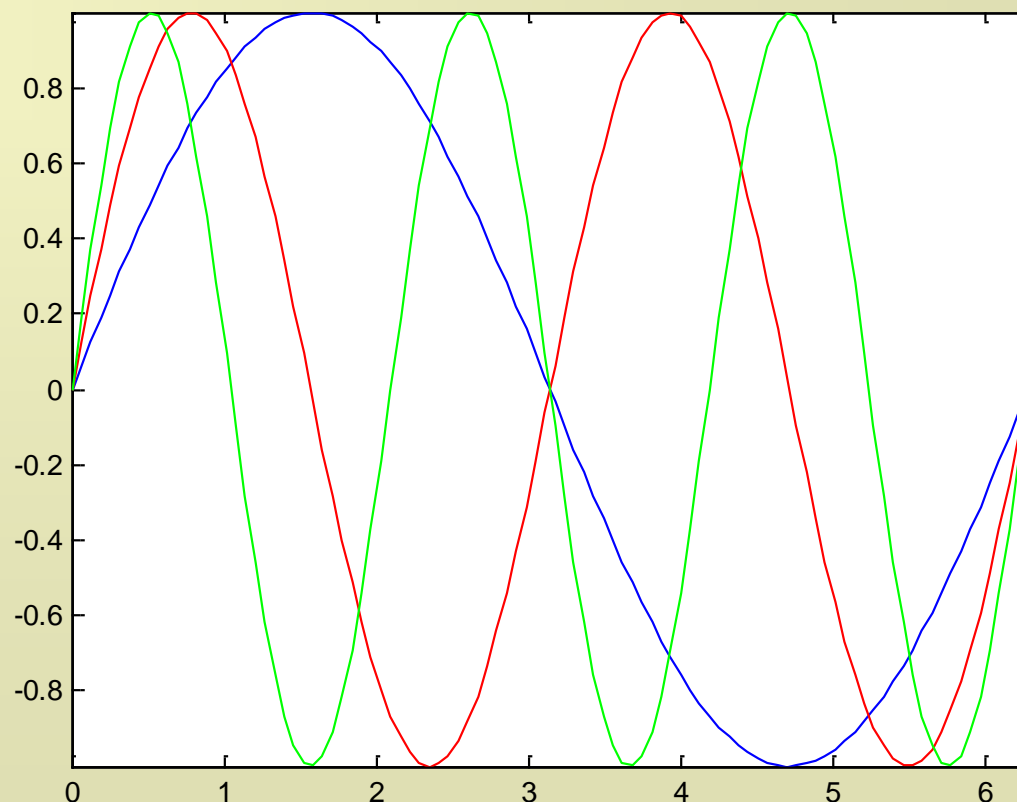
- Em Matlab existem várias formas de sobrepor curvas na mesma janela gráfica
 - Colocar na função **plot** todas as curvas que se pretende representar
plot(x1,y1,x2,y2,...)
plot(x1,y1,'string1',x2,y2,'string2',...)
 - O número de elementos dos pares **(x1,y1)** e **(x2,y2)** deve ser o mesmo.
 - No entanto o número de elementos do par **(x1,y1)** pode ser diferente do par **(x2,y2)**



Sobreposição de funções no mesmo gráfico

- Também se pode usar as funções **Hold on /off** que activa/desactiva a sobreposição de gráficos sobre a mesma janela

```
>> x = linspace(0,2*pi);  
>> plot(x,sin(x))  
>> hold on  
>> plot(x,sin(2*x),'r')  
>> plot(x,sin(3*x),'g')  
>> hold off  
>> % limpa a figura  
>> % corrente e imprime  
>> % novo gráfico  
>> plot(x,sin(4*x))
```





Exercícios (1)

- Considere a função do tempo $B(t)=g(t).\sin(6\pi t)$, $t \in [-4,4]$ e

$$g(t) = \frac{1}{\sqrt{2\pi}} t^2 e^{-\frac{t^2}{2}}$$

- Calcule $g(t)$ e $B(t)$ utilizando um passo de amostragem de 0.01s
- Produza os gráficos de $B(t)$ e $g(t)$ sobrepostos numa só figura; utilize para tal dois procedimentos, recorrendo ou não a hold on. A curva de $B(t)$ deverá ser a **azul** e a de $g(t)$ a **vermelho**. Os eixos devem estar confinados ao intervalo $[-4,4]$ para as abcissas e $[-0.4,0.4]$ para as ordenadas. Documente devidamente o gráfico.



Subgráficos

- Objectivo:
 - Desenhar numa única janela vários gráficos
- Vantagens
 - Permite comparar mais facilmente diferentes gráficos
 - Figuras mais compactas

subplot (L , C , N)

L – Número de subgráficos na vertical

C – Número de subgráficos na horizontal

N – Número do subgráfico para onde são desenhados os gráficos

Subgráficos

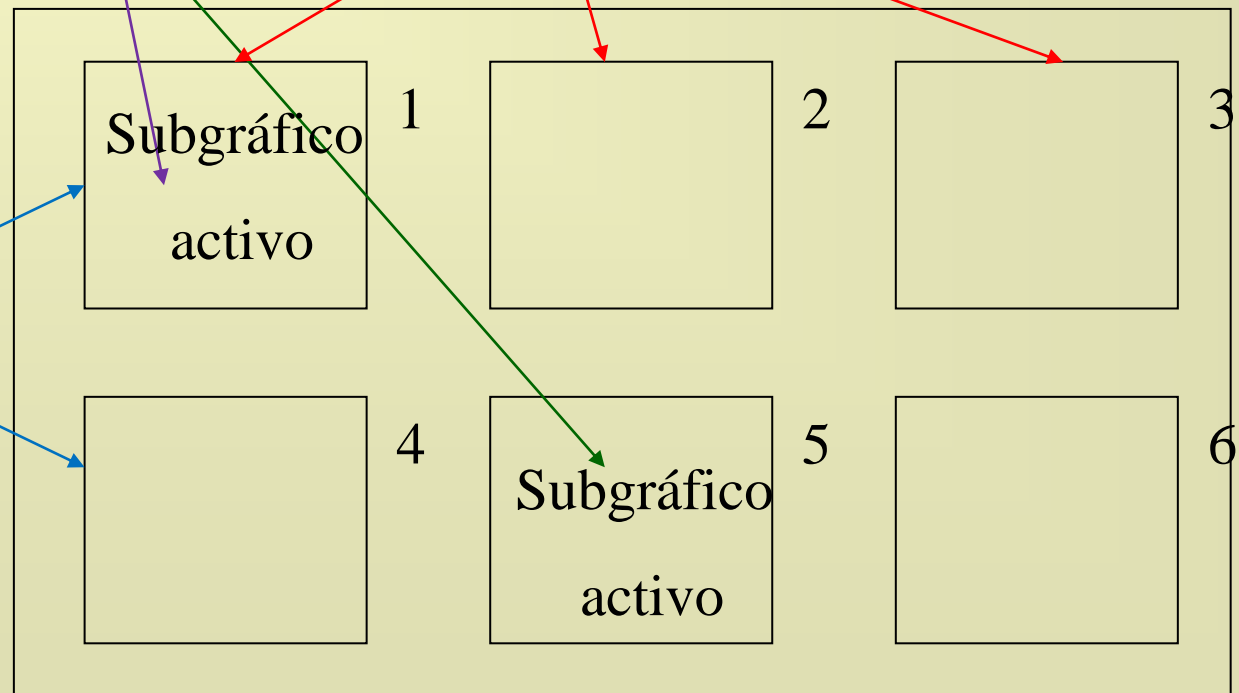
- Definição de uma matriz de subáreas gráficas

```
>> subplot(2,3,1)
```

```
>> subplot(2,3,5)
```

3 colunas

2 linhas





Subgráficos

- **Exemplo**

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

```
>> subplot(1,3,1);
```

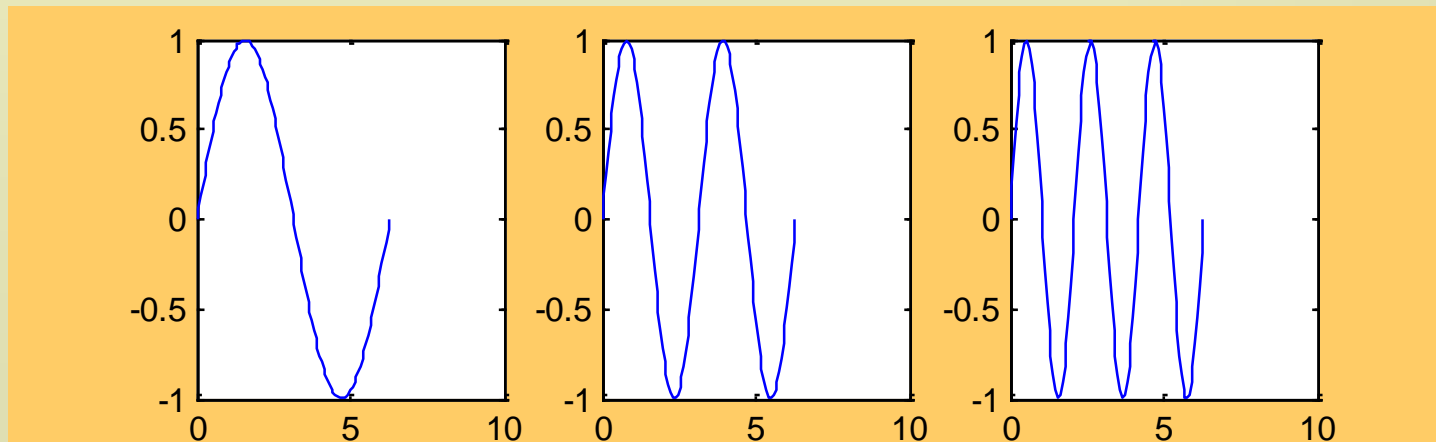
```
>> plot(w,sin(w))
```

```
>> subplot(1,3,2);
```

```
>> plot(w,sin(2*w))
```

```
>> subplot(1,3,3);
```

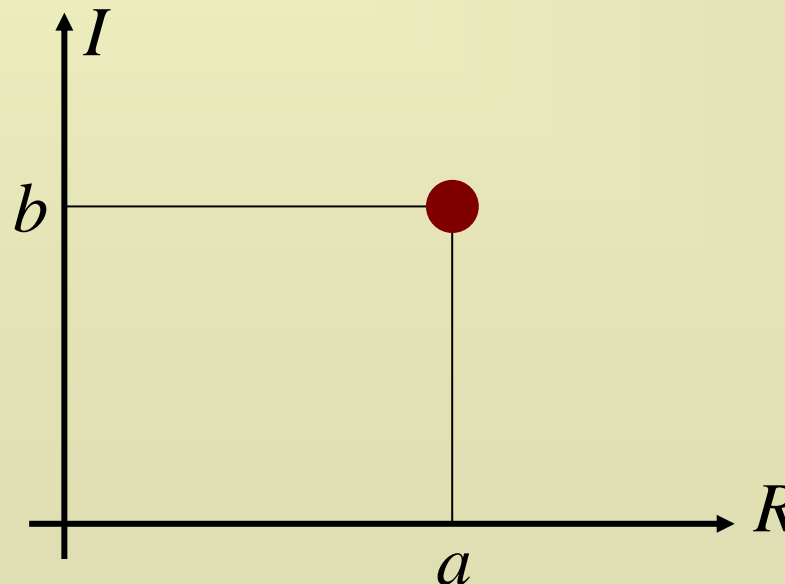
```
>> plot(w,sin(3*w))
```





Gráficos de variáveis complexas

- Um número complexo é da forma
$$\mathbf{a+jb}$$
- E pode ser representado num plano complexo como um ponto





Gráficos de variáveis complexas

- Um número complexo possui duas dimensões sendo equivalente a um vector com duas dimensões
- O código Matlab em baixo desenha o mesmo ponto no plano.

```
>> x= 1; y= 2
```

```
>> z=x+j*y;
```

```
>> plot(z,'o')
```

```
>> plot(x,y,'o')
```



Gráficos de variáveis complexas

- **Exemplo**

- Considere-se a seguinte função complexa

$$z = e^{iw}$$

pela fórmula de Euler, podemos escrever a equação anterior da seguinte forma

$$z = \cos w + i \sin w$$

- A função é periódica (com período 2π), pelo que basta gerar pontos para x entre 0 e 2π para se observar um período completo da função.



Gráficos de variáveis complexas

```
>> w=linspace(0,2*pi,200);  
>> z=exp(i*w);  
>> x=real(z);y=imag(z);ph=phase(z);m=abs(z);  
>> subplot(3,2,1),plot(w,x),xlabel('w'),ylabel('real(z)')  
>> subplot(3,2,2),plot(w,y),xlabel('w'),ylabel('imag(z)')  
>> subplot(3,2,3),plot(w,ph),xlabel('w'),ylabel('fase(z)')  
>> subplot(3,2,4),plot(w,m),xlabel('w'),ylabel('modulo(z)')  
>> subplot(3,2,[5,6]),plot(x,y)
```

Ou



```
>> subplot(3,2,[5,6]),plot(z)  
>> xlabel('real'),ylabel('imag')  
>> axis square
```



Exercício

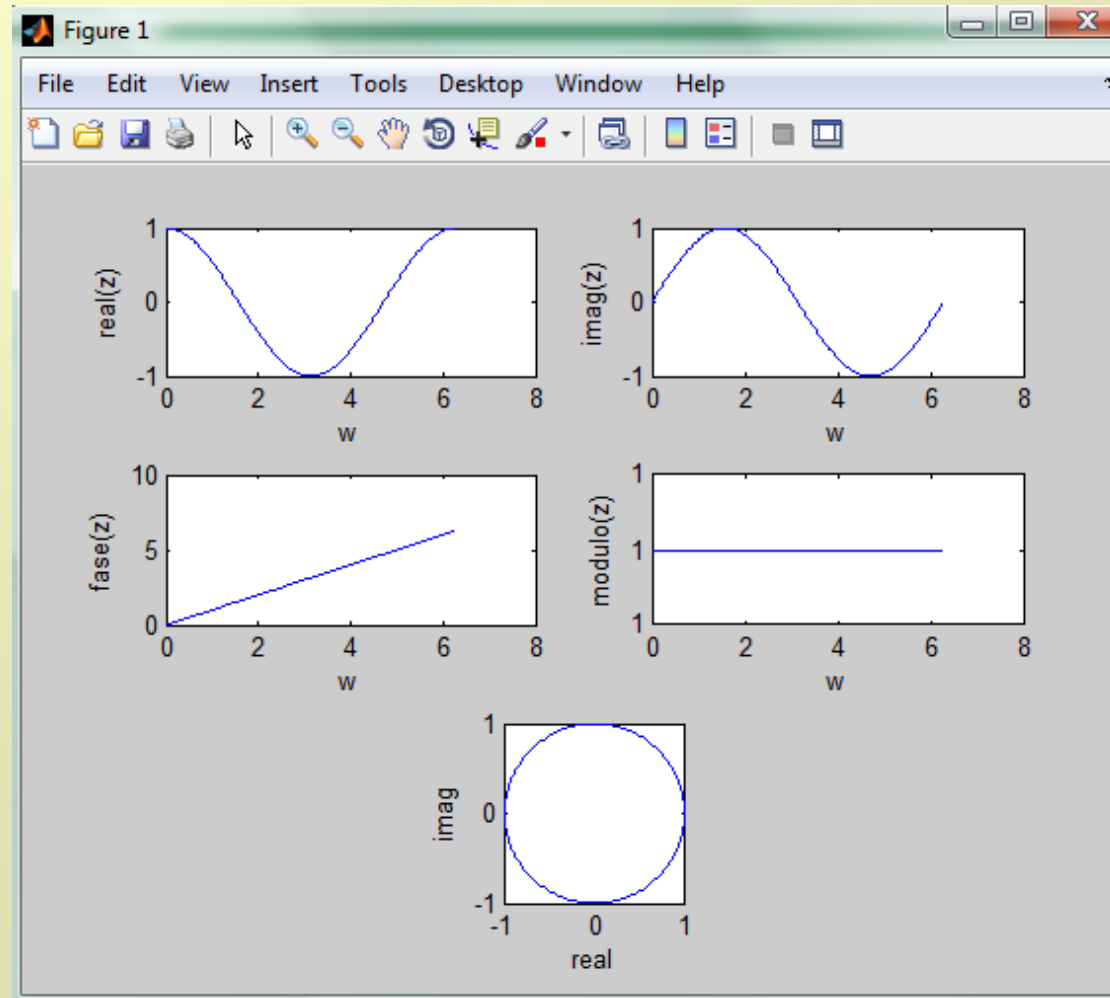
- Considere as seguintes funções complexas:

$$f(w) = \sin(4w)e^{iw} \quad \text{e} \quad g(w) = \sin(8w)e^{iw}$$

- Calcule o valor de $f(\omega)$ e $g(\omega)$ para $\omega \in [0, 2\pi]$ com 200 pontos linearmente espaçados. Represente, em dois sub-gráficos dispostos verticalmente, os gráficos da parte imaginária em função da parte real das duas funções, sendo a primeira representada a **verde** na parte superior e a segunda a **vermelho** na parte inferior. Coloque no gráfico as legendas e etiquetas necessárias à sua correta interpretação.

Gráficos de variáveis complexas

•Exemplo-Gráfico





Gráficos 3D de Superfícies

- Problema:

Como representar funções matemáticas da forma

$$z = f(x, y)$$

em que x e y pertencem a um dado intervalo.

- Este tipo de função pode ser representado por uma superfície num espaço tridimensional.



Gráficos 3D de Superfícies

- Funções do Matlab que desenhavam superfícies
 - **mesh** (**X**, **Y**, **Z**)
 - **surf** (**X**, **Y**, **Z**)
- As variáveis X , Y e Z são matrizes em que cada elemento representa o valor da respectiva coordenada nesse ponto.
- A função **meshgrid** auxilia na criação das matrizes X e Y :

[XX, YY]=meshgrid(x, y)

em que x e y são vectores com a grelha em x e y .

Gráficos 3D de Superfícies

- Exemplo

```
>> x=1:3
```

```
>> y=1:4
```

```
>> [xx, yy]=meshgrid(x, y)
```

xx =

1	2	3
---	---	---

1	2	3
---	---	---

1	2	3
---	---	---

1	2	3
---	---	---

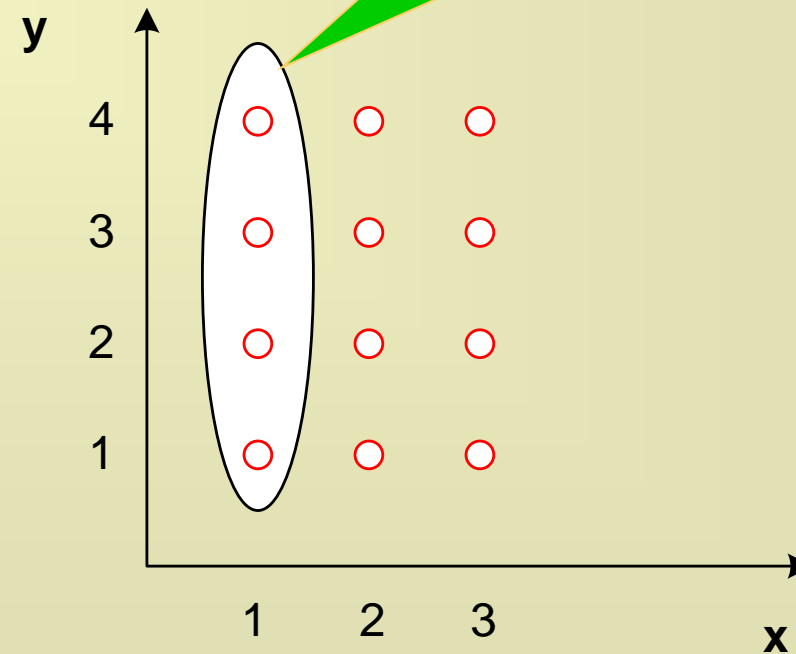
yy =

1	1	1
---	---	---

2	2	2
---	---	---

3	3	3
---	---	---

4	4	4
---	---	---





Gráficos 3D de Superfícies

- Considere-se a seguinte função matemática

$$f(x, y) = 2e^{-((x+1)^2 + (y+1)^2)} + e^{-5((x-1)^2 + (y-1)^2)}$$

em que x e y são representados por 51 pontos no seguinte intervalo

$$x \in [-3, 3] \wedge y \in [-3, 3]$$



Gráficos 3D de Superfícies

- Previamente temos discretizar o domínio da função:
 - define-se uma grelha (malha) rectangular de pontos

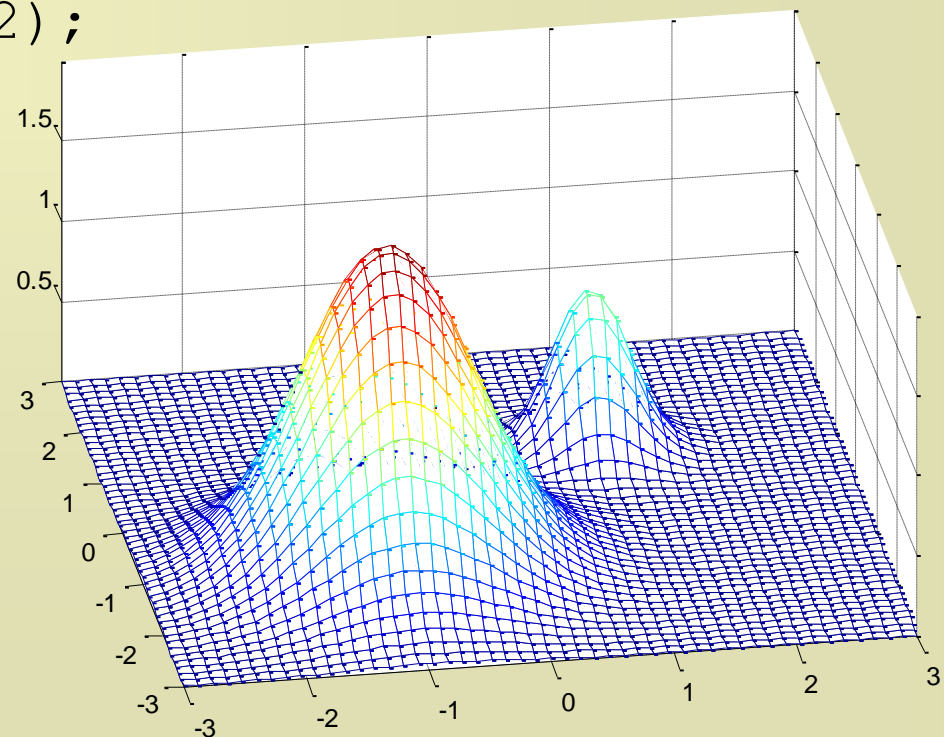
```
>> x = linspace(-3,3,51);  
>> y = linspace(-3,3,51);  
>> [XX,YY] = meshgrid(x,y);  
% XX matriz com todas as coordenadas x da grelha  
% YY matriz com todas as coordenadas y da grelha  
% Impressão da grelha  
>> plot(XX,YY(:,1),'k')  
>> hold on  
>> plot(XX(1,:),YY','k')
```

Gráficos 3D de Superfícies

- **Mesh**

```
>> %cálculo da função  
>> expo1 = -((XX+1).^2 + (YY+1).^2);  
>> expo2 = -5*((XX-1).^2 + (YY-1).^2);  
>> f = 2*exp(expo1) + exp(expo2);  
>> mesh(x,y,f), axis tight
```

Podem-se passar apenas os vectores x e y em vez das matrizes



Gráficos 3D de Superfícies

- **Surf**, análogo ao mesh mas ladrilhos são preenchidos com uma cor

```
>> surf(x,y,f), axis tight, %fig 1
```

```
>> surf(x,y,f), axis tight, shading interp %fig 2
```

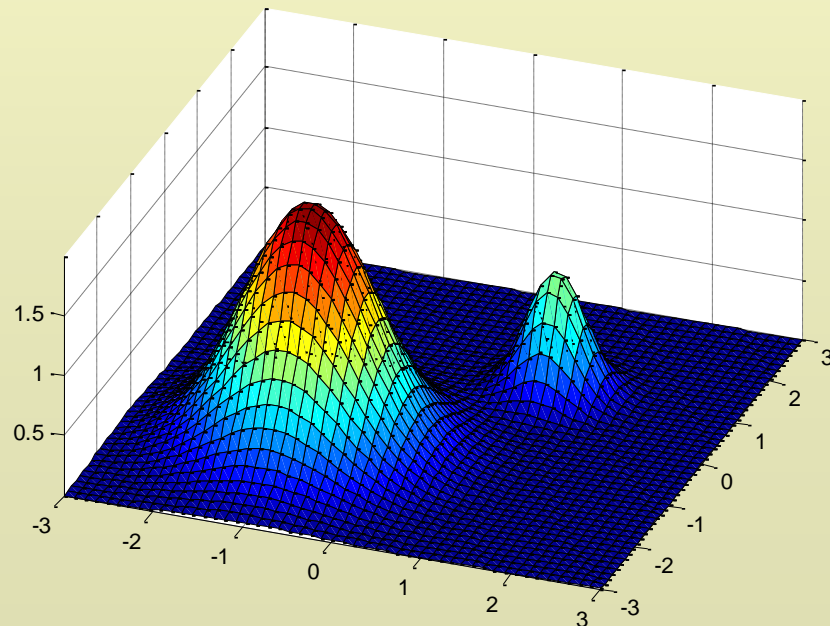


fig.1

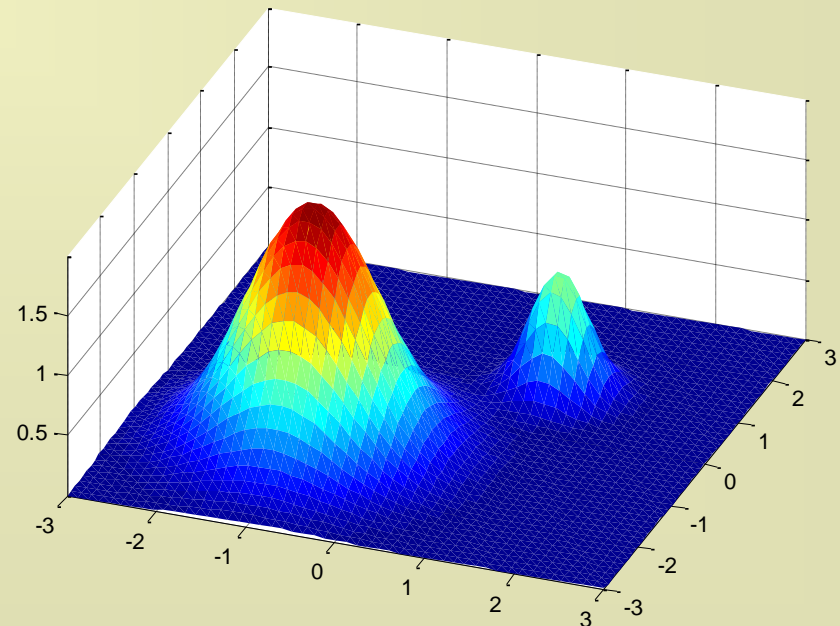


fig.2

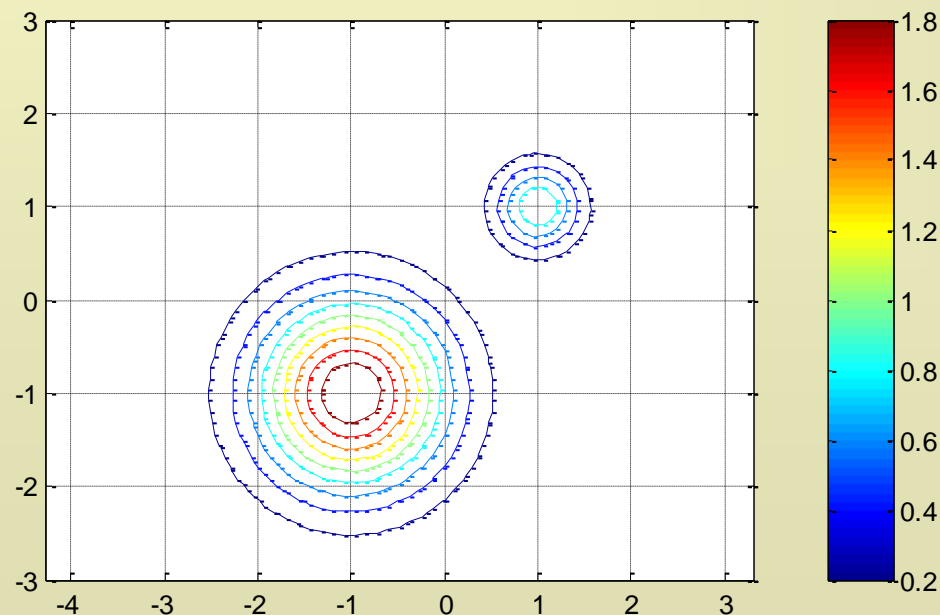
Curvas de nível

- Curvas que unem pontos de igual valor (isolinhas)

contour(x,y,z)

```
>> contour(x,y,f), grid on, colorbar
```

% importante informação de cor

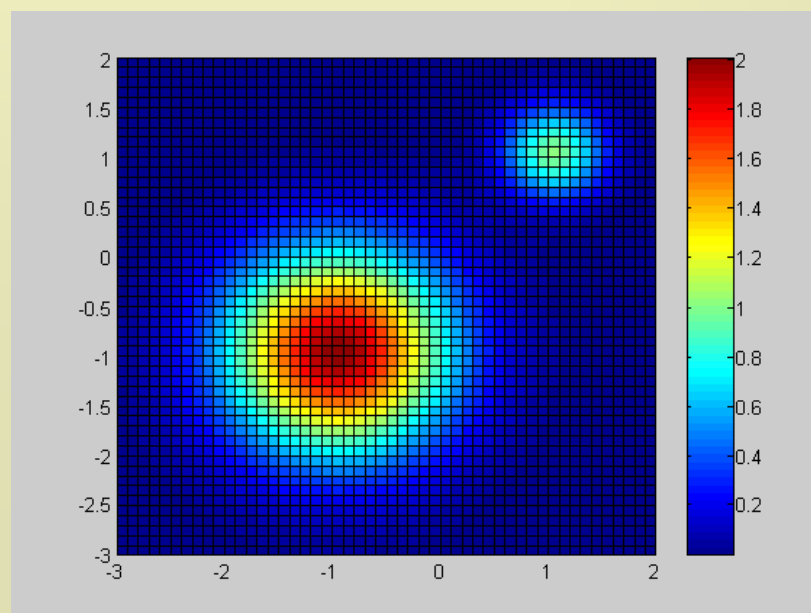


Gráficos de pseudo cor

- Neste tipo de gráfico cada cor representa uma dada amplitude segundo o eixo dos z . Cores iguais representam valores iguais em z .

pcolor(x,y,z)

```
>> pcolor(x,y,f), axis equal, grid on, colorbar
```





Sombra, Luz e Reflexão nas Superfícies

- Tipos de sombra
 - Depois de desenhar uma superfície 3D é possível modificar a forma como a superfície é colorida utilizando a função

Shading

com uma das seguintes opções:

- **shading flat** (cor única em cada quadrilátero)
- **shading faceted** (acrescenta contorno)
- **shading interp** (faz a interpolação das cores)



Exercício

- Considere a seguinte função $f(x,y)$ definida no domínio $x \in [-1,1] \wedge y \in [-1,1]$:

$$f(x, y) = \cos(4\pi(x + y))e^{-|x+y|}$$

- Calcule a função numa grelha de 51×51 pontos e elabore um gráfico de superfície utilizando um sombreado interpolado. Acrescente as legendas necessárias para aumentar a legibilidade do gráfico.



Programação no Matlab

- Sintaxe da Instrução **for**
 - A instrução **for** permite repetir um conjunto de instruções utilizando uma variável como contador de controlo.

Sintaxe

```
for n= ini:passo:fim  
    instrução1;  
    instrução2;  
    :  
end
```



Programação no Matlab

- Sintaxe da Instrução **while**

while condição

Instrução 1

Instrução 2

:

end

Enquanto a condição lógica for verdadeira o conjunto de instruções é executado. Note-se que se a condição for falsa no início as instruções nunca são executadas.



Programação no Matlab

- A instrução **if**

```
if condição1
    Instruções
elseif condição2
    Instruções
else
    Instruções
end
```

- A instrução **elseif** pode ocorrer mais do que uma vez.



Programação no Matlab

- Exemplo

Calcular a seguinte soma

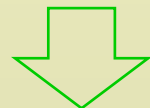
$$S = \sum_{n=1}^N \frac{1}{n^2}$$

Com o ciclo **for** calculam-se os primeiros N termos do somatório

```
S= 0 ;  
for n= 1:N  
    S= S + 1/n^2;  
end
```

Funções em Matlab

- Corresponde ao conceito de programa ou subprograma com **entradas/saídas** definidas formalmente
- Uma função aceita argumentos de entrada e devolve argumentos na saída
- Uma função manipula objectos (variáveis) cujo domínio de existência se restringe a um **“workspace” privado** criado no momento da execução da função.



- Este “workspace” é espaço de memória logicamente separado do “workspace” genérico criado para executar comandos nativos do Matlab



Funções em Matlab

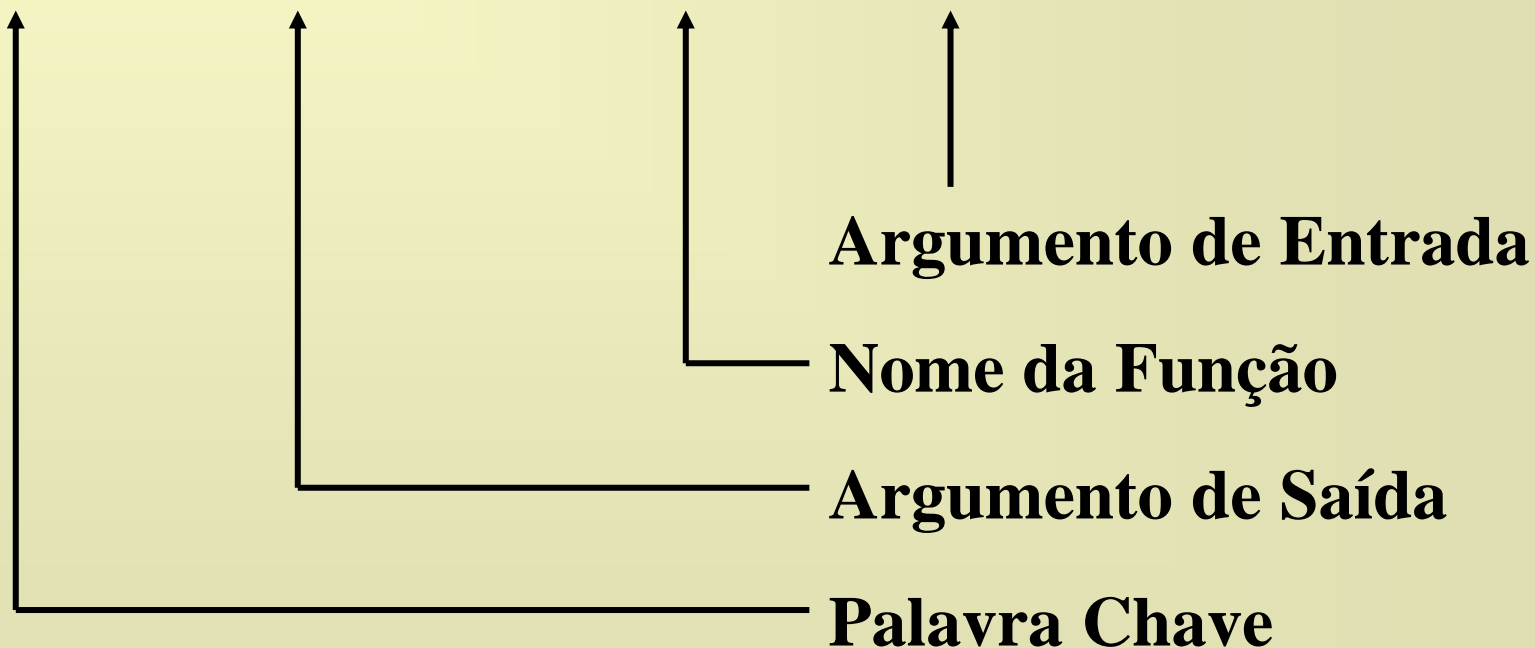
- Um função em Matlab é um ficheiro “.m”
- Um ficheiro “.m” onde se pretende definir uma função deve obedecer à seguinte organização mínima
 - Linha de definição
 - 1ª linha informativa
 - Texto de help
 - Corpo da função
 - Comentários



Definição de funções

- Linha de definição (caso mais simples)

function f = fibo (n)

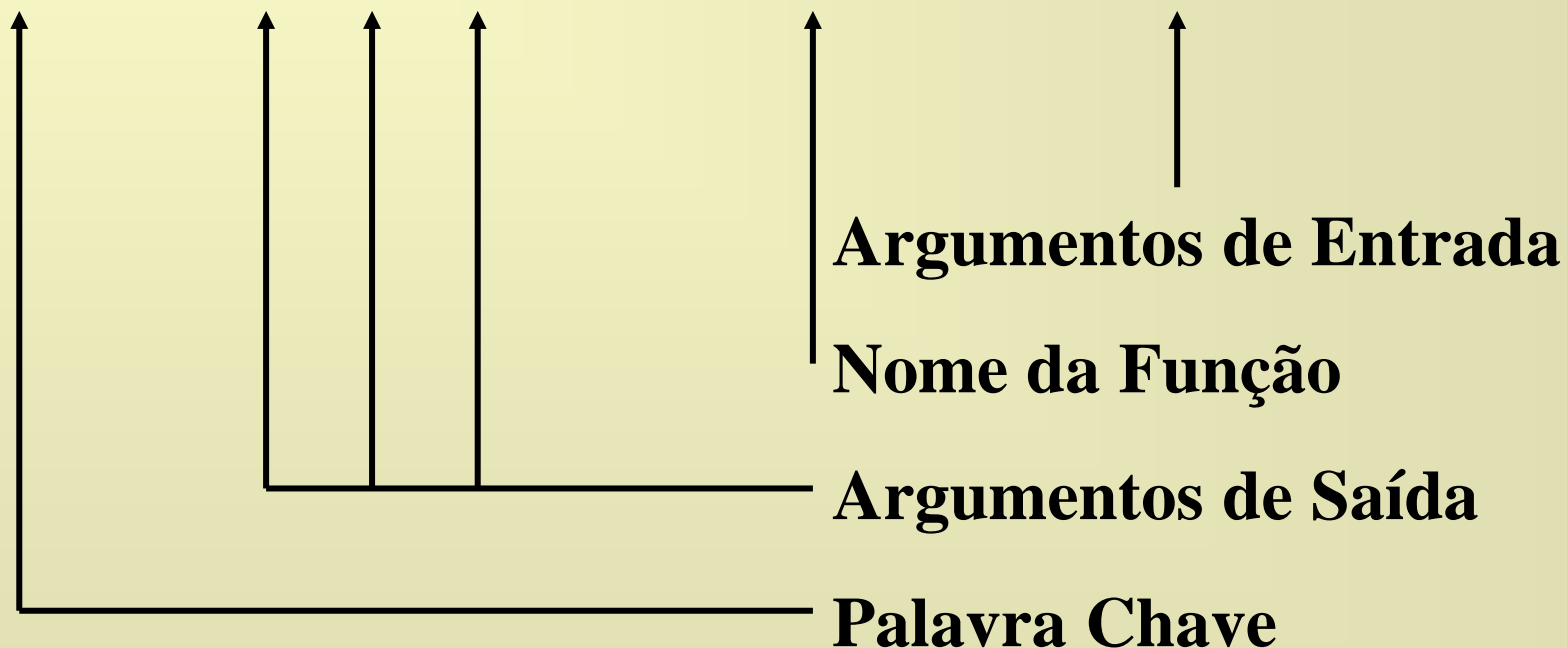




Definição de funções

- Linha de definição (caso geral)

function **[y w z]** = qqcoisa(x,u,v)

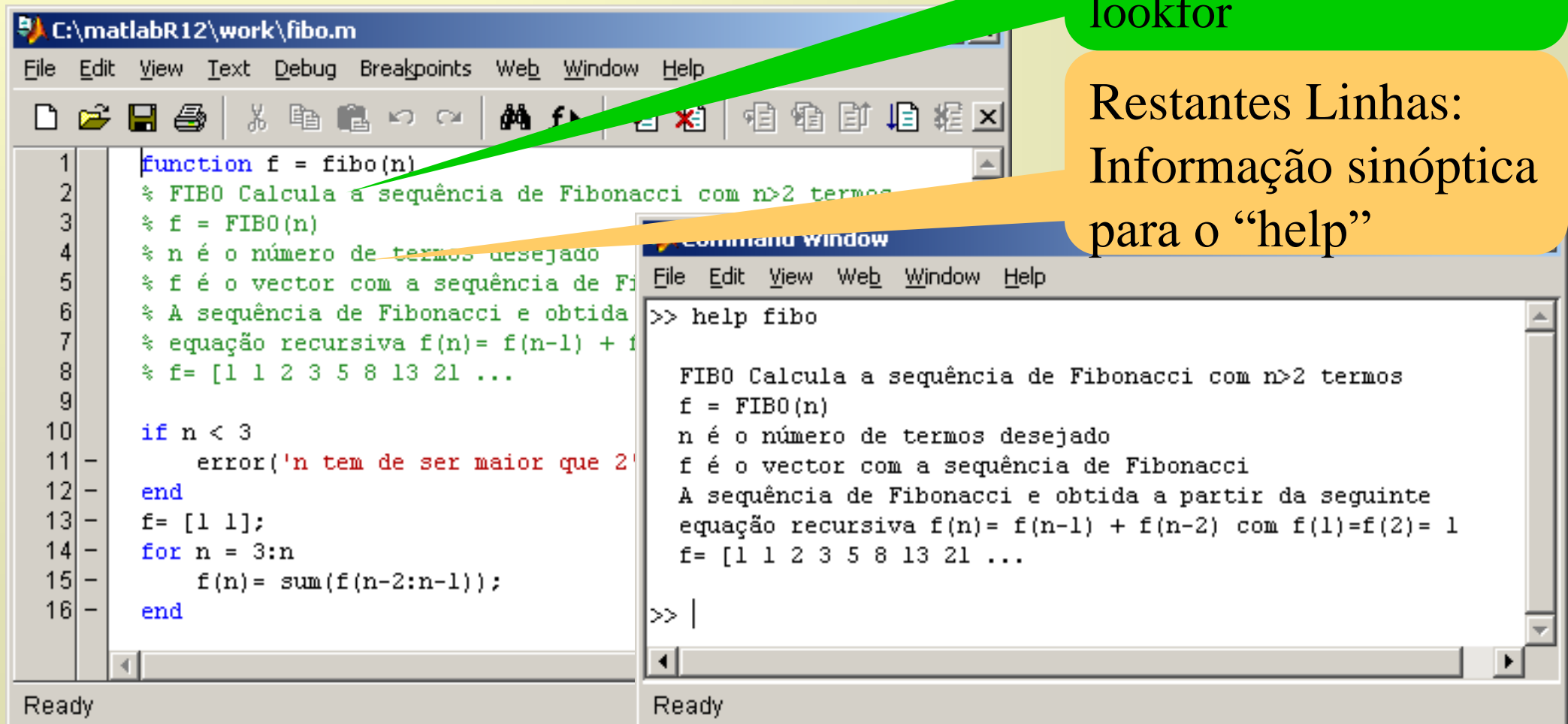


Definição de funções

• Documentação

1ª Linha: Informação sumária. Utilizada pelo lookfor

Restantes Linhas: Informação sinóptica para o “help”



```
C:\matlabR12\work\fibom
File Edit View Text Debug Breakpoints Web Window Help
[Icons]
1 function f = fibo(n)
2 % FIBO Calcula a sequência de Fibonacci com n>2 termos
3 % f = FIBO(n)
4 % n é o número de termos desejado
5 % f é o vector com a sequência de Fibonacci
6 % A sequência de Fibonacci é obtida a partir da seguinte
7 % equação recursiva f(n)= f(n-1) + f(n-2) com f(1)=f(2)= 1
8 % f= [1 1 2 3 5 8 13 21 ...]
9
10 if n < 3
11     error('n tem de ser maior que 2')
12 end
13 f= [1 1];
14 for n = 3:n
15     f(n)= sum(f(n-2:n-1));
16 end

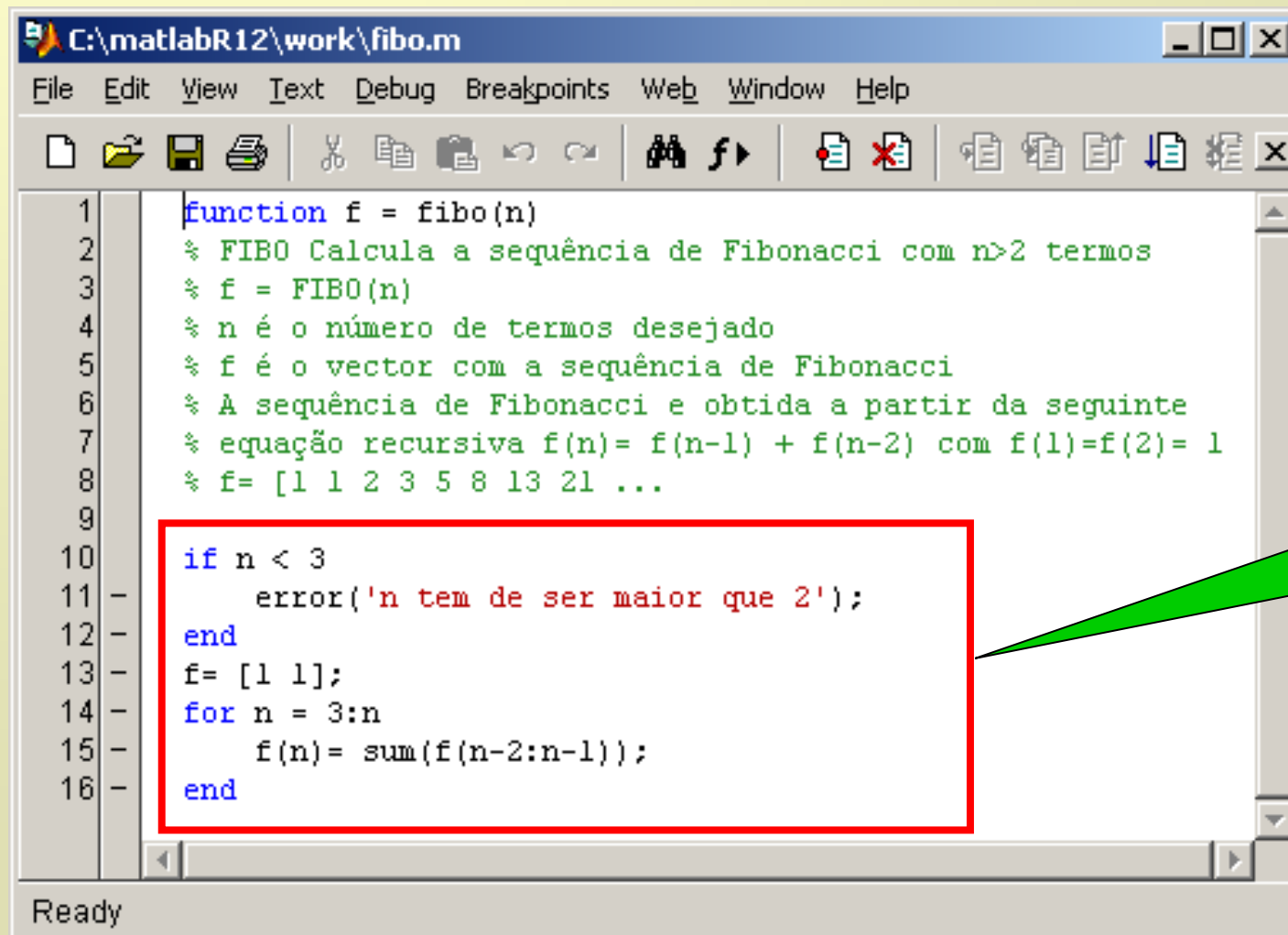
Command Window
File Edit View Web Window Help
>> help fibo

FIBO Calcula a sequência de Fibonacci com n>2 termos
f = FIBO(n)
n é o número de termos desejado
f é o vector com a sequência de Fibonacci
A sequência de Fibonacci é obtida a partir da seguinte
equação recursiva f(n)= f(n-1) + f(n-2) com f(1)=f(2)= 1
f= [1 1 2 3 5 8 13 21 ...]

>> |
```

Definição de funções

- Corpo da função

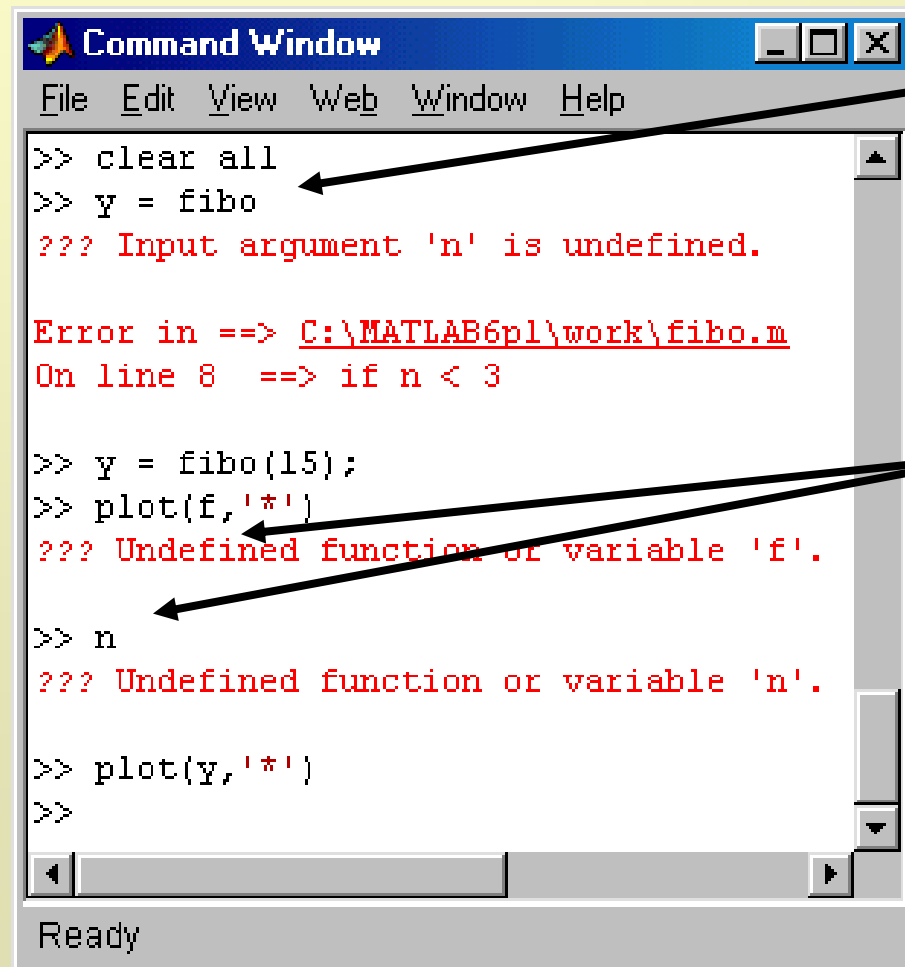


```
1 function f = fibo(n)
2 % FIBO Calcula a sequência de Fibonacci com n>2 termos
3 % f = FIBO(n)
4 % n é o número de termos desejado
5 % f é o vector com a sequência de Fibonacci
6 % A sequência de Fibonacci é obtida a partir da seguinte
7 % equação recursiva f(n)= f(n-1) + f(n-2) com f(1)=f(2)= 1
8 % f= [1 1 2 3 5 8 13 21 ...]
9
10 if n < 3
11     error('n tem de ser maior que 2');
12 end
13 f= [1 1];
14 for n = 3:n
15     f(n)= sum(f(n-2:n-1));
16 end
```

Corpo da
função

Definição de funções

• Erros frequentes



```
Command Window
File Edit View Web Window Help

>> clear all
>> y = fibo
??? Input argument 'n' is undefined.

Error in ==> C:\MATLAB6p1\work\fibonacci.m
On line 8 ==> if n < 3

>> y = fibo(15);
>> plot(f, '*')
??? Undefined function or variable 'f'.

>> n
??? Undefined function or variable 'n'.

>> plot(y, '*')
>>
```

**Invocação
deficiente**

**Objectos privados da função.
Não existem no “workspace”
genérico**



Definição de funções

- Regras para atribuição de nomes de funções
 - Os nomes de funções seguem as mesmas regras de nomeação de variáveis.
 - Aceitam-se no máximo 31 caracteres incluindo “_”. O primeiro caracter tem de ser uma letra.
 - O nome do ficheiro “.m” que contém a função **deverá ser gravado** como “**nome_da_função.m**”
Exemplo: `function y = aveg(x) ... => ficheiro aveg.m`
 - Caso assim não seja o nome interno é ignorado.
 - Esta prática é **fortemente desaconselhada**

**Nomes
iguais**



Exercício

- Considere a expansão em série de Taylor da função seno dada pelo somatório

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots = \sum_{n=1}^N \frac{x^n}{n!} \sin\left(\frac{n\pi}{2}\right)$$

- Elabore uma função em Matlab que calcule o valor deste somatório. Resolva com um ciclo for. Os parâmetros de entrada são o valor de **N** e o vector **x** e o de saída o valor da série.
- Teste a função para $x \in [0, 2\pi]$ e para **N**=2 e **N**=10