



Sistemas Multimédia

Breve Introdução ao MatLab

Universidade de Aveiro



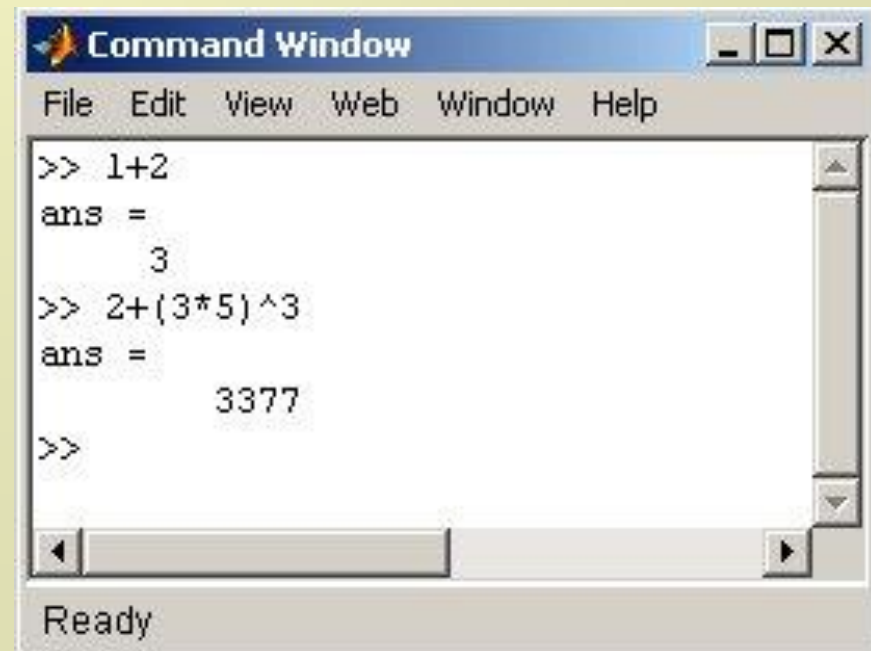
Sumário

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



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:

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

1 2 3 4

Índices das
colunas

Matrizes

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 0 & 7 & 6 & 12 \end{bmatrix}$$

Diagram illustrating the extraction of the submatrix $A_{2,3}$ from matrix A . The element 10 is circled in red, and a green arrow points from it to a green circle labeled $A_{2,3}$. The matrix is indexed with red numbers: rows 1, 2, 3 and columns 1, 2, 3, 4.

16	2	3	13
5	11	10	8
0	7	6	12

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
```



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:

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



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.



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”

- Cálculo 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}

$$\mathbf{x}(\mathbf{x} < 0)$$

$$-1 \ -3$$

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);
```

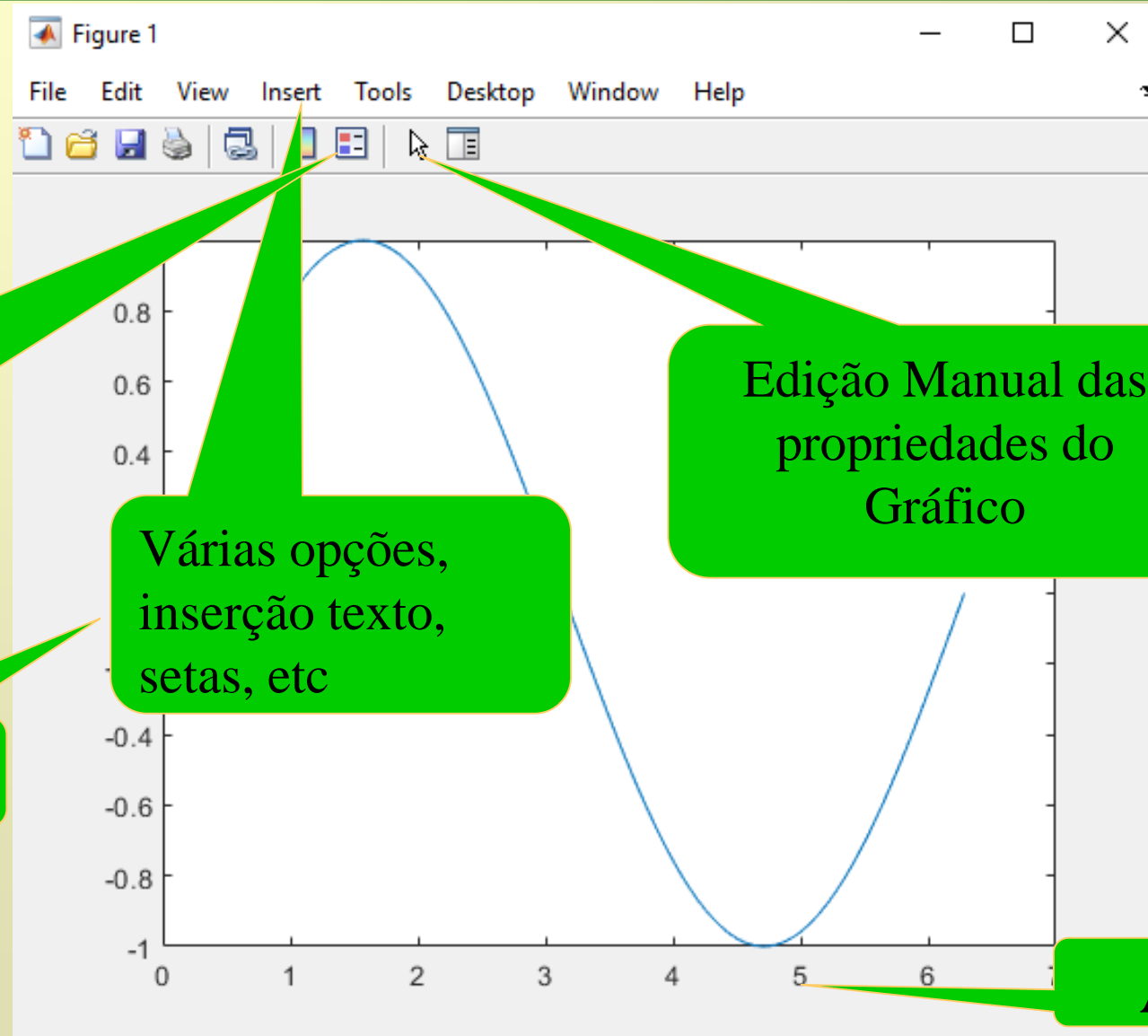
```
>>plot(x,z)
```

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
:	ponteada
-.	traço ponto
--	tracejada

Gráficos elementares

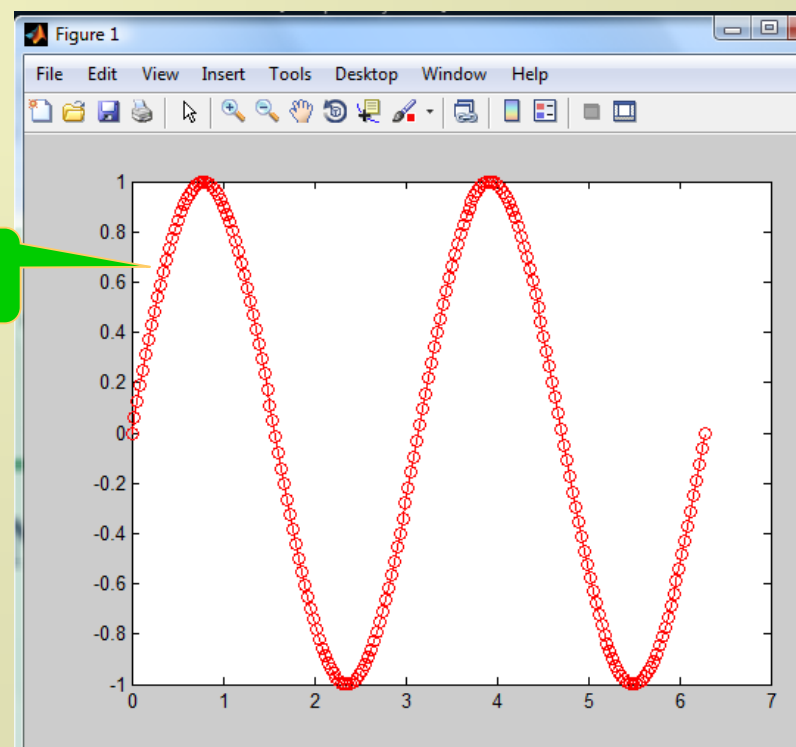
- Título, labels dos eixos, legenda
 - Na sequência do plot anterior:

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

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

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

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



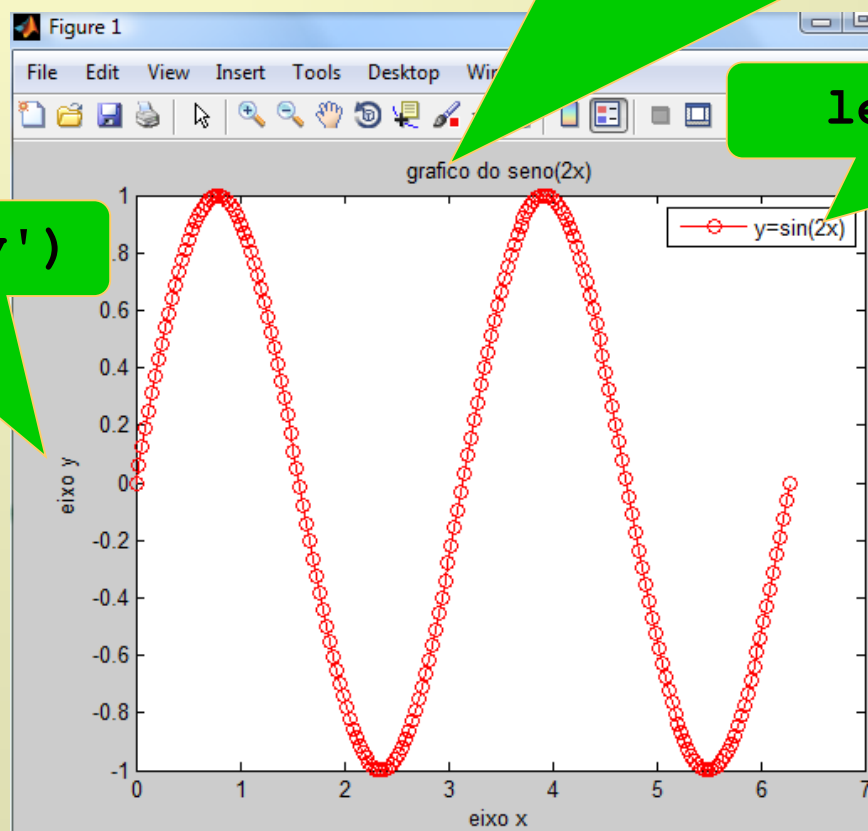
Gráficos elementares

- Título, labels dos eixos, legenda

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

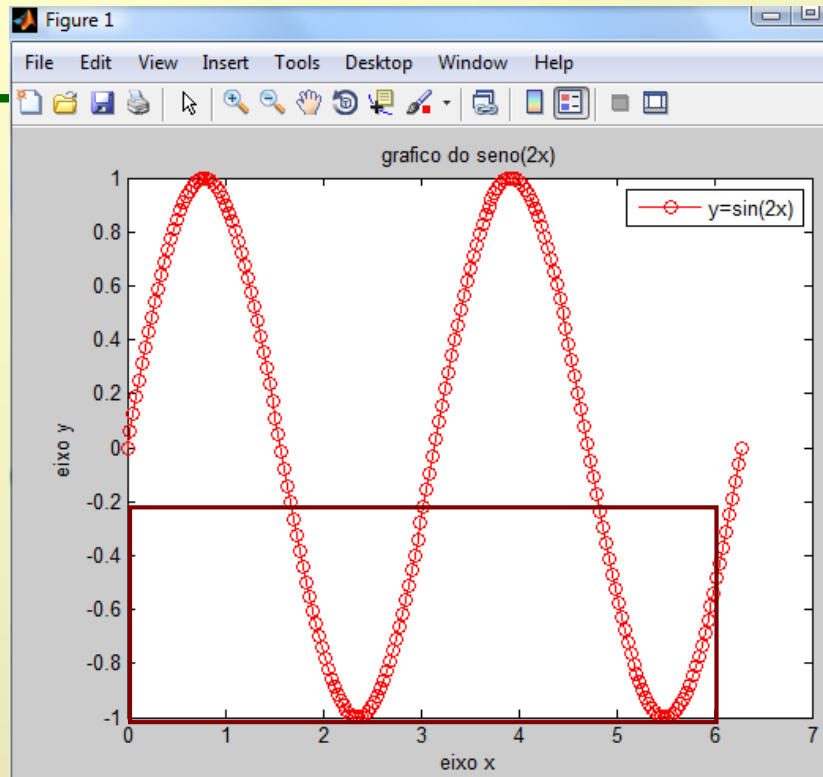
`legend('y=sin(2x)')`

`ylabel('eixo y')`

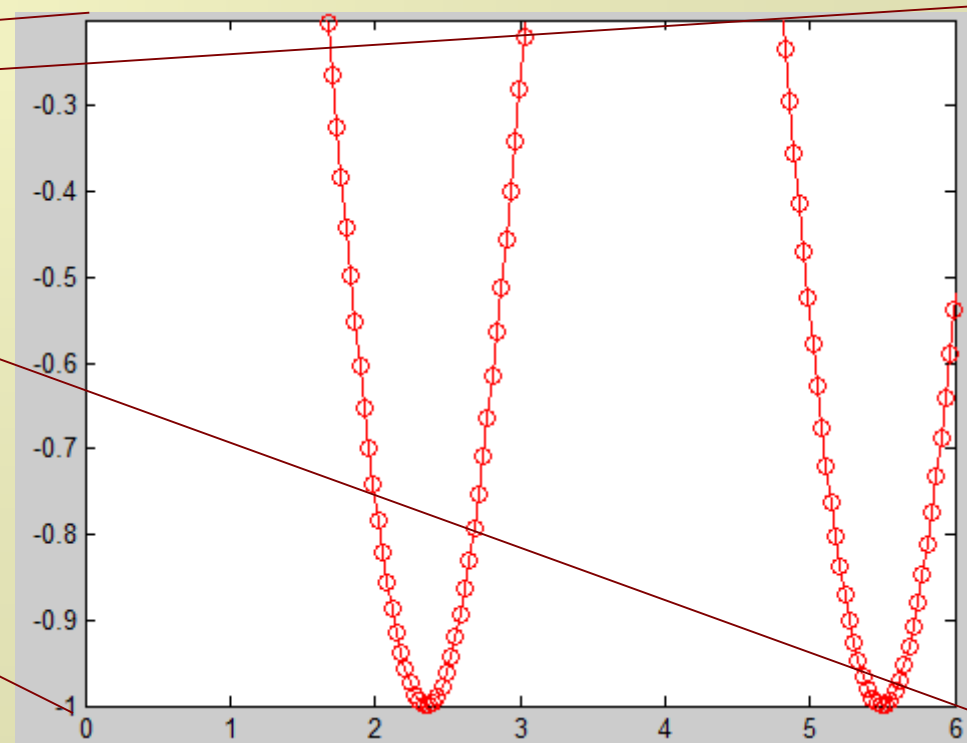


`xlabel('eixo x')`

Eixos – Função **axis**



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





Gráficos elementares

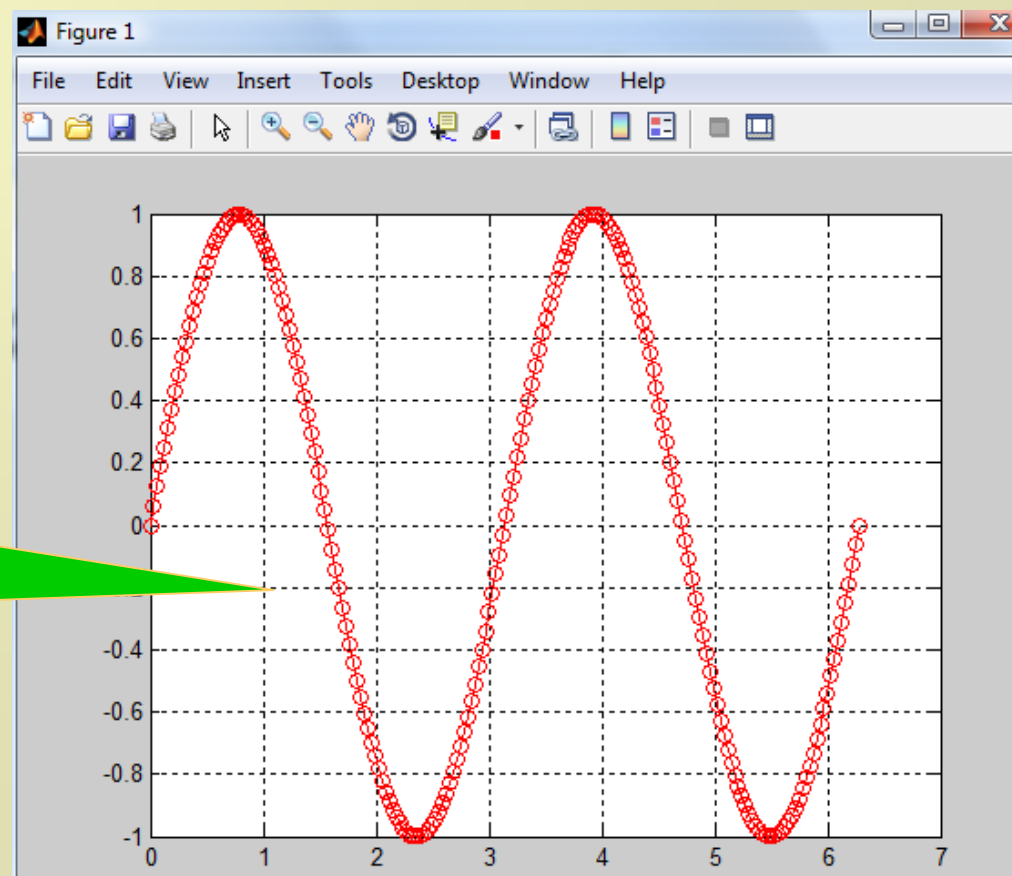
- Sintaxe da função **axis**

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

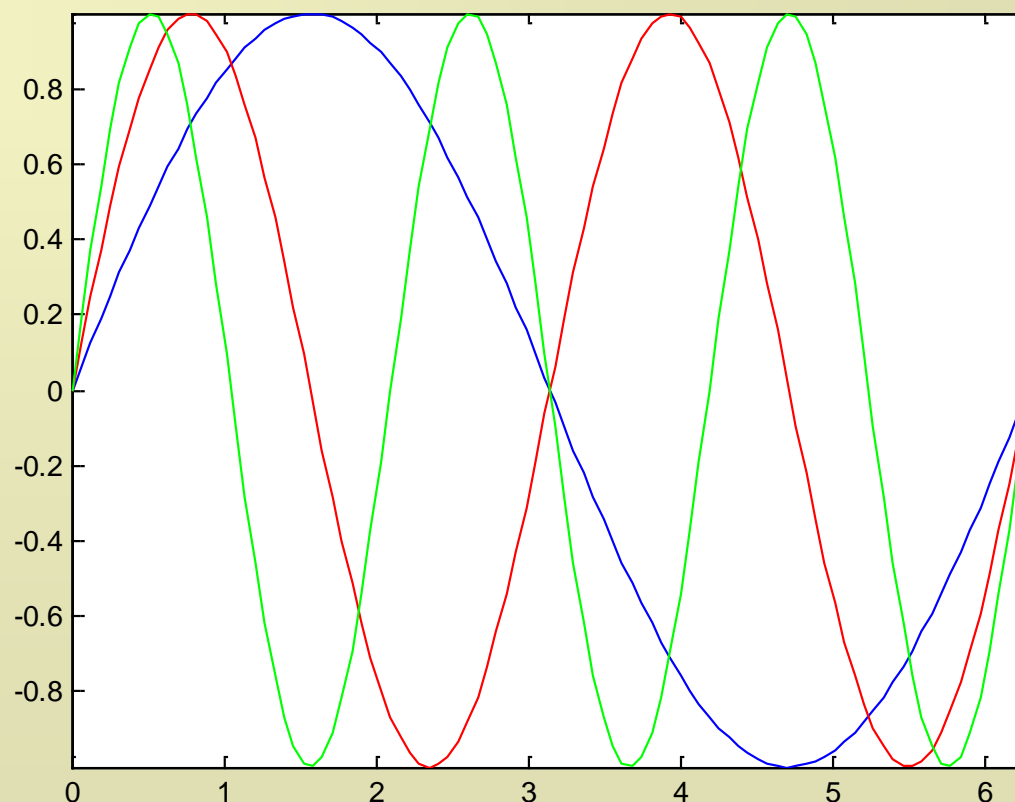
- Em Matlab existem várias formas de sobrepor curvas na mesma janela gráfica
 - Colocar na função **plot** todas as curvas que se pretende representar
plot(x1,y1,x2,y2,...)
plot(x1,y1,'string1',x2,y2,'string2',...)
 - O número de elementos dos pares **(x1,y1)** e **(x2,y2)** deve ser o mesmo.
 - No entanto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



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 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
---	---	---

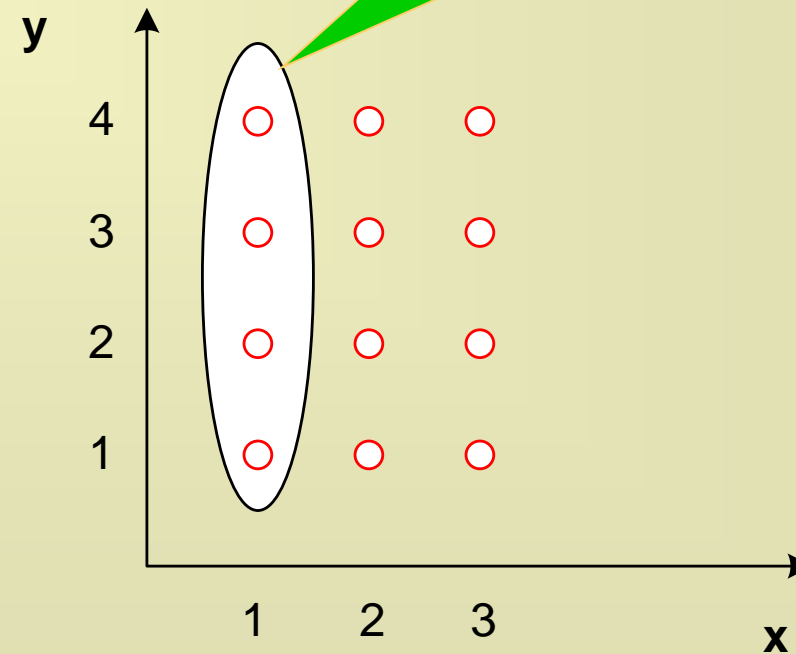
yy =

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')
```

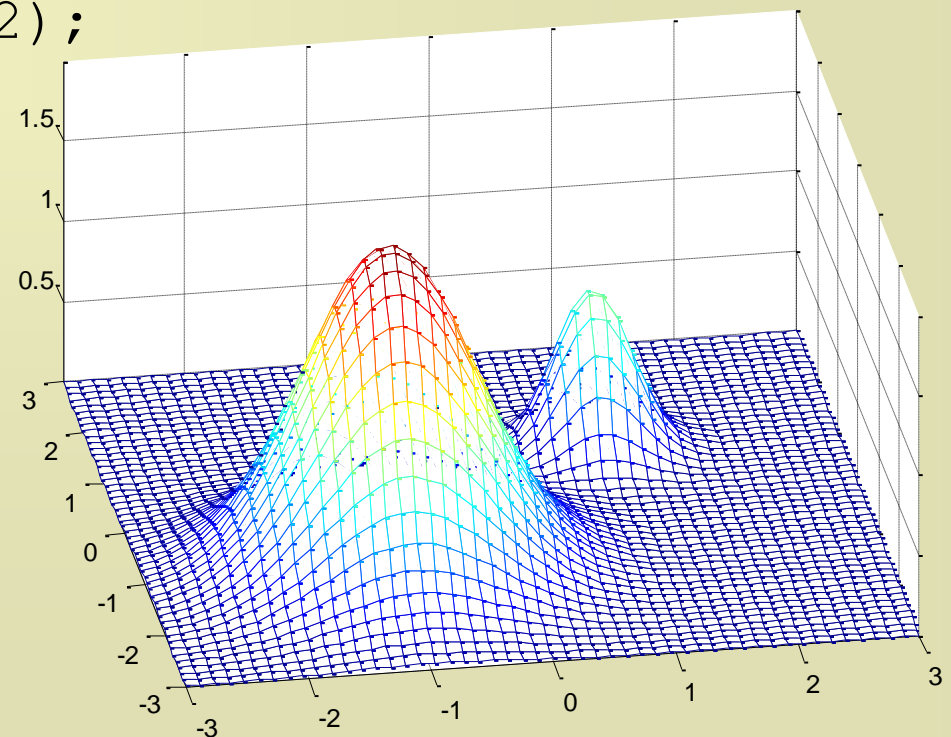



Gráficos 3D de Superfícies

- **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
```

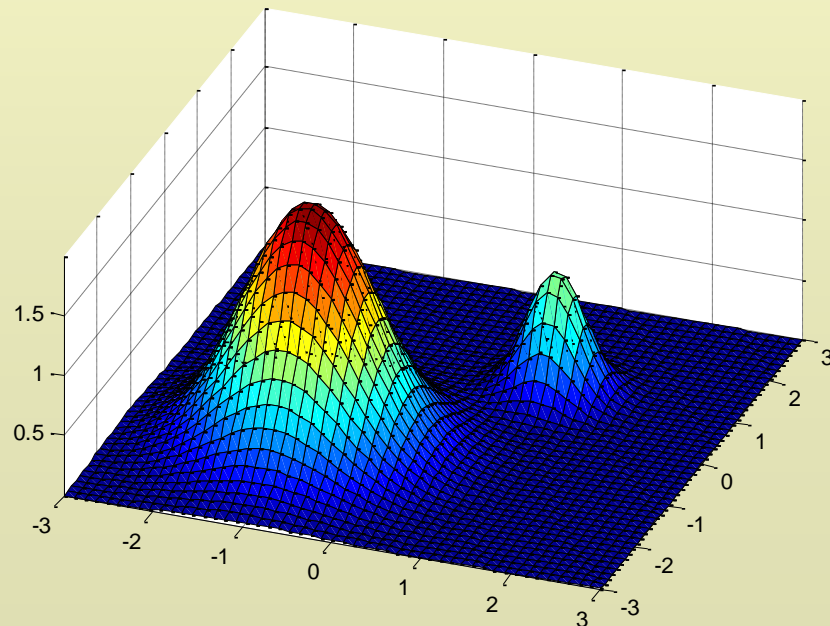


fig.1

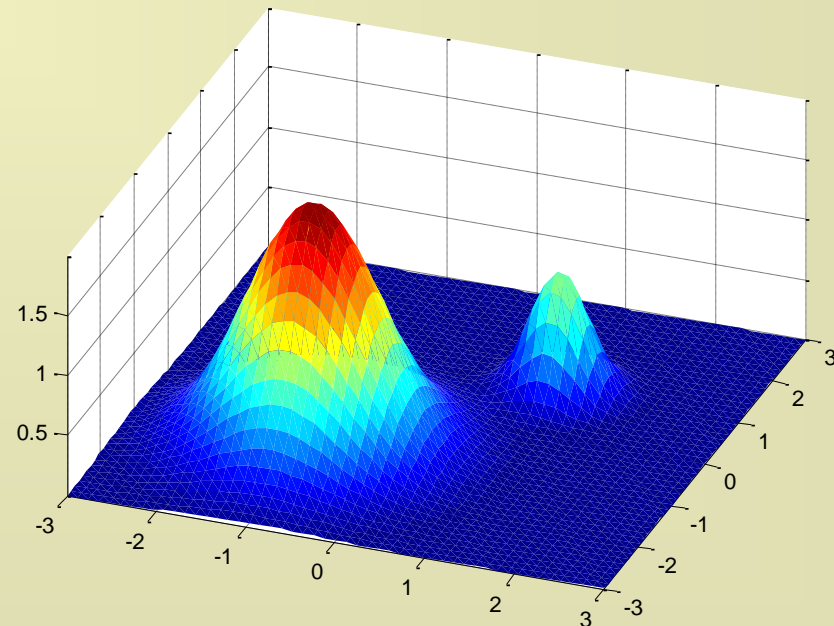


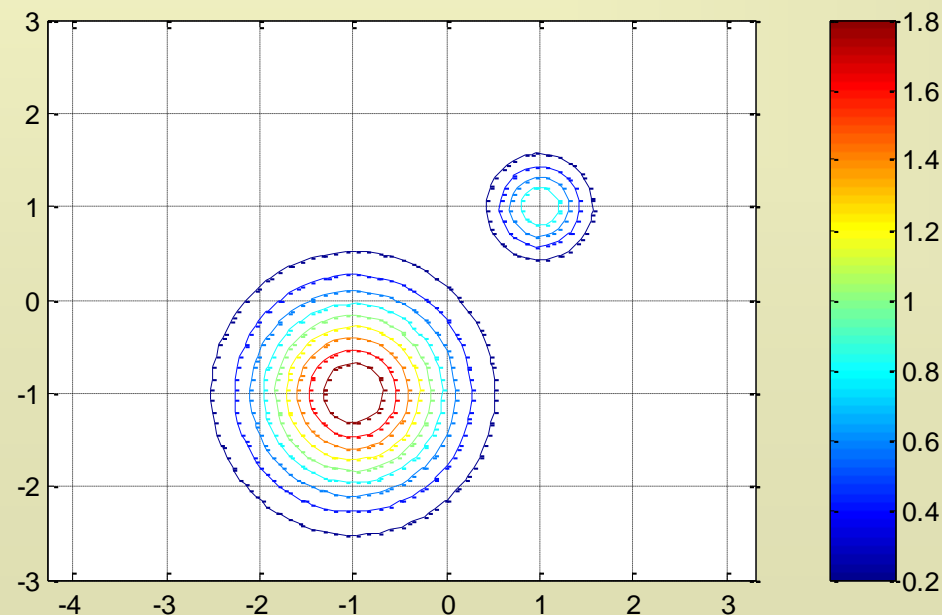
fig.2

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
```

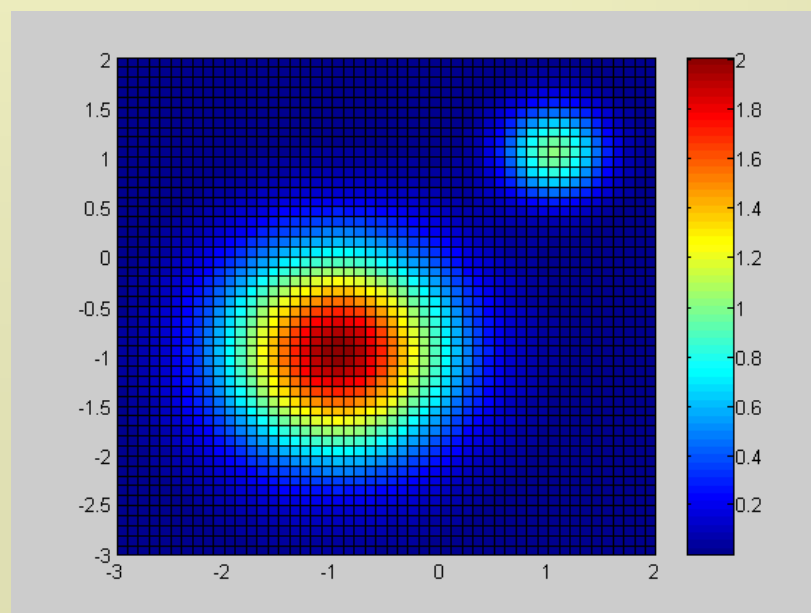


Gráficos de pseudo cor

- 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

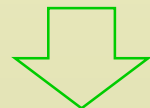
$$S = \sum_{n=1}^N \frac{1}{n^2}$$

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

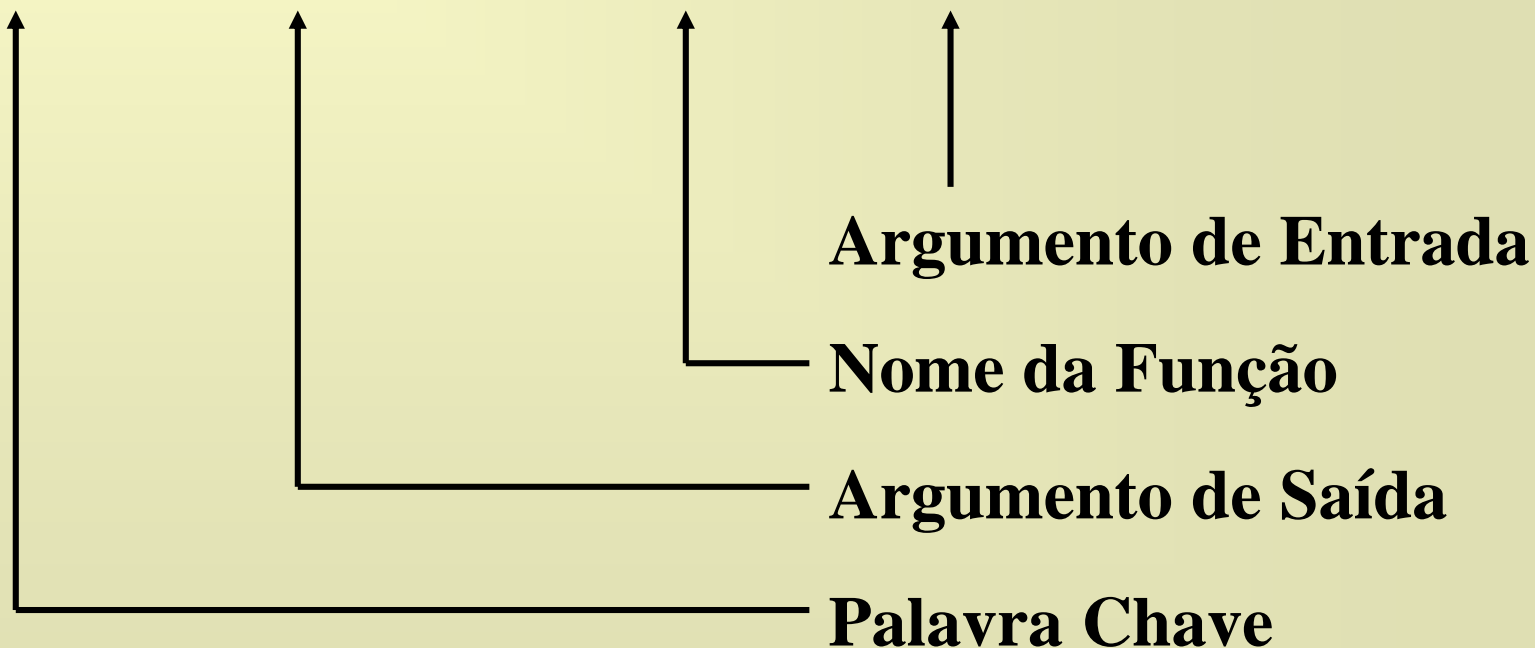
- Um função em Matlab é um ficheiro “.m”
- Um ficheiro “.m” onde se pretende definir uma função deve obedecer à seguinte organização mínima
 - Linha de definição
 - 1ª linha informativa
 - Texto de help
 - Corpo da função
 - Comentários



Definição de funções

- Linha de definição (caso mais simples)

function **f** = **fibonacci** (**n**)

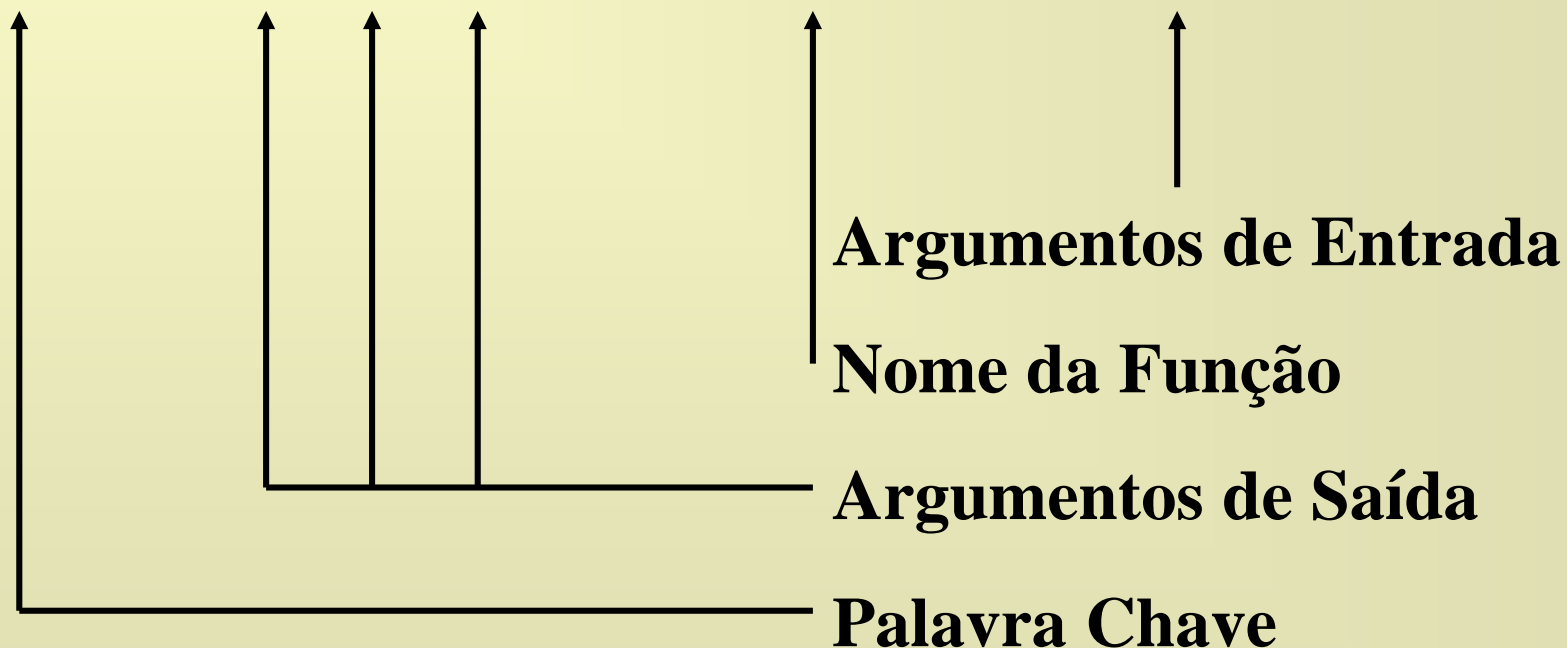




Definição de funções

- Linha de definição (caso geral)

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

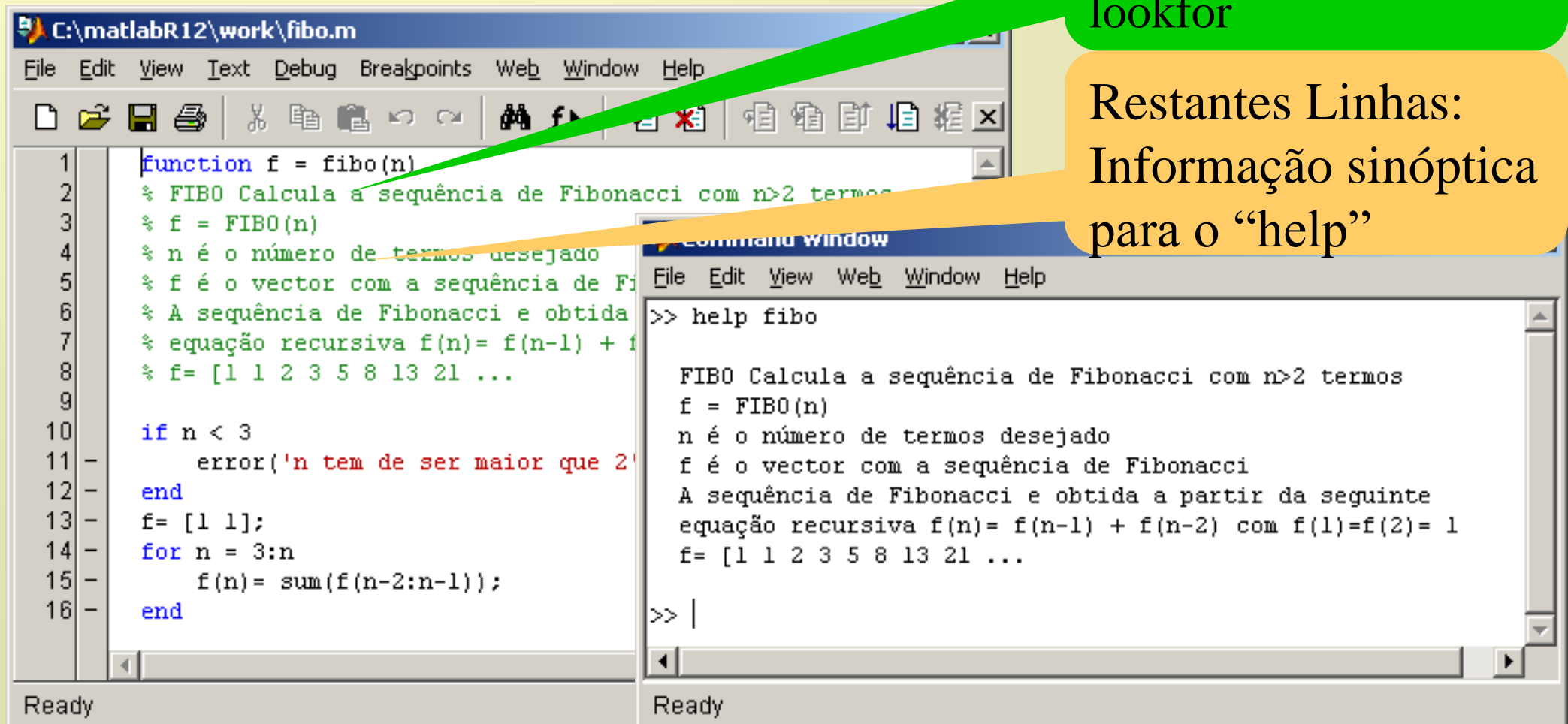


Definição de funções

• 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

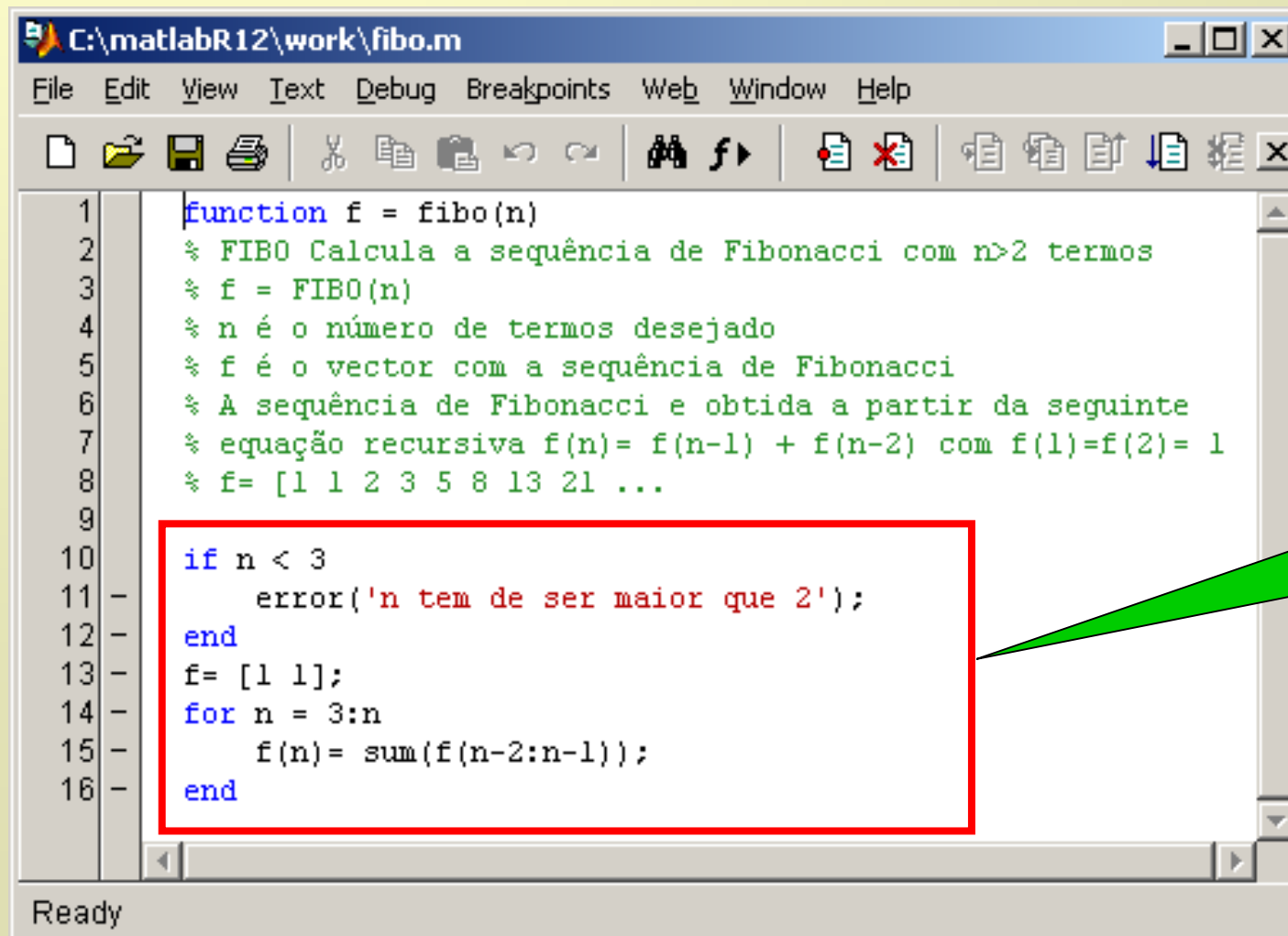
Command Window
File Edit View Web Window Help
>> help fibo

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

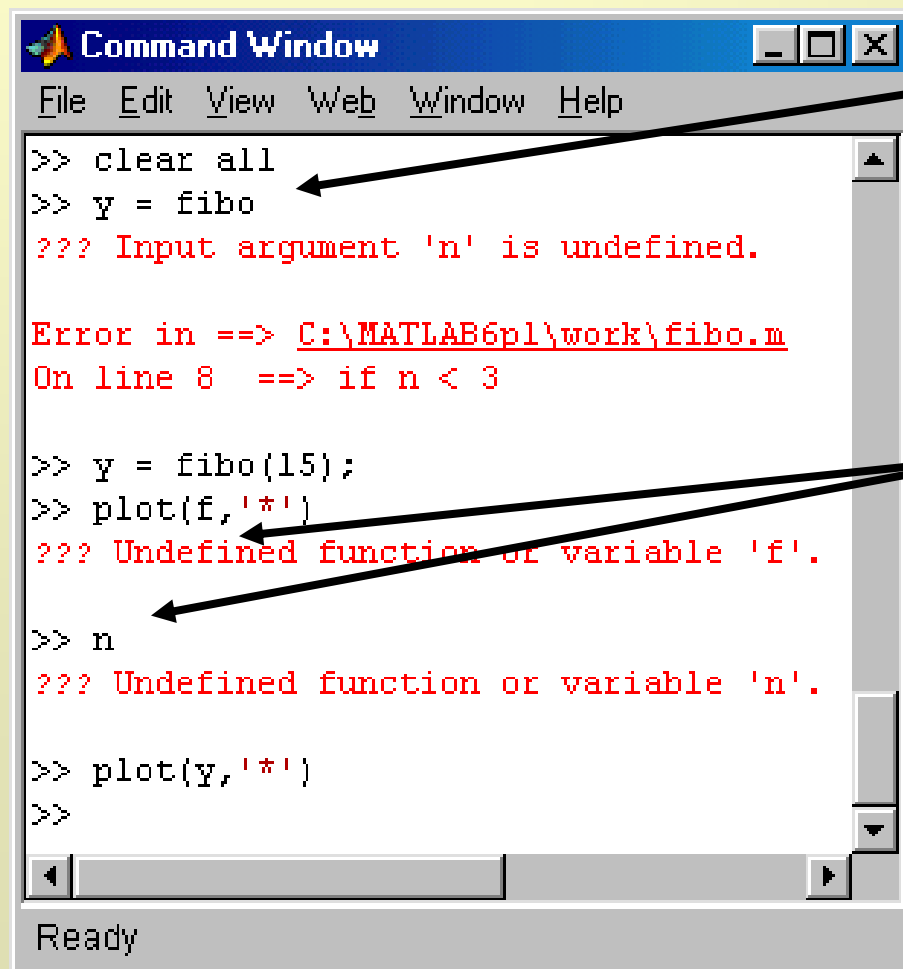


```
1 function f = fibo(n)
2 % FIBO Calcula a sequência de Fibonacci com n>2 termos
3 % f = FIBO(n)
4 % n é o número de termos desejado
5 % f é o vector com a sequência de Fibonacci
6 % A sequência de Fibonacci é obtida a partir da seguinte
7 % equação recursiva f(n)= f(n-1) + f(n-2) com f(1)=f(2)= 1
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:\MATLAB6pl\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

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

- 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