



# Matlab Workshop - IEEE



# O que é o Matlab ?

- Aplicação informática vocacionada para o cálculo numérico
- Aplicações
  - Análise de dados
  - Visualização científica
  - Simulação de sistemas



# 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 Matlab como calculadora

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

```
Command Window
>> 1+2
ans =
    3
>> 2+3*4
ans =
   14
>> 2^2
ans =
    4
fx >> |
```



# Variáveis

- Variáveis

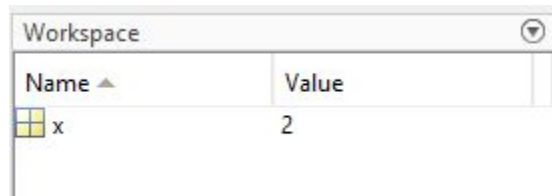
- No Matlab é possível guardar em variáveis conjuntos de números, exemplo:

```
>> x = 2
```

- Os nomes das variáveis distinguem as letras maiúsculas das minúsculas.

Exemplo:  $\pi \neq \text{Pi}$

- As variáveis são guardadas no espaço de trabalho “workspace”



| Workspace |       |
|-----------|-------|
| Name ▲    | Value |
| x         | 2     |

- As variáveis podem ser utilizadas nas operações da mesma forma que os números.



# Variáveis

- Apagar variáveis

- `clear v1 v2` apaga as variáveis `v1` e `v2`
- `clear all` apaga todas as variáveis

- Ver as variáveis no espaço de trabalho (“workspace”)

- `whos` mostra todas as variáveis do espaço de trabalho com informação adicional de dimensão e tipo

- Guardar variáveis

- `save` Guarda em disco todas as variáveis do “workspace”
- `load` Carrega do disco as variáveis guardadas
- `save ficheiro v1 v2` Guarda as variáveis `v1` e `v2` no ficheiro
- `load ficheiro` Carrega as variáveis do ficheiro



# Números complexos

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

$$1 + 2i$$

basta introduzir na janela de comandos:

»1+2i

ou

»1+2\*i

```
Command Window
>> 1 + 2i

ans =

    1.0000 + 2.0000i

fx >> |
```



# Números complexos

- Algumas funções matemáticas podem devolver números complexos para determinados valores do argumento. Exemplos:

$$\sqrt{-1} = i$$

$$\log(-1) = \pi i$$

```
Command Window
>> sqrt(-1)

ans =

    0.0000 + 1.0000i

>> log(-1)

ans =

    0.0000 + 3.1416i

fx >>
```





# Funções matemáticas

O Matlab dispõe dum vasto conjunto de funções matemáticas.

|      |                    |       |                                    |
|------|--------------------|-------|------------------------------------|
| 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           |



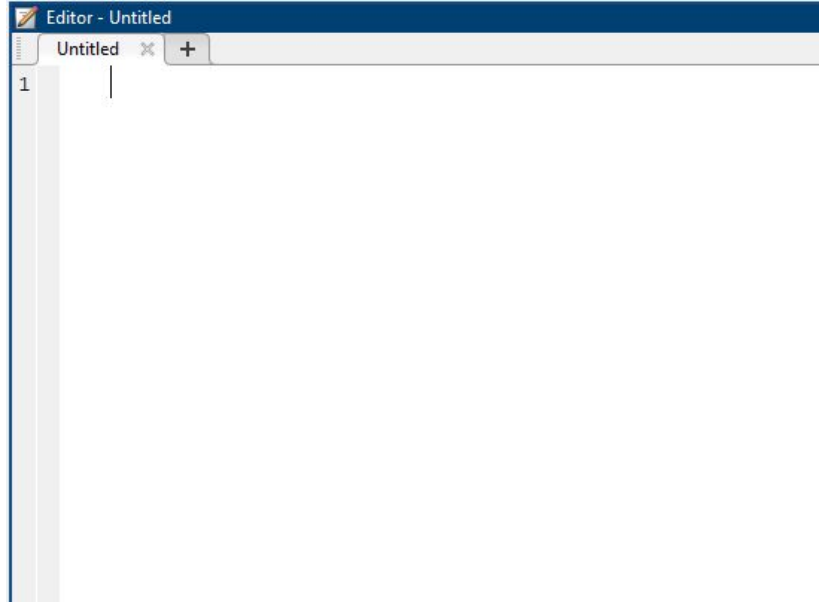
# Funções matemáticas

|         |   |
|---------|---|
| pi      | $\pi$   |
| i       | $\sqrt{-1}$                                     |
| j       | $\sqrt{-1}$                                     |
| eps     | Precisão relativa do formato “double” $2^{-52}$ |
| realmin | Menor número real $2^{-1022}$                   |
| realmax | Maior número real $(2-\text{eps})2^{1023}$      |
| Inf     | Infinito  |
| NaN     | “Not-a-Number”                                  |



# “Scripts” no Matlab

- Os “scripts” no Matlab são ficheiros de texto com instruções Matlab. Quando na janela de comandos do Matlab se escreve o nome do “script” as instruções nele contidas são executadas sequencialmente. Os “scripts” permitem assim automatizar um conjunto de procedimentos.





## Código por secções

```
1      % Relatorios no Matlab
2
3      %% Parte 1 - Declaracao de variaveis
4
5 -    a= 1;
6 -    b= 2;
7
8      %% Parte 2 - Processamento das variaveis
9      % $c= a+b$
10
11 -   c= a+b;
12
```

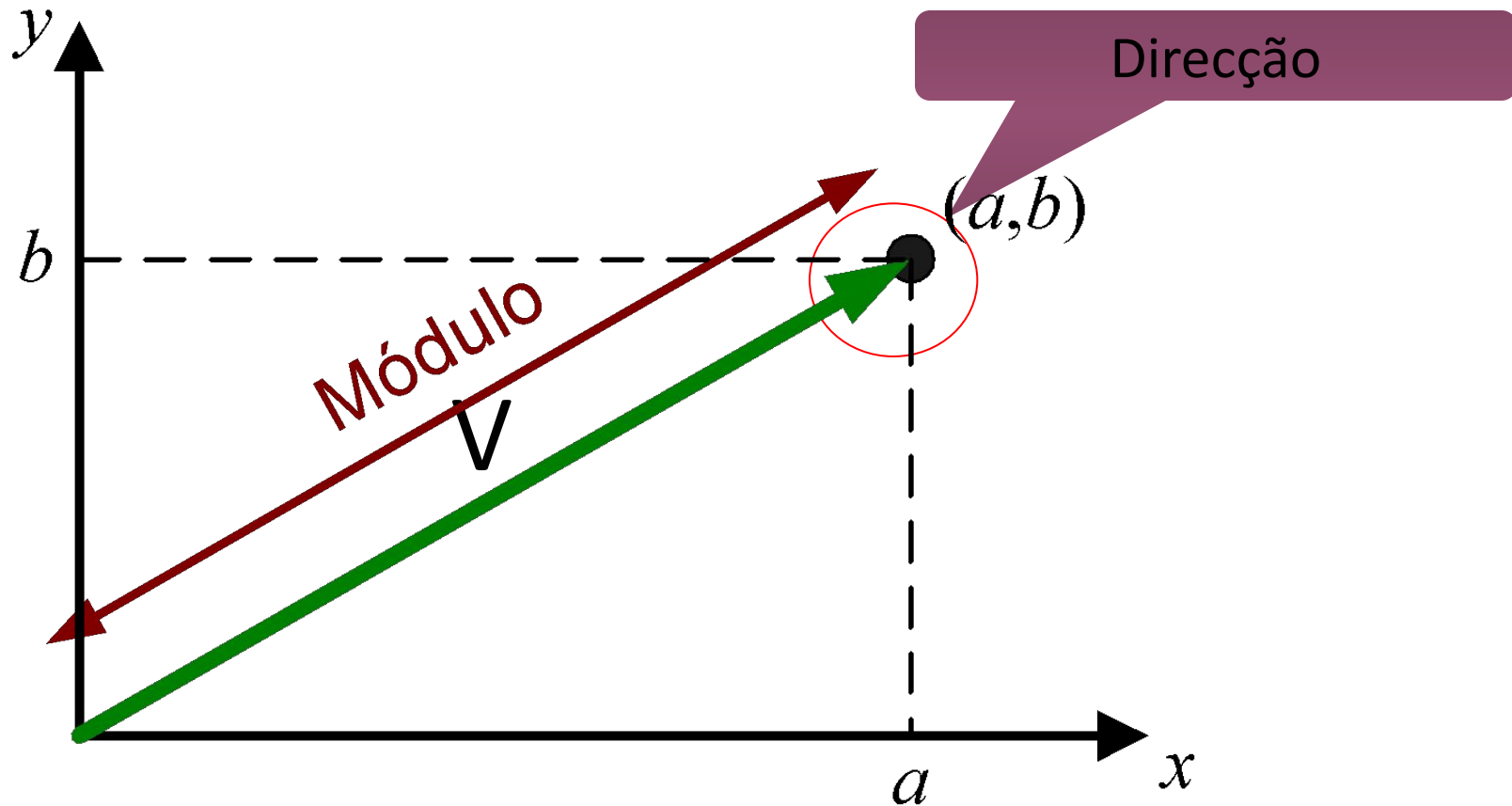


# Vetores e Matrizes



# Vetores

Conceito geométrico de vetor (duas dimensões)





# Vectores

- Da figura anterior pode-se concluir que bastam duas grandezas numéricas para representar um vetor num espaço de **duas** dimensões.

$(a, b)$



# Vectores

Num espaço com três dimensões são necessárias três grandezas:

$$(a, b, c)$$

Generalizando, um vector com  $N$  elementos pertence a um espaço com  $N$  dimensões.

Elementos de um espaço com mais de 3 dimensões são difíceis de representar graficamente.





# Vetores

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

```
>> v = [4 , 5 , 4 , 2 , 1 , 7]
```

```
>> v = [4 5 4 2 1 7]
```

Os elementos são separados por espaços ou vírgulas



# Matrizes

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

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

$$v = [1 \ 2 \ 3 \ 4]$$

$$n = [7]$$

Vector  
linha

Vector  
coluna



# Matrizes

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

```
» A= [4 , 5 , 4 ; 2 , 1 , 7]
```

```
» A= [4 5 4 ; 2 1 7]
```

Os elementos são separados por espaços ou vírgulas. Para mudar de linha, coloca-se um ponto e vírgula.



## Matrizes - Índices

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

Diagram illustrating the matrix  $A$  and its elements. The matrix is a 3x4 grid. The element 10 is highlighted with a red circle. A blue arrow points from the label  $A_{2,3}$  to the element 10. The indices 1, 2, 3, and 4 are shown below the columns, and 1, 2, and 3 are shown to the right of the rows.

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

$A_{2,3}$



# Transposta de uma matriz

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

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

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



# Definição funcional 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
- `ones(N,M)` gera uma matriz de uns com  $N$  linha e  $M$  colunas (bom para alocar matrizes)
- `rand(N,M)` gera uma matriz de elementos pseudo aleatórios com  $N$  linha e  $M$  colunas
- `magic(N)` gera um quadrado mágico de dimensão  $N$
- `eye(N)` gera uma matriz identidade de dimensão  $N$
- `randi(n_max, N, M)` gera uma matriz com números inteiros pseudo aleatórios de 1 a  $n\_max$  com dimensão  $N$  linhas e  $M$  colunas

## Exemplos

```
» A = eye(3)
```

A =

|   |   |   |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

```
» B = zeros(2,3)
```

B =

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |



# Concatenação

Com o Matlab é possível construir matrizes a partir de outras de menor dimensão. Eis alguns exemplos:

```
» x = [1 2; 3 4];
```

```
» A = [x x; x x]
```

A =

|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 |
| 1 | 2 | 1 | 2 |
| 3 | 4 | 3 | 4 |

```
» % Problema de consistência
```

```
» x = [1 2 3 4; 4 5 6]
```

```
??? = [1 2 3 4; 4 5 6]
```

Todas as linhas na matriz têm de ter os mesmos elementos.



# Representação de polinómios

Um polinómio pode ser representado no Matlab por um vector com os seus coeficientes. Vejamos um exemplo:

Este polinómio representa-se no Matlab como:

$p = [2, 0, -3, 9]$

$$p(x) = 2x^3 - 3x + 9$$

O termo nulo tem de ser representado de forma explícita





## Operações com polinómios

| Operação  | Matlab                      |
|---|-----------------------------|
| $p(x)+q(x)$   | <code>p+q</code>            |
| $p(x)\times q(x)$   | <code>conv (p, q)</code>    |
| raízes de $p(x)$  | <code>roots (p)</code>      |
| polinómio com as raízes $r_1, r_2,$<br>...                | <code>poly (r)</code>       |
| Valor do polinómio $p(x)$ para<br>vários valores de $x$ . | <code>polyval (p, x)</code> |



# Divisão e Sistemas de Equações

- Considere-se agora o sistema de equações

$$\begin{cases} 2u + v + w & = 5 \\ 4u - 6v & = -2 \\ -2u + 7v + 2w & = 9 \end{cases}$$

que se pode escrever na forma algébrica e resolver da mesma forma

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix} = \mathbf{Ax} = b \therefore x = \mathbf{A}^{-1}b$$

$\mathbf{A}$ 
 $x$ 
 $b$



# Divisão e Sistemas de Equações

- Considere a seguinte equação com uma incógnita
- Resolve-se fazendo

$$\begin{aligned}ax &= b \\ a^{-1}ax &= a^{-1}b \\ x &= a^{-1}b\end{aligned}$$



## Exemplos

- Resolução de um sistema de equações pelo método da eliminação Gaussiana utilizando divisão de matrizes

```
>> A = [2 1 1; 4 -6 0; -2 7 2];
```

```
>> b = [5 -2 9]';
```

```
>> x = A\b    %Left Division
```

- Resolução de um sistema de equações pelo cálculo directo da inversa de uma matriz

```
>> X = inv(A)*b    %Inverse of A
```

```
>> X = linsolve(A,b)
```



## O operador ":"

- O operador mais versátil do MATLAB
- Permite definir de forma compacta um conjunto de valores (vector) em progressão aritmética.

```
>> x = início: passo : fim
```

exemplo

```
>> x= 2:2:10
```

```
>> 2, 4, 6, 8, 10
```

```
>> x= linspace(2,10,5)
```

```
>> 2, 4, 6, 8, 10
```



# Tipos de dados elementares

- Vetores numéricos

```
» x = 1:10;  
» x = 1 + linspace(1,20,10)*j; % vetor complexo  
» y = (x >= 5) % vetor booleano  
y =  
    0    0    0    0    1    1    1    1    1    1
```

- Vetores de caracteres

```
» x = ['c','h','a','r']  
x = char  
» x = ['char']  
x = char
```



# Indexação

- Referência ao elemento  $i,j$  numa matriz

```
» A(3,2)
```

```
ans =
```

```
0.7621
```

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

```
» x = 1:2:50;
```

```
» x(10:15)
```

```
ans =
```

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

- Vectors de índices

```
» v1 = 10:15;
```

```
» x(v1)
```

```
ans =
```

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



# Índices lógicos

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  $\mathbf{x}$

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

$$-1 \ -3$$





# Dimensões

- Número de elementos dum vector ou matriz

```
» x = 1:10;  
» y = 3 + j*linspace(1,10,20);  
» dim_x = size(x), dim_y = size(y)  
dim_x =  
      1      10  
dim_y =  
      1      20  
» A = rand(3,2);  
» n_elementos = prod(size(A));  
» maior_dim = length(A); % maior dimensão matriz  
» first = A(1,1); %Primeira linha, primeira coluna  
» last = A(end,end); %Última linha, última coluna
```



# Aritmética

- 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);
```



# Multiplicação Aritmética

- Multiplicação aritmética “.\*” (“elemento a elemento”)

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

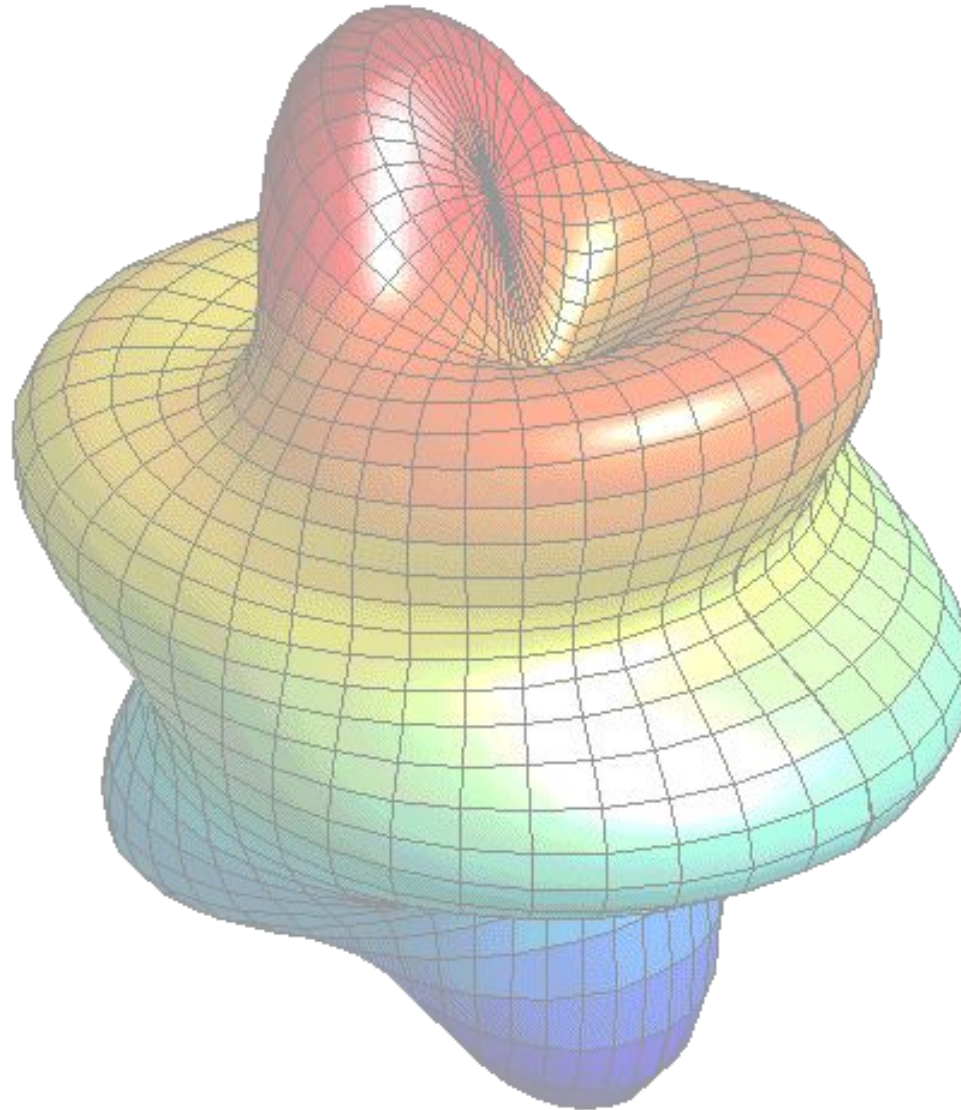
```
>> p = x .* y % Pointwise multiplication
```

```
p =
```

```
2      4     30     40
```



# Gráficos com o Matlab





# Gráficos de uma Variável

- Sintaxe do comando `plot`

```
v= rand(1,10) ;  
plot(v)
```

Nesta versão mais simples é desenhado um gráfico de linha contínua com a amplitude dos elementos do vector **v**. Nas abcissas aparecem os índices dos elementos de **v**.



# Sintaxe do comando `plot`

`plot(x1,y1,x2,y2,...)`

Os vectores das ordenadas **x1**, **x2**, ... podem ter um número diferente de elementos.

O número de elementos dos pares (**x1**, **y1**) e (**x2**, **y2**) deve ser o mesmo.

Exemplo:

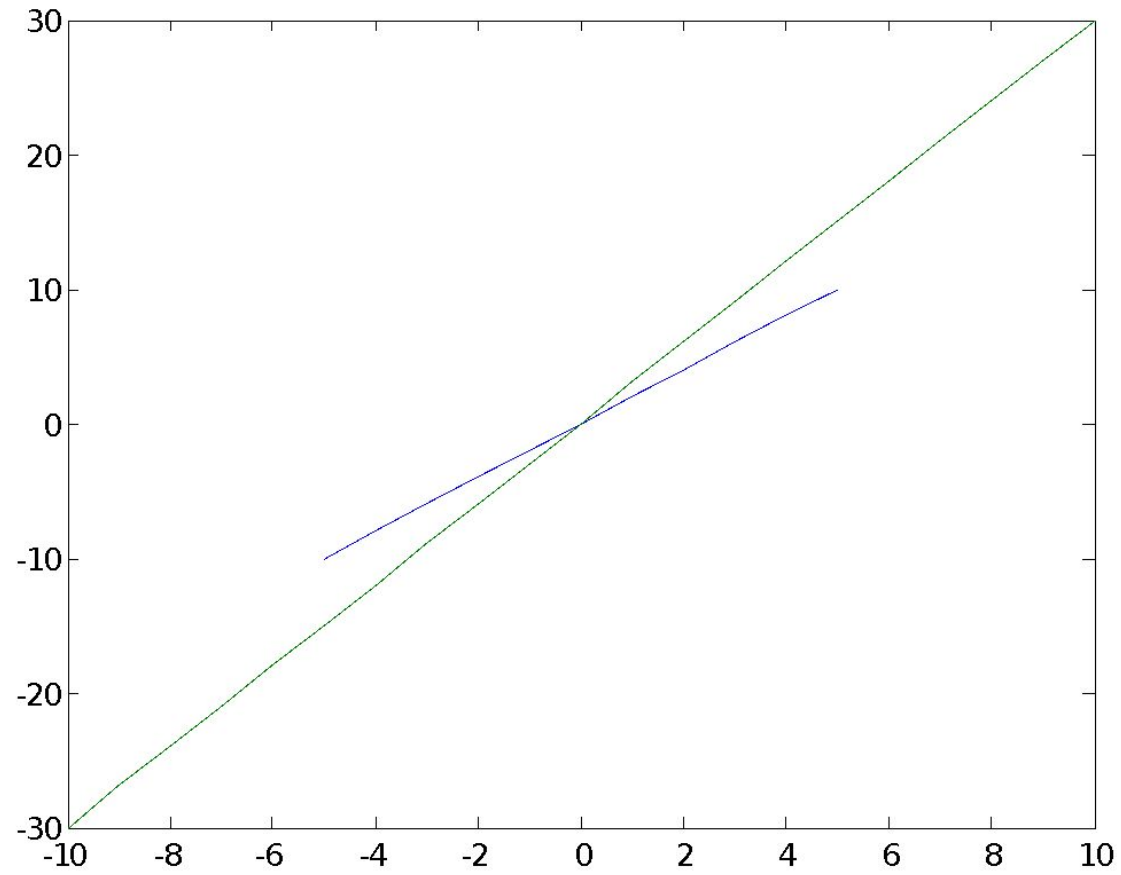
```
x1= -5:5; x2= -10:10
```

```
y1= 2*x1; y2=3*x2;
```

```
plot(x1,y1,x2,y2)
```



# Exemplo





## Sintaxe do comando **plot**

Alternativamente, podemos usar a *keyword* **hold on**, para obter o mesmo resultado. Assim, o código ficaria,

```
x1= -5:5; x2= -10:10  
y1= 2*x1; y2=3*x2;  
plot(x1,y1)  
hold on  
plot(x2, y2)
```





## Alteração do aspecto gráfico

Para além dos argumentos vetoriais 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 vetores dos pontos.

```
plot(x1,y1,'string1',x2,y2,'string2',...)
```

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



# Caracteres definidores de atributos

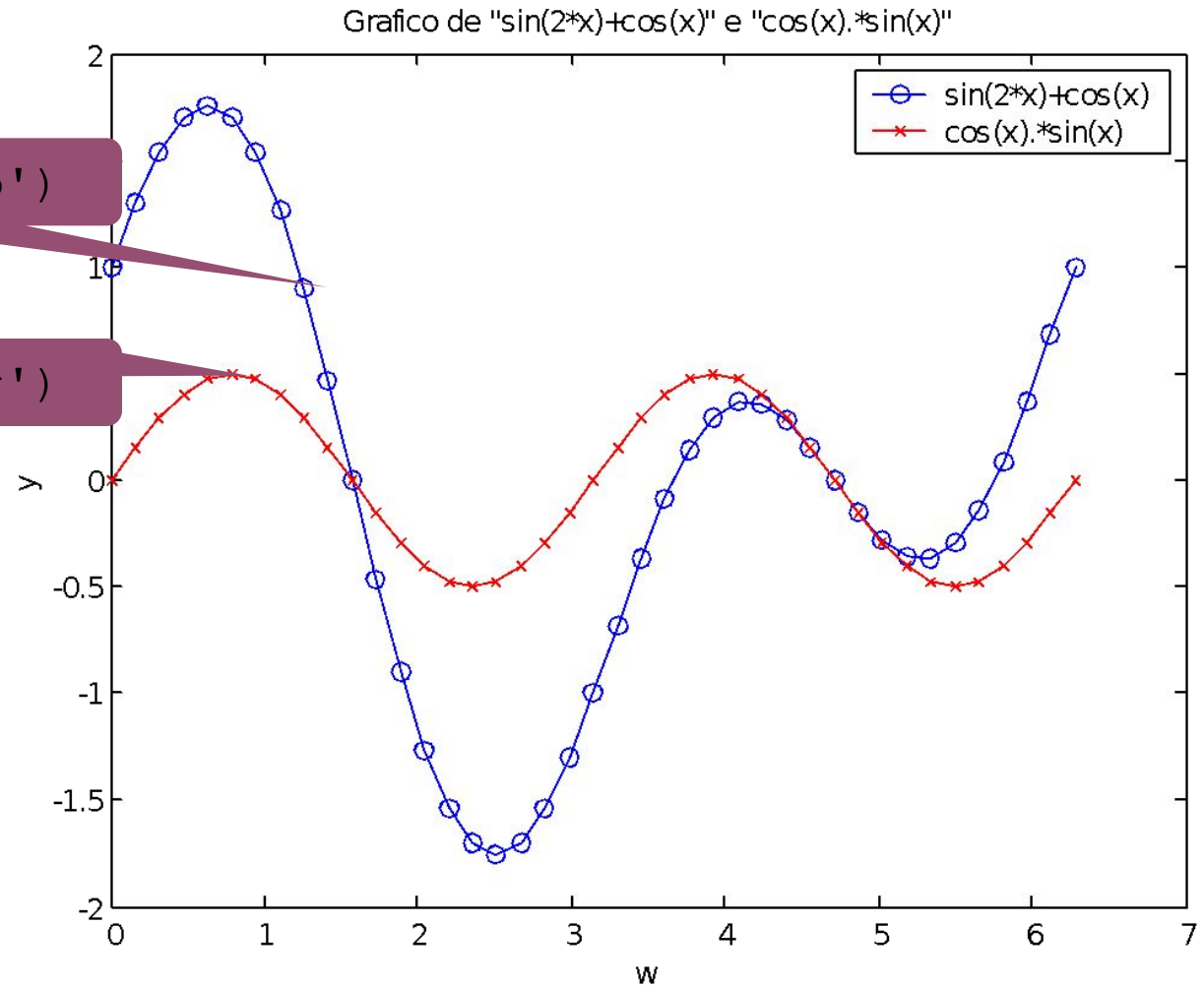
| Cor |           | Marcadores |                      | Linhas |               |
|-----|-----------|------------|----------------------|--------|---------------|
| y   | amarelo   | .          | ponto                | -      | linha a cheio |
| m   | rosa      | o          | círculo              | :      | pontuada      |
| c   | ciano     | x          | marca x              | -.     | traço ponto   |
| r   | encarnado | +          | marca mais           | --     | tracejada     |
| g   | verde     | *          | estrela              |        |               |
| b   | azul      | s          | quadrado             |        |               |
| w   | branco    | d          | diamante             |        |               |
| k   | preto     | v          | triângulo (cima)     |        |               |
|     |           | ^          | triângulo (baixo)    |        |               |
|     |           | <          | triângulo (esquerda) |        |               |
|     |           | >          | triângulo (direita)  |        |               |
|     |           | p          | pentagrama           |        |               |
|     |           | h          | "hexagram"           |        |               |



# Alteração do aspecto gráfico

`plot(x1,y1, '-ob')`

`plot(x2,y2, '-+r')`





# Subplots

```
subplot(2,1,1);  
x = linspace(0,10);  
y1 = sin(x);  
plot(x,y1)
```

```
subplot(2,1,2);  
y2 = sin(5*x);  
plot(x,y2)
```

