

## Matrizes e vetores

No Scilab existem várias formas de criar matrizes e operar com as mesmas. Nesta ficha prática são explorados métodos para definir e manipular matrizes, aceder aos seus elementos ou realizar operações elemento a elemento.

- De seguida apresentam-se os símbolos necessários à criação de uma matriz de uma forma simples:

parêntesis retos	[ ]	Indicam o início e o fim da matriz
vírgula	,	Separa elementos pertencentes à mesma linha
ponto e vírgula	;	Indica a quebra/passagem para a linha seguinte

### Exemplo:

```
-->A = [1 , 2 , 3 ; 4 , 5 , 6]
A =
1.  2.  3.
4.  5.  6.
```

Uma forma alternativa de definir uma matriz  $m \times n$  sem fazer uso da vírgula ou do ponto e vírgula consiste em separar os elementos da mesma linha usando o *space* e definir a mudança de linha usando o *return*:

```
A=[a11 a12 ... a1n
   a21 a22 ... a2n
   ...
   am1 am2 ... amn]
```

### Exemplo:

```
-->A = [1 2 3
-->4 5 6]
A =
1.  2.  3.
4.  5.  6.
```

- No seguinte quadro são apresentadas funções que permitem a criação de certos tipos de matrizes:

Matrizes específicas	
<code>eye(m,n)</code>	Matriz $m \times n$ com 1's na diagonal e 0's nas restantes posições
<code>ones(m,n)</code>	Matriz $m \times n$ com todos os elementos iguais a 1
<code>zeros(m,n)</code>	Matriz $m \times n$ com todos os elementos iguais a 0

### Exemplo:

```
-->A = ones(2,3)
A =
1.  1.  1.
1.  1.  1.
```

- No seguinte quadro são apresentadas funções que permitem obter informação sobre uma matriz ou criar uma nova matriz a partir de outra:

<code>A</code>	Apresenta a matriz <code>A</code>
<code>size(A)</code>	Permite saber a dimensão (n.º de linhas e n.º de colunas) da matriz <code>A</code>
<code>matrix(A,m,n)</code>	Cria uma nova matriz a partir de <code>A</code> , redistribuindo os <code>k</code> elementos desta matriz por <code>m</code> linhas e <code>n</code> colunas (de notar que <code>m×n</code> tem que ser igual a <code>k</code> )
<code>resize_matrix(A,m,n)</code>	Cria uma nova matriz com dimensão <code>m×n</code> , truncando as linhas e/ou colunas da matriz original <code>A</code> , ou completando as linhas/colunas em falta com 0's

#### Exemplos:

```
-->B=eye(3,3)
```

```
B=
```

```
1.  0.  0.
0.  1.  0.
0.  0.  1.
```

```
-->size(B)
```

```
ans =
```

```
3.  3.
```

```
-->C=resize_matrix(B,3,2)
```

```
C=
```

```
1.  0.
0.  1.
0.  0.
```

- No seguinte quadro são apresentadas algumas funções que permitem trabalhar com determinado elemento, determinada linha/coluna ou determinada parte de uma matriz:

<code>A(i,j)</code>	Apresenta o elemento da matriz <code>A</code> que se encontra na linha <code>i</code> e coluna <code>j</code>
<code>A(i,j)=k</code>	Substitui o elemento da matriz <code>A</code> que se encontra na linha <code>i</code> e coluna <code>j</code> por <code>k</code>
<code>A(:, :)</code>	Apresenta a matriz <code>A</code>
<code>A(i, :)</code>	Apresenta a linha <code>i</code> da matriz <code>A</code>
<code>A(:, j)</code>	Apresenta a coluna <code>j</code> da matriz <code>A</code>
<code>A(i:j,k)</code>	Apresenta os elementos da matriz <code>A</code> que se encontram na coluna <code>k</code> , da linha <code>i</code> até à linha <code>j</code>
<code>A(i, :)=[]</code>	Faz com que a linha <code>i</code> da matriz <code>A</code> seja truncada

#### Exemplos:

```
-->A = [1 , 2 , 3 ; 4 , 5 , 6]
```

```
A =
```

```
1.  2.  3.
4.  5.  6.
```

```
-->A(2 ,1)
```

```
ans =
```

```
4.
```

```
-->A(12 ,1)
```

```
!-- error 21
```

```
Invalid index .
```

```
-->A(2,:)
```

```
ans=
```

```
4.  5.  6.
```

```
-->A(:,3)
```

```
ans=
```

```
3.
```

```
6.
```

```
-->A =[ 1 2 3
```

```
4 5 6
```

```
7 0 0]
```

```
A=
```

```
1.  2.  3.
```

```
4.  5.  6.
```

```
7.  0.  0.
```

```
A(2,3)=9
```

```
A=
```

```
1.  2.  3.
```

```
4.  5.  9.
```

```
7.  0.  0.
```

```
-->A(: ,3) = []
```

```
A =
```

```
1.  2.
```

```
4.  5.
```

```
7.  0.
```

```
-->B = matrix (A ,1 ,6)
```

```
B =
```

```
1.  4.  7.  2.  5.  0.
```

- No Scilab podem ser aplicados a matrizes vários operadores: a adição, a subtração, a multiplicação e a potência (os operadores usuais).

Operadores com matrizes	
+	Adição
−	Subtração
*	Multiplicação
^	Potência

**Exemplo:**

```

A=[3,-24,30];
B=[
  9 -36 30
 -36 192 -180
 30 -180 180
];

-->A*B
ans =
1791.    - 10116.    9810.

```

- Além destes, existem as operações de multiplicação e potenciação elemento a elemento. Neste caso, deve ser adicionado um ponto antes do símbolo do operador. Por exemplo, sendo  $X$  e  $Y$  matrizes da mesma dimensão, a multiplicação elemento a elemento significa que o resultado de  $Z = X.*Y$  será dado por

$$Z(i,j) = X(i,j) * Y(i,j)$$

e não pela regra habitual da multiplicação de matrizes. Estas operações têm vantagens, por exemplo, quando é necessário realizar operações sobre matrizes de grande dimensão.

Operadores aritméticos sobre matrizes com a mesma dimensão (elemento a elemento)	
.*	Multiplicação elemento a elemento
.^	Potência elemento a elemento

Funções Matriciais e Álgebra Linear Numérica	
A'	Transposta da matriz A
rank(A)	Característica da matriz A
det(A)	Determinante da matriz quadrada A
trace(A)	Soma dos elementos da diagonal principal da matriz quadrada A
inv(A)	Inversa da matriz quadrada A de determinante não nulo

**Exemplos:**

```

A=[3,-24,30];
B=[ 9 -36 30
 -36 192 -180
 30 -180 180];

```

```

-->det(B)
ans =
2160.

```

```

-->rank(B)
ans =
3.

```

# Programação

O Scilab contém diversos comandos para controlar a sequência dos programas, como por exemplo o teste de condições **if** ou os ciclos de controlo. Os ciclos permitem repetir instruções um número pré-determinado de vezes ou até que uma condição se verifique.

- Instrução Condicional: **If ... else ... end**

O teste **if** avalia uma proposição e executa um grupo de instruções se essa proposição for verdadeira. Se a proposição tiver valor lógico falso, as instruções a executar podem opcionalmente ser definidas através do comando **else**. O comando **end** termina o teste. A sintaxe do teste **if** é:

```
if expressão lógica 1,
    instruções executáveis se a expressão lógica 1 for verdadeira
else
    instruções executáveis se a expressão lógica 1 for falsa
end
```

Exemplo:

```
a=cos(%pi/5)+ log(1/2)-exp(2);
if a>0
    s=a;
else
    s=0;
end
```

- Ciclo de controlo: **for ... end**

Este ciclo consiste em atribuir um valor inicial a uma variável, executar instruções especificadas e incrementar a variável, ou em 1 (por defeito), ou em determinado passo. O ciclo repete-se até que seja atingido ou ultrapassado um valor definido inicialmente. O ciclo termina com a instrução **end**. A sintaxe do ciclo **for** é:

```
for índice = início:incremento:fim
    instruções executáveis
end
```

Exemplo:

```
for i=1:3
    for j=1:3
        H(i,j)=1/(i+j-1);
    end
end
```

- Ciclo de controlo: **while ... end**

Neste ciclo de controlo as instruções especificadas são executadas enquanto a expressão lógica for verdadeira. A sua sintaxe é:

```
while expressão lógica,
    instruções executáveis
end
```

Exemplo:

```
s=1; i=0;
while s < 19/10
    s = s + 2^(-i)
    i=i+1;
end
```

Exemplos de outros comandos relacionados com ciclos: **pause**, **resume**, **return** e **abort** (ver **HELP**).

## Funções

- Quando for necessário definir uma função, teremos em consideração a seguinte estrutura:

```
out_var=myfunction(in_var)
```

onde:

\* **out\_var** é o nome da variável que contém os dados de saída (por exemplo **y**)

\* **myfunction** é o nome da função (por exemplo **f**)

\* **in\_var** é o nome da variável que contém os dados de entrada (por exemplo **x**)

Assim:

$$y=f(x)$$

Caso a função tenha **n** dados de entrada e **m** dados de saída, devemos ter em consideração a seguinte forma:

```
[out_1,...,out_n]=myfunction(in_1,...,in_m)
```

- A função é implementada através dos comandos **function** e **endfunction**.

```
function [lista de parâmetros de saída] = nome_função(lista de parâmetros de entrada)
instruções
endfunction
```

### Exemplo:

```
function y = f(x)
y = 2*x
endfunction
```

**Exemplo:** Função soma de dois números: nome da função **fun**, parâmetros de entrada **a** e **b** e parâmetro de saída **y**.

```
function y = fun(a,b)
//fun(a,b) - Soma de a + b
y=a+b;
endfunction
```

**Exercício:** Guarde a função que implementou anteriormente num ficheiro SciNotes designado por *Exemplo\_função.sce*.

Estas formas de definir uma função podem ser implementadas, quer na janela de comandos, quer num ficheiro ".sce". Neste último caso, é relevante salientar que, antes da função ser usada, é necessário executar o ficheiro onde a função foi definida.

```
-->exec ("C:\...\Exemplo_função.sce")
--> f(6)
ans =
12.
```

## Visualização gráfica

- O Scilab tem muitas opções relativas à criação e edição de figuras e gráficos, sejam estes em  $2D$  ou em  $3D$ . Os gráficos em  $3D$  podem ser gerados usando o comando **surf** (ver help), enquanto que, para criar gráficos em  $2D$ , usamos o comando **plot**. Este permite diferentes possibilidades de utilização.

<b>plot(Y)</b>	Representa a lista de pontos $(X,Y)$ cujas ordenadas são os elementos do vetor $Y$ e as abcissas são os índices de 1 a $n$ , onde $n$ é o número de elementos do vetor $Y$
<b>plot(X,Y)</b>	Representa a lista de pontos $(X,Y)$ cujas abcissas são os elementos do vetor $X$ e as ordenadas são os elementos do vetor $Y$
<b>plot(X,Y,'S')</b>	Igual ao <b>plot(X,Y)</b> onde <b>S</b> é uma sequência de caracteres constituída por elementos das colunas da tabela seguinte, escritos entre plicas

Nota: O vector **X** pode ser definido usando qualquer um dos comandos de definição de vectores estudados anteriormente.

Caso tenha sido definida uma função **f**, o vector **Y** pode ser definido pelo comando **Y=f(X)**.

A seguinte tabela apresenta os caracteres correspondentes às várias características do gráfico que são possíveis de definir através da string '**S**', usando o comando **plot(X,Y,'S')**:

Símbolo	Cor	Símbolo	Marcador	Símbolo	Tipo de linha
<b>b</b>	azul	<b>.</b>	ponto	<b>-</b>	linha contínua
<b>g</b>	verde	<b>o</b>	círculo	<b>---</b>	linha tracejada
<b>r</b>	vermelho	<b>x</b>	X	<b>-.</b>	traço e ponto
<b>c</b>	ciano	<b>+</b>	+	<b>:</b>	linha pontilhada
<b>m</b>	magenta	<b>*</b>	asterisco		
<b>y</b>	amarelo	<b>s</b>	quadrado		
<b>k</b>	preto	<b>d</b>	losango		
<b>w</b>	branco	<b>v</b>	triângulo para baixo		
		<b>^</b>	triângulo para cima		
		<b>&lt;</b>	triângulo para a esquerda		
		<b>&gt;</b>	triângulo para a direita		
		<b>p</b>	pentagrama		

- O Scilab permite personalizar os gráficos, com diversas opções relacionadas com o controle dos eixos e com a inserção de texto em gráficos. Para saber como funciona cada uma das opções apresentadas a seguir consulte o menu HELP.

Inserção de texto em gráficos	
<b>legend</b>	Legenda das curvas do gráfico
<b>title</b>	Título do gráfico
<b>xlabel</b>	Legenda do eixo dos xx
<b>ylabel</b>	Legenda do eixo dos yy
<b>xtitle("T1","T2","T3")</b>	Adiciona o título do gráfico ( <b>T1</b> ), o título do eixo dos xx ( <b>T2</b> ) e o título do eixo dos yy ( <b>T3</b> )

O controle das definições dos eixos e outras definições do gráfico pode ser feito diretamente através do editor da janela gráfica (Graphic Window).

No seguinte quadro são apresentados comandos usados na definição da janela gráfica:

<b>figure(n)</b>	Esboça o gráfico da função na janela <b>n</b>
<b>clf</b>	Apaga as figuras anteriores

### Exemplo:

```
function y = f(x)
    y = x.^2
endfunction
xdata = linspace (1,10,50);
ydata = f(xdata);
plot (xdata,ydata,"+ g")
xlabel ("Função quadrática ", "eixo dos xx", "eixo dos yy");
legend ("y=x^2");
```

## Exercícios

1. Considere as seguintes matrizes:

$$A = \begin{pmatrix} 2 & 1 & -1 \\ -1 & 2 & 1 \\ 1 & 3 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & -2 & 2 \\ 0 & 2 & 3 & 0 \\ 1 & 0 & -1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & -1 & 0 & 2 \end{pmatrix}.$$

- (a) Determine, se possível:

$$\begin{array}{llll} \text{(i)} & AB; & \text{(ii)} & CD; & \text{(iii)} & CD^T; & \text{(iv)} & AB^T; \\ \text{(v)} & A^{-1}C; & \text{(vi)} & 3AB + 2CD^T; & \text{(vii)} & A.^2; & \text{(viii)} & A^2; \\ \text{(ix)} & \det(AA^T); & \text{(x)} & BDC^T; & \text{(xi)} & A^{-2}; & \text{(xii)} & A^2B^T. \end{array}$$

- (b) Obtenha todos os elementos de todas as linhas que se encontram entre as colunas 2 e 3 da matriz  $C$ .

2. Faça a representação gráfica da função polinomial definida por

$$f(x) = x^5 - 3x^4 - 3x^3 + 7x^2 + 6x$$

no intervalo  $[-1, 5; 2, 5]$  com incremento de 0,125.

3. Seja  $f$  a função definida por  $f(x) = \cos(x) + e^x$ .

- (a) Defina/implemente a função  $f$ .
- (b) Determine o valor de  $f(0)$  e de  $f(\pi)$ .
- (c) Construa um vetor linha  $X$  cujo primeiro elemento seja 1, o último seja 2 e tenha um total de 98 elementos.
- (d) Calcule  $f(X)$ . O que acontece?
- (e) Faça a representação gráfica da função  $f$

4. O polónio tem uma meia-vida de 140 dias, o que significa que, devido ao decaimento radioactivo, a quantidade de polónio restante depois de 140 dias é metade da original. A quantidade restante, após um certo período de tempo  $t$ , é dada por

$$r(t) = C_0(0.5)^{t/v}, \quad (t \text{ em dias})$$

sendo  $C_0$  a quantidade inicial e  $v$  o tempo de meia-vida. Se hoje tivermos 10 gramas de polónio, qual a quantidade restante ao fim de cada uma das próximas 10 semanas? Elabore um gráfico que mostre o comportamento observado ao longo desse período.



5. Considere a sucessão  $Z$  definida por  $Z(n) = \sum_{k=1}^n \left(\frac{1}{2}\right)^k$ . Usando comandos do Scilab:
- (a) faça uma representação gráfica de  $Z$  onde se possam visualizar os seus 20 primeiros termos.
  - (b) determine o primeiro valor de  $n$  tal que seja verificada a condição  $|Z(n) - Z(n-1)| < 10^{-10}$ .
6. Elabore um algoritmo que permita determinar a nota final (por avaliação periódica) de um estudante na UC de Matemática Discreta. Os elementos de avaliação, os respectivos pesos e a nota mínima estão descritos na tabela seguinte:

	Nº de provas	Peso	Nota mínima
Componente Teórica	2 provas ( $PT1, PT2$ )	60%	8.0 na média das 2 provas teóricas
Componente prática	1 prova ( $PP$ )	40%	8.0 na prova prática única

Apenas devem ser pedidas ao utilizador as notas parciais.

#### Algoritmo - Pseudo-código

- Dados de entrada:  $nota\_PT1$ ,  $nota\_PT2$  e  $nota\_CP$
- Dados de saída: classificação final.
- Definir  $nota\_CT = (nota\_PT1 + nota\_PT2) * 0.5$
- Se  $nota\_CT < 8.0$  ou  $nota\_CP < 8.0$  então
  - “Não atingiu o mínimo numa das componentes.”
- Caso contrário
  - $resultado = 0.60 * nota\_CT + 0.40 * nota\_CP$ .
  - “A classificação final na UC é  $resultado$ .”