Métodos Computacionais para a Engenharia Eletrotécnica

Ficha Laboratorial 01 – Introdução ao MATLAB

1. Objetivo

Pretende-se com este trabalho fazer uma introdução à ferramenta informática MATLAB. Trata-se de um trabalho constituído por um conjunto de atividades, cada uma delas orientada a explorar alguns dos conceitos básicos e sintaxe do MATLAB.

O objetivo do presente trabalho laboratorial é, fundamentalmente, a introdução dos seguintes tópicos:

1. O ambiente de trabalho do MATLAB.
2. Sintaxe básica para criação de vetores e matrizes.
3. Operações básicas: operações matriciais e operações elemento-a-elemento (*elemento-wise*).
4. Edição de *scripts* básicos.

Deverá apresentar ao docente a resolução de cada uma das atividades na forma de *script*. No entanto, os alunos deverão experimentar a resolução dos problemas na janela de comando, copiando depois as linhas de comando para o ficheiro de *script* referente à atividade que está a ser desenvolvida.

2. MATLAB

O MATLAB, abreviatura de *MATrix LABoratory*, é uma plataforma de programação desenvolvida para engenheiros e cientistas, que tem por base, e tal como o nome sugere, uma linguagem baseada em matrizes. As primeiras aplicações do MATLAB prendiam-se com problemas de álgebra linear e análise numérica. Com o passar do tempo, o MATLAB sofreu melhoramentos e atualizações, tendo sendo adicionadas novas funcionalidades que o tornariam numa ferramenta fundamental nas áreas de educação, investigação e desenvolvimento. Atualmente, é possível encontrar aplicações de MATLAB em áreas tão diversas como engenharia de controlo (uma das primeiras áreas em que foi adotado), processamento de sinal, processamento de imagem, estatística, finanças, controlo difuso, neuronal e inteligente, e *machine learning,* apenas para nomear algumas.

O presente trabalho pretende fazer uma abordagem sucinta a esta ferramenta informática, onde será introduzido o ambiente de trabalho, a sintaxe elementar, e algumas funções básicas.

É importante realçar que este trabalho não dispensa, de forma alguma, a leitura da bibliografia recomendada sobre MATLAB.

3. Instalação do MATLAB

A Universidade de Coimbra tem uma licença de *campus* do MATLAB, da qual podem usufruir os alunos, podendo estes instalar uma cópia nos seus computadores pessoais. Se não pretenderem fazer uma instalação deste software (que é exigente em recursos), podem, em alternativa, usar a versão *web* da aplicação, disponível em https://matlab.mathworks.com/. No caso de se optar por esta solução, existem algumas ligeiras nuanças

na sua utilização quando comparado com a versão pessoal, mas nada de relevante que impeça a aprendizagem correta da ferramenta.

Para instalar o MATLAB e/ou aceder à respetiva versão online, os alunos deverão fazer o registo no *site* da MathWorks, em https://www.mathworks.com/, utilizando o endereço de correio eletrónico da Universidade de Coimbra (@student.uc.pt). Para informações mais detalhadas, por favor consulte a página de suporte do Gabinete de Rede Informática (GRI) do DEEC, em https://kb.deec.uc.pt/books/deec/page/matlab-alunos. Em caso de dificuldades neste processo, contacte o GRI através do endereço de correio eletrónico helpdesk@deec.uc.pt, ou contacte o docente.

4. Ambiente de trabalho

Ao iniciar uma sessão do MATLAB, surge uma janela em todo semelhante à apresentada na Figura 1, dividida em três áreas distintas[1]. Ao centro, encontra a **Janela de comandos**, ou *Command Window*, através da qual o utilizador pode introduzir e executar comandos e funções de MATLAB. Nesta janela deverá estar visível o *prompt* do MATLAB, o sinal gráfico `>>`, o qual informa ao utilizador que o MATLAB está pronto para receber um comando.

No lado esquerdo, encontra-se a janela referente à **Pasta atual** (*Current Folder*) onde é apresentado o conteúdo da pasta corrente de trabalho, assim como informações sobre o ficheiro que estiver, nesse momento, selecionado.

No lado direito, encontra-se a janela do **Espaço de Trabalho** (*Workspace*). Nesta janela, é apresentada uma lista das variáveis criadas até ao momento pelo utilizador, assim como as suas respetivas propriedades. A partir daqui, também é possível modificar não só o valor atribuído a cada variável mas também o nome, ou etiqueta, de cada variável.

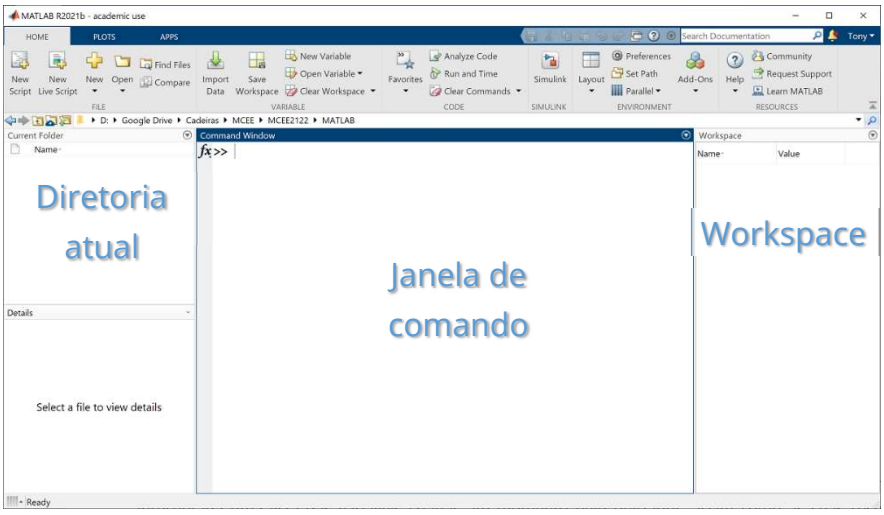

Figura 1 – Ambiente de trabalho do MATLAB

5. Operações básicas

Para executar operações algébricas básicas, basta introduzir a expressão que se pretende avaliar na janela de comando, a seguir ao *prompt*, `>>`, e premir `<enter>`:

```
>> 10 + 20
```

[1] A versão do MATLAB seguida nesta ficha é a R2021b. Neste caso, a Figura 1 diz respeito à configuração por omissão (*default*) do ambiente de trabalho. Eventualmente, a versão na qual está a trabalhar pode não ter uma apresentação idêntica à indicada na figura. Em todo o caso, e tratando-se de um programa em ambiente Windows, a organização desta janela pode ser configurada a partir do botão *Layout*, no separador *Home* da barra de ferramentas.

```
ans =
    30

>> 2^5
ans =
    32

>> 2*cos(pi/4)
ans =
    1.4142
```

Também é possível atribuir um valor a uma **variável**, sendo usado o símbolo igual (=) como operador de atribuição:

```
>> a = 10
a =
    10

>> b = 20
b =
    20

>> c = a + b
c =
    30

>> d = a^2 + b^2
d =
   500
```

Note que o nome de **cada variável deve começar sempre por uma letra, sendo permitidos apenas letras, números e o caractere sublinhado (*underscore*)**. Note ainda que o MATLAB é sensível aos carateres maiúsculos e minúsculos, e.g., a variável `var1`, é diferente de `Var1` ou `VAR1`.

6. Vetores e matrizes

6.1 Vetores

Tal como foi referido no início deste texto, o MATLAB é uma aplicação que permite a manipulação matricial. Os valores escalares utilizados nos exemplos anteriormente apresentados, são tratados como matrizes 1x1. Para definir um vetor-linha podemos, por exemplo, escrever:

```
>> v1 = [1 2 3 4]
v1 =
     1     2     3     4
```

Note que os elementos do vetor são delimitados por **parêntesis retos**, sendo cada coluna separada por um **espaço**. Alternativamente, a separação de colunas pode ser feita com recurso a **vírgulas**:

```
>> v2 = [10, 20, 30, 40];
```

Uma chamada de atenção para o **ponto-e-vírgula no final da linha de comando**. Ao terminar o comando com este sinal, evita o "eco" da operação, isto é, a variável `v2` é criada, mas não vemos o resultado da operação, ao contrário ao que aconteceu ao criar a variável v1, no exemplo imediatamente anterior.

Um vetor-coluna pode ser definido da seguinte forma:

```
>> v3 = [1
2
3];
```

ou

```
>> v4 = [11; 22; 33];
```

Repare que, neste caso, **as linhas podem ser separadas com recurso ao <enter>, ou separadas com o sinal ponto-e-vírgula**. No caso de as linhas serem separadas através da tecla <enter>, note que, enquanto não for introduzido o parêntesis reto a fechar o vetor, o *prompt* não é apresentado na janela de comando.

O operador **transposição** é representado por uma plica ('). Assim, podemos, por exemplo, simplesmente transpor um vetor-linha para obter um vetor-coluna:

```
>> v5 = [100  200  300]';   % note a colocação da plica, '
```

De salientar, no exemplo anterior, que qualquer informação colocada a seguir ao sinal de percentagem, `%`, é considerado um comentário.

Por vezes, é bastante comum a necessidade de definir vetores que apresentam valores igualmente espaçados:

```
>> v6 = 1:6    % valores de 1 a 6 com incrementos de 1
v6 =
     1     2     3     4     5     6

>> v7 = 0:2:10 % valores de 0 a 10 com incrementos de 2
v7 =
     0     2     4     6     8    10
```

Outra forma de gerar vetores com valores igualmente espaçados é com recurso à função `linspace`. Para gerar *n* pontos igualmente espaçados entre os valores `x1` e `x2` podemos escrever `linspace(x1, x2, n)`:

```
>> v8 = linspace(0, 10, 6) % 6 valores de 0 a 10 com incrementos de (10-0)/(6-1) = 2
v8 =
     0     2     4     6     8    10
```

6.2 Matrizes e operações com matrizes

A criação de uma matriz não passa de uma extensão da definição de vetores, tal como atrás apresentado:

```
>> A = [1 2 3
4 5 6
7 8 9];
```

ou

```
>> B = [9 8 7; 6 5 4; 3 2 1]; % matriz B de dimensão 3x3
```

As operações de soma, subtração, divisão e multiplicação são matriciais, pelo que a dimensão das matrizes é um aspeto importante a ter em conta, caso contrário obterá um erro.

Para além das operações matriciais, é possível realizar **operações elemento a elemento**, designadas na literatura por operações *element-wise*. Estas operações, associadas à multiplicação, divisão e exponenciação,

implicam a utilização de um ponto antes do respetivo operador: **.*** (multiplicação elemento a elemento), **./** (divisão elemento a elemento) e .^ (exponenciação elemento a elemento).

```
>> v1 = [1 2 3; 4 5 6];
>> v2 = [10 20 30; 5 10 15];

>> a = v1 .* v2 % a(i, j) = v1(i, j) * v2(i, j)
a =
    10    40    90
    20    50    90
>> b = v1 ./ v2 % b(i, j) = v1(i, j) / v2(i, j)
b =
    0.1000    0.1000    0.1000
    0.8000    0.5000    0.4000

>> c = a.^2 % c(i, j) = a(i, j)^2
c =
         100        1600        8100
         400        2500        8100
```

Para além do operador divisão, que em MATLAB é designado de divisão à direita (/), existe ainda o operador *divisão à esquerda*, identificado com o caractere *blackslash* (\). Este operador é extremamente útil para a resolução de sistemas de equações lineares, $Ax = b$, em que a solução pode ser facilmente obtida calculando $x = A^{-1}b$, ou `x = A\b`.

7. Indexação de matrizes

A indexação de matrizes consiste na possibilidade de selecionar um subconjunto de elementos de uma dada matriz, i.e., selecionar uma sub-matriz. Nesta secção, será brevemente tratada a indexação com recurso aos índices dos elementos que constituem a matriz.

7.1 Indexação de vetores

Os elementos de um vetor estão guardados de forma indexada, i.e., o primeiro elemento do vetor tem associado o índice 1, o segundo o índice 2, e assim sucessivamente. Então, podemos aceder aos elementos de um vetor recorrendo a um valor de índice inteiro:

```
>> v = [18  20  14  17  6  9  12];

>> v(2)             % obtém o elemento de índice 2
ans =
    20

>> v([1 5 7])       % obtém o primeiro, quinto e sétimo elementos
ans =
    18     6    12

>> v([end-1 end])   % obtém o penúltimo e o último elementos
ans =
     9    12

>> v(1:2:end)       % obtém todos os elementos de índice ímpar
ans =
    18    14     6    12
```

Note a utilização da palavra-chave **end** em alguns dos exemplos acima apresentados. Esta palavra-chave serve, neste caso, para indicar o último termo do vetor.

Da mesma forma que podemos aceder a elementos de um vetor, também podemos alterar um ou vários dos seus valores:

```
>> v = [18  20  14  17  6  9  12];
>> v(4) = 100        % altera o valor de índice 4
v =
    18    20    14   100     6     9    12

>> v(1:3) = [11 22 33]  % altera os três primeiros elementos
v =
    11    22    33   100     6     9    12
```

Notar que, no último exemplo, o número de elementos do lado direito da expressão tem que coincidir com o número de elementos indexados no lado esquerdo da expressão, caso contrário é devolvido um erro.

7.2 Indexação de matrizes

A indexação de matrizes pode ser vista como uma extensão da indexação de vetores, recorrendo-se a dois índices: o primeiro índice corresponde ao número da linha que se pretende aceder, e o segundo corresponde ao número da coluna. Com isto presente, o procedimento é bastante semelhante ao que já foi anteriormente discutido sobre vetores.

Considere-se, como exemplo, uma matriz *A* gerada a partir da função `magic()`[2]:

```
>> A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> A(2, 3)       % obtém o elemento A(2, 3)
ans =
    10

>> A(1:3, 3:4)   % obtém os elementos comuns às três primeiras linhas e às colunas 3 e 4
ans =
     3    13
    10     8
     6    12

>> A(3, :)      % obtém a terceira linha de A
ans =
     9     7     6    12

>> A(:, 2)      % obtém a segunda coluna de A
ans =
     2
```

[2] Esta função devolve uma matriz quadrada, *n* x *n*, formada pelos números inteiros de 1 a n^2, com a particularidade de que a soma dos elementos de cada linha, de cada coluna e das diagonais é igual.

```
    11
     7
    14
```

É, inclusive, possível discriminar que linhas e/ou colunas se pretendem aceder:

```
>> A([1 4], [2 3])   % obtém os elementos comuns às linhas 1 e 4 e às colunas 2 e 3
ans =
     2     3
    14    15
```

8. Ficheiros *script*

Um ficheiro *script* não é mais do que um ficheiro de texto, de extensão `.m`, que contém uma sequência de funções e comandos. Tem como objetivo permitir executar um conjunto de instruções de uma só vez sem ter que as introduzir uma a uma na janela de comando. Para tal, basta apenas invocar o nome do ficheiro na linha de comando e executar.

As instruções executadas podem operar sobre os dados existentes no *workspace*, pelo que um *script* tem acesso a todas as variáveis lá existentes. De igual modo, qualquer variável criada durante a execução de um *script* passará a constar no *workspace* e persistirá após a conclusão do mesmo (desde que não tenha sido executada nenhuma instrução para apagar estas variáveis).

Os ficheiros *script* podem ser editados num qualquer editor de ficheiros de texto (notepad, notepad++, Brackets…) ou, mais comummente, através do editor do MATLAB, que pode ser invocado na linha de comando através do comando `edit`:

```
>> edit
```

Depois de editado, o *script* pode ser guardado. Para o nome do ficheiro *script*, pode ser usada qualquer combinação de carateres alfanuméricos e *underscore*, sendo imperativo que o **nome do *script* comece com uma letra**.

Também pode invocar a instrução `edit` passando o nome do ficheiro:

```
>> edit teste.m
```

Neste caso, se o ficheiro `teste.m` existir, este é aberto no editor. Se não existir, surgirá uma caixa de diálogo a referir este facto e a perguntar se deseja criar um ficheiro com este nome.

Finalmente, para executar o *script*, basta escrever o nome do mesmo na linha de comando:

```
>> teste
```

De ressalvar que, quando invocado o *script* através da linha de comando, este deve encontrar-se na pasta atual de trabalho, indicada na barra de caminho (*path*), localizada por cima da janela de comando (Figura 1). Caso o *script* não se encontre na pasta atual de trabalho, é devolvido um erro. Por exemplo, na Figura 2, o *script* `meu_script` não se encontra na pasta atual de trabalho, `d:\matlab`. Ao tentar executar este *script*, é devolvida a mensagem de erro *"Unrecognized function or variable 'meu_script'"*.

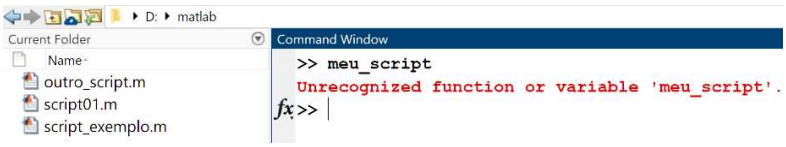

Figura 2 – Erro ao invocar um *script* não presente na pasta de trabalho.

Pode, naturalmente, alterar a pasta atual de trabalho para àquela onde se encontra o script, definindo o novo caminho na barra de caminho. Pode também executar um *script* que não se encontre na pasta de atual de trabalho, adicionando o caminho correspondente através do botão **Set Path**, disponível na barra de ferramentas do MATLAB (Figura 3).

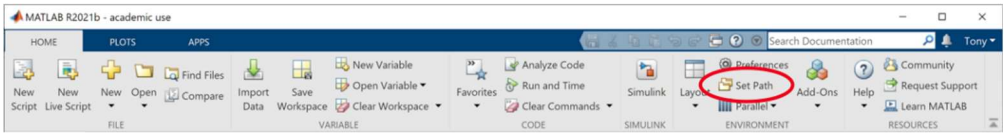

Figura 3 – Localização do botão **Set Path**.

Outra alternativa para executar o *script*, consiste em utilizar o botão **Run** disponível na barra de ferramentas do editor (Figura 4). Neste caso, se o *script* não estiver na pasta atual de trabalho, ou se não estiver contido em nenhum dos caminhos (*paths*) definidos no MATLAB, é então aberta automaticamente uma janela onde é permitido ao utilizador quer mudar de pasta, quer adicionar o caminho onde se encontra o ficheiro, entre outras opções (Figura 5).

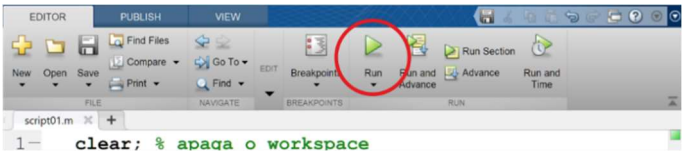

Figura 4 – Localização do botão **Run** no editor do MATLAB.

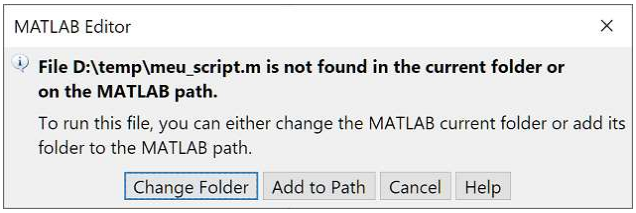

Figura 5 – Caixa de diálogo aberta quando se tenta executar um *script* a partir do editor, estando o *script* numa pasta diferente da pasta atual do MATLAB.

9. Atividades

Para a realização das seguintes atividades, **tenha presente a tabela no final do enunciado**, com as funções de MATLAB necessárias para a realização dos exercícios propostos. Para informações mais detalhadas, pode recorrer ao *help* do MATLAB.

9.1 Atividade 1 – Criação de um *script*

Nesta primeira atividade, pretende-se apenas que os alunos criem um pequeno *script*, com o objetivo de se familiarizarem com a criação e execução deste tipo de ficheiros em ambiente MATLAB.

a) Crie um *scritpt* com nome **`atividade_01.m`**.

```
>> edit atividade_01.m
```

O MATLAB irá abrir uma janela de diálogo a informar que o ficheiro não existe (caso não o tenha criado anteriormente). Clique *Yes* para continuar.

b) Interprete as linhas de comando abaixo indicadas e edite o *script* de forma a incluí-las:

```
clear; % apaga o workspace
clc;   % apaga a janela de comando
clf;   % apaga o conteúdo do gráfico
```

```
% Gera um vetor-coluna, A, com 10 valores aleatórios
% segundo uma distribuição normal de média 2
% e desvio-padrão 1
m = 2;
s = 1;
A = m + s*randn(10, 1)

% gráf. distribuição de frequências com 5 classes
histogram(A, 5);

media = mean(A); % cálculo da média
desvio = std(A); % cálculo do desvio padrão

disp(' '); % força uma linha em branco na janela de comandos
disp('A média dos valores do vetor A é');
disp(media);
disp(' ');
disp('O desvio-padrão dos valores do vetor A é');
disp(desvio);
shg; % SHow Graph - traz o gráfico para 1.º plano
```

c) Grave e execute o *script*.

d) Edite agora o *script* de forma a gerar um vetor A com 5000 elementos. Grave e volte a executar o *script*. Tenha atenção à necessidade de colocar um ponto-e-vírgula ao criar a nova matriz A (porquê?).

e) Altere agora o script de forma a alterar o valor médio dos valores para cinco.

f) No gráfico de frequências (histograma), altere o número de classes para 25.

g) Verifique o conteúdo do *workspace* após a execução do *script*. Como pode constatar, todas as variáveis criadas durante a execução do *script* permaneceram no *workspace* após aquele ter terminado.

h) Guarde o *script* obtido.

9.2 Atividade 2

a) Crie um *scritpt* com nome **`atividade_02.m`** e edite-o de forma a responder às alíneas seguintes.

b) Crie duas variáveis, *x* e *y*, atribuindo a cada uma um valor real positivo diferente de zero à escolha.

c) Usando estas variáveis, escreva as instruções necessárias para obter o resultado das seguintes expressões. **Notar que as expressões estão representadas em notação matemática, não seguindo, por isso, a sintaxe do MATLAB!** No final do enunciado, está disponível uma lista reduzida com algumas das funções matemáticas mais comuns, e a respetiva sintaxe em MATLAB, com uma explicação resumida. Para mais detalhes, pesquise o termo "elementary math" na documentação de ajuda do MATLAB, acessível através do comando `doc`.

1. $f = \dfrac{2\sin(x)\cos(y)}{4}$
2. $g = 5x^{1/2}y^{1/3}$
3. $h = y \cdot e^{-2x}$

4. $k = \left| \dfrac{5\log_{10} x}{1 - \tan\left(\dfrac{y}{2}\right)} \right|$

d) Grave e execute o *script*. **Não termine as linhas com ponto-e-vírgula para que os resultados aparecem diretamente na janela de comando**.

9.3 Atividade 3

a) Crie um *scritpt* com nome **`atividade_03.m`** e edite-o de forma a responder às alíneas seguintes.

b) Crie um vetor **x** contendo valores entre 0 e 18, com incrementos de 3 unidades (0, 3, 6, 9, 12, 15 e 18) recorrendo ao operador dois-pontos (`:`)[3].

c) Recorrendo à função **`length`**, determine o número de elementos do vetor **x**. Guarde o resultado na variável **`comp`**.

d) Adicione o valor 2 a cada elemento de **x** e guarde numa variável **`v1`**.

e) Calcule a raiz quadrada de cada elemento de **x** e guarde numa variável **`v2`**. *Consulte a lista de funções no final do enunciado.*

f) Calcule o quadrado de cada elemento de **x** e guarde numa variável **`v3`**.

g) Guarde numa variável **`v4`** apenas os valores dos elementos de índice ímpar de **x** (não se esqueça de que os índices das matrizes e dos vetores começam em 1).

h) Adicione o valor 5 apenas aos elementos de índice par de **x** e guarde numa variável **`v5`** (eventualmente, poderá precisar de mais do que uma linha de instrução).

i) Grave e execute o *script*.

9.4 Atividade 4

a) Crie um *scritpt* com nome **`atividade_04.m`** e edite-o de forma a responder às alíneas seguintes.

b) Defina o vetor **`t = 0:0.5:3`**.

c) Escreva as expressões em MATLAB que permitam calcular corretamente as seguintes expressões, para cada um dos valores de **`t`** (cada um dos resultados é um vetor e, mais uma vez, as expressões estão representadas em notação matemática). Guarde o resultado de cada expressão numa variável. *Consulte a lista de funções no final do enunciado.*

1. $\ln\left(1 + 3t + t^2\right)$
2. $e^{-t}\left(1 + \cos 2t\right)$
3. $\dfrac{t+1}{(t+2)^2}$

d) Grave e execute o *script*.

[3] Para procurar ajuda sobre o operador dois-pontos, pode recorrer aos comandos **`>> help colon`** ou **`>> doc colon`**.

9.5 Atividade 5

a) Crie um *scritpt* com nome **`atividade_05.m`** e edite-o de forma a responder às alíneas seguintes.

b) Defina a matriz **`A`** como sendo:

$$A = \begin{bmatrix} 3 & 2 & 2 & 8 & 8 \\ 1 & 4 & 9 & 6 & 10 \\ 2 & 1 & 5 & 4 & 6 \\ 5 & 8 & 6 & 10 & 4 \end{bmatrix}$$

c) Com recurso à função **`randi`**, crie ainda uma matriz **`B`**, de dimensões 4x5, com valores aleatórios inteiros entre −5 e 5.

d) Recorrendo ao comando **`size`**, obtenha o número de linhas e de colunas da matriz **`A`**, e guarde num vetor **`d`**. Como faria para guardar o valor do número das linhas e das colunas de **`A`** em variáveis distintas?

e) Atribua a um vetor **`x`** a segunda linha de **`A`**.

f) Atribua a uma matriz **`C`** as colunas de índice ímpar da matriz **`A`**.

g) Atribua a uma matriz **`sub_A`** a submatriz [2 2 8; 1 5 4] obtida a partir de **`A`**.

h) Recorra à função **`sum`** para guardar na variável **`soma_col`** a soma de cada uma das colunas de **`A`**. Note que a variável **`soma_col`** resultante é um vetor linha.

i) Novamente com a função **`sum`**, guarde na variável **`soma_lin`** a soma de cada uma das linhas de **`A`**. Tire proveito da sintaxe **`sum(A, dim)`** para realizar esta operação (consulte o *help* do MATLAB). Note que a variável **`soma_lin`** resultante é um vetor coluna.

j) Guarde na variável **`S`** a soma das matrizes **`A`** e **`B`**.

k) Guarde na variável **`M`** o produto elemento a elemento (*element-wise*) de **`A`** e **`B`** (por que não pode multiplicar matricialmente **`A`** e **`B`**?).

Algumas funções

Função	Descrição
`abs(A)`	Devolve o valor absoluto de cada elemento de `A`. Se a é um número complexo, devolve o seu módulo.
`clc`	Apaga a janela de comando
`clear`	Elimina todas as variáveis do *workspace*
`clear a, b, c`	Elimina as variáveis `a`, `b` e `c` do *workspace*
`cos(X)`	Cosseno de cada elemento de *X*, com *X* definido em radianos
`det(A)`	Determinante de *A*
`edit fich.m`	Abre o ficheiro `fich.m` no editor do MATLAB. Se o ficheiro não existe, é perguntado ao utilizador se o deseja criar. Invocar apenas o comando edit, abre o editor pronto a editar um ficheiro sem nome
`exp(A)`	Retorna o exponencial de cada elemento de *A*
`inv(A)`	Inversa de uma matriz *A*
`histogram(X, nbins)`	Gera um gráfico de distribuição de frequências dos valores de *X* ao longo de *nbins* classes. Se *nbins* for omitido, o número de classes é determinado automaticamente.
`length(v)`	Devolve o número de elementos no vetor *v*
`linspace(a, b, n)`	Gera um vetor de valores linearmente espaçados, gerando `n` pontos entre `a` e `b`
`log(X)`	Determina o logaritmo natural de cada elemento de *X*.
`log10(X)`	Determina o logaritmo de base 10 de cada elemento de *X*.
`magic(n)`	Produz uma matriz *n* x *n* a partir dos valores inteiros 1 a n^2, sendo igual a soma das colunas, das linhas e das diagonais
`max(A)`	Se *A* for um vetor, `max(A)` devolve o valor máximo contido no vetor. Se *A* for uma matrix, devolve um vetor contendo o valor máximo de cada coluna de *A*.
`max(A, B)`	Devolve uma matriz cujos elementos correspondem aos valores máximos entre *A* e *B*.
`rand(n, m)`	Cria uma matriz *n* x *m* de valores aleatórios entre 0 e 1. Se o parâmetro *m* for omitido, será gerada uma matriz quadrada *n* x *n.*
`randn(n, m)`	Cria uma matriz *n* x *m* de valores aleatórios de média nula e desvio-padrão 1. Se o parâmetro *m* for omitido, será gerada uma matriz quadrada *n* x *n.*
`randi(a, n, m)`	Cria uma matriz *n* x *m* de valores aleatórios inteiros entre 1 e *a*. Se *a* = [*nmin*, *nmax*], é gerada uma matriz *n* x *m* de valores aleatórios inteiros entre *nmin* e *nmax.* Se o parâmetro *m* for omitido, será gerada uma matriz quadrada *n* x *n.*
`sin(X)`	Seno de cada elemento de *X*, com *X* definido em radianos
`size(A)`	Devolve um vetor linha [*m n*] com o número de linhas (*m*) e número de colunas (*n*) de *A*
`sqrt(A)`	Calcula a raiz quadrada de cada elemento de *A*
`sum(A)`	Se *A* for um vetor, devolve a soma de todos os seus elementos. Se *A* for uma matriz, o resultado é um vetor-linha em que cada elemento corresponde à soma dos elementos de cada coluna de *A*
`tan(X)`	Tangente de cada elemento de *X*, com *X* definido em radianos
`who`	Apresenta uma lista de todas as variáveis no *workspace*
`whos`	Apresenta uma lista de todas as variáveis no *workspace*, incluindo informação sobre a sua classe (*int*, *float*, *bool*,...) e o seu tamanho em bytes