Computação e Programação

Jaime Ramos e Francisco Miguel Dionísio

Introdução ao sistema MATLAB

1 Introdução

Esta aula prática tem como objectivo a apresentação e familiarização com o sistema MATLAB e algumas das suas capacidades. Começamos por ver o MATLAB como uma máquina de calcular gráfica com inúmeras funcionalidades.

O espaço de trabalho do MATLAB divide-se em várias janelas. Para já concentramo-nos na janela de comandos (*Command Window*). É através desta janela que se processa maior parte da interacção com o sistema. Em particular, é através desta janela que introduzimos as expressões a avaliar (*input*) e visualizamos os resultados (*output*).

As expressões são introduzidas na janela de comandos na linha assinalada por `>>` e avaliadas premindo *Enter*.

2 Primeiros exemplos

Inicie a aplicação MATLAB.

Operações aritméticas

Começamos por avaliar algumas expressões muito simples, envolvendo operações aritméticas. A primeira é a expressão `1+1`. Para tal, introduz-se na janela de comandos a seguinte instrução:

```
>> 1+1
```

e carrega-se em *Enter*. Observe o resultado. De seguida, avalie as seguintes expressões:

```
45+2*3

(45+2*3)

2^3*2

2^(3*2)
```

Existe a possibilidade de guardar valores em *variáveis*. A *atribuição* de um valor a uma variável é feita através do comando `=`. Por exemplo, se se pretender atribuir o valor 4 à variável `a` basta introduzir na janela de comandos a seguinte instrução:

```
>> a=4
```

A partir de agora a variável `a` pode ser usada noutra qualquer expressão, sendo o seu valor (enquanto não for alterado) 4. Por exemplo, a avaliação da expressão:

```
>> a+2
```

resulta do valor 6. Introduza agora a seguinte sequência de comandos:

```
>> a=4
>> b=a+5
>> a=a+1
```

Observe o valor de `a`. Qual será o valor de `b`?

Help e lookfor

Para obter ajuda acerca da utilização de uma determinada função podemos recorrer ao comando `help`. Para tal basta escrever na janela comandos o comando `help` seguido do nome da função que pretende consultar. Por exemplo, se pretendermos obter informação acerca da função `sin` escrevemos na janela de comandos

```
>> help sin
```

Quando se pretende encontrar uma função para resolver um problema, mas não se sabe se existe uma função adequada em MATLAB, pode recorrer-se ao comando `lookfor` que pesquisa na descrição da biblioteca do comando `help` todas as funções cuja descrição usa a palavra que se procura. Por exemplo, podemos procurar todas as funções relacionadas com a palavra `inverse` escrevendo na janela de comandos a seguinte instrução:

```
>> lookfor inverse
```

Funções trignométricas

O MATLAB dispõe de uma enorme biblioteca de funções pré-definidas. Entre elas, encontram-se as funções trigonométricas, das quais se destacam: `sin`, `cos`, `tan`, `cot`, e as suas inversas `asin`, `acos`, `atan`, `acot`. Todas estas funções têm argumentos em radianos. Por exemplo

```
>> sin(pi/2)
>> tan(pi/4)
>> acot(1)/pi
```

Existem também funções com argumentos em graus: `sind`, `cosd`, `tand`, `cotd`, e as suas inversas `asind`, `acosd`, `atand`, `acotd`. Repetindo os exemplos anteriores:

```
>> sind(90)
>> tand(45)
>> acotd(1)
```

3 Operações sobre vectores e matrizes

Uma matriz é uma estrutura de dados bidimensional que permite armazenar números. Por exemplo, a matriz

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

é uma matriz com 2 linhas e 3 colunas. Para criar esta matriz em MATLAB basta introduzir na janela de comandos a instrução seguinte:

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

A vírgula (ou o espaço em branco) servem para separar os elementos dentro da mesma linha, o ponto e vírgula serve para separar as diferentes linhas. Para consultar o valor da variável `A` introduz-se na janela de comandos a instrução:

```
>> A
```

Vectores linha e vectores coluna são casos particulares de matrizes. O vector linha

$$V = [1, 2, 3, 4]$$

é uma matriz com 1 linha e 4 colunas que pode ser criado introduzindo na janela de comandos

```
>> V=[1,2,3,4]
```

Já o vector coluna

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

é uma matriz com 4 linhas e 1 coluna que pode ser criado introduzindo na linha de comandos

```
>> U=[1;2;3;4]
```

O elemento na linha i coluna j de uma matriz A é denotado pela expressão `A(i,j)`. Por exemplo, no caso da matriz anterior, se pretenderemos saber qual o elemento que se encontra na 1^a linha e 2^a coluna basta introduzir na janela de comandos:

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

No caso dos vectores, basta referir a sua posição. Por exemplo, o elemento na segunda posição do vector V é obtido através do comando:

```
>> V(2)
```

As operações aritméticas usuais estão disponíveis em MATLAB. Por exemplo, para somar

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

basta introduzir na linha de comandos

```
>> [1,2,3;4,5,6] + [6,5,4;3,2,1]
```

ou, alternativamente,

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

(Note que deve introduzir cada uma das linhas anteriores de cada vez, premindo *Enter* após cada comando.)

Se, por outro lado, se pretender multiplicar a matriz A definida acima por um escalar (por exemplo 3), basta introduzir na linha de comandos:

```
>> 3*A
```

Grande parte das funções $\mathbb{R}$ para $\mathbb{R}$ podem ser aplicadas a matrizes sendo o resultado uma matriz de igual dimensão que resultada da aplicação da função a cada um dos elementos da matriz original. Por exemplo, considere-se a matriz

$$B = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

que pode ser definida em MATLAB através do comando

```
>> B=[1,2,3,4,5,6,7,8,9,10];
```

Podemos em seguida determinar quais destes números são primos aplicando a função `isprime` a `B`

```
>> isprime(B)
```

O resultado será uma matriz (vector) com o valor `1` nas posições onde ocorrem números primos e `0` nas restantes posições. Confirme o resultado.

Exercício Defina a matriz

$$C = \begin{bmatrix} \pi & \pi/2 \\ \pi/4 & \pi/6 \end{bmatrix}$$

e, em seguida, calcule o resultado de aplicar a função *seno* (`sin`) a esta matriz.

Operações de construção de matrizes

Existem algumas operações que permitem a construção expedita de algumas matrizes. A função `zeros` serve para construir uma matriz constituída exclusivamente por zeros. Pode ser usada com uma argumento ou com dois argumentos. No caso de ser usada apenas com um argumento define uma matriz quadrada de dimensão igual ao argumento. Por exemplo, o comando

```
>> C=zeros(3)
```

define uma matriz de zeros de dimensão 3 que fica atribuída à na variável `C`. A função pode também ser usada com dois argumentos em que o primeiro especifica o número de linhas da matriz e o segundo especifica o número de colunas. Por exemplo, o comando

```
>> D=zeros(2,4)
```

define uma matriz de zeros com 2 linhas e 4 colunas que fica armazenada na variável `D`.

A função `ones` comporta-se exactamente como a função `zeros` mas, tal como o nome indica, utiliza-se para definir matrizes constituídas exclusivamente por uns. Por exemplo, a matriz

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

é definida pelo comando

```
>> ones(3,4)
```

Finalmente, a função `eye` utiliza-se para definir uma matriz diagonal. Os argumentos da função funcionam com na função `zeros`. Por exemplo, a matriz

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

é definida pelo comando

```
>> eye(4)
```

Exercício Defina as matrizes seguintes recorrendo às funções apresentadas acima:

1. $W = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$
2. $X = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$
3. $Y = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 5 \end{bmatrix}$
4. $Z = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Existe em MATLAB um operador que permite gerar sequências de números de uma forma bastante simples: o operador `:`. Este operador tem duas formas possíveis:

```
a:b
a:i:b
```

em que `a` é o valor inicial da sequência, `b` é o valor final da sequência e `i` é o incremento entre valores da sequência (quando este valor não é indicado, o valor por omissão é `1`). Assim, por exemplo o vector

$$V = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$$

pode ser gerado simplesmente a partir do comando

```
>> V=1:15
```

enquanto o vector

$$U = [1, 3, 5, 7, 9, 11, 13, 15]$$

pode ser gerado a partir do comando

```
>> U=1:2:15
```

Este operador pode também ser usado para referenciar certas posições de uma matriz (ou de um vector). Por exemplo, o comando

```
>> V(1:5)
```

devolve o subvector de V constituído pelas primeiras 5 posições, enquanto o comando

```
>> V(1:2:15)
```

devolve o subvector de V constituído pelas posições ímpares. Verifique que os valores obtidos são, de facto, os esperados.

Podemos ainda usar este operador para extrair, por exemplo, linhas de uma matriz. Recorde-se a matriz

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

Nestas condições, o comando

```
>> A(1,:)
```

refere-se à primeira linha da matriz (o operador `:` sem valor inicial nem final serve para referência a linha completa).

Exercício Recorrendo às funções anteriores defina as matrizes seguintes

1. $Z = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$
2. $V = [0, 3, 6, 9, 12, 15, 18, 21]$

Polinómios

O sistema MATLAB dispõe de uma colecção de funções para manipular polinómios. A representação de um polinómio faz-se através um vector (linha) em que os coeficientes do polinómio surgem por ordenados do mais significativo para o menos significativo. Por exemplo, o polinómio

$$x^2 - 2x - 3$$

é representado pelo vector

```
>> P=[1,-2,-3]
```

Para encontrar as raízes do polinómio, basta introduzir o comando

```
>> roots(P)
```

Reciprocamente, para obter um polinómio a partir das suas raízes utiliza-se o comando `poly`. Por exemplo, para determinar o polinómio cujas raízes são 3 e -1 basta introduzir os comandos

```
>> r=[3,-1]
>> poly(r)
```

Para calcular o valor de um polinómio num ponto ou num conjunto de pontos recorre-se ao comando `polyval`. Relativamente ao polinómio anterior, podemos determinar o seu valor no ponto 4 introduzindo o comando

```
>> polyval(P,4)
```

4 Definição de funções

Para além da extensa biblioteca de funções o MATLAB permite a definição de funções adicionais. Uma das formas de definir uma *função* é criar um ficheiro (com extensão `.m`) contendo a definição da função pretendida. Para criar um destes ficheiros basta seleccionar na barra de ferramentas do MATLAB a opção *File* → *New* → *M-file*; em alternativa, carregar em *Ctrl-N* produz o mesmo efeito. Um vez aberto o editor de ficheiros, escreve-se o código da função e grava-se.

Como exemplo, crie um novo ficheiro `.m` contendo a definição da seguinte função:

```
function y=exemplo(x)
% a minha primeira funcao
y=x^2-2*x+3;
```

Grave o ficheiro. Por omissão, o MATLAB assume para nome do ficheiro o nome da função.

A partir deste momento a função `f` pode ser usada como qualquer outra função MATLAB. Por exemplo, para calcular o valor da função no ponto 4 basta introduzir na janela de comandos:

```
>> exemplo(4)
```

Se obtiver uma mensagem de erro poderá ser porque se esqueceu de acrescentar à lista *path* a pasta onde está a trabalhar. Por cima da janela de comandos encontra-se a indicação de *Current Directory*. Esta é a pasta de trabalho. O MATLAB só procura ficheiros em directorias que tenham sido previamente adicionadas à lista *path*. Por isso, é importante acrescentar a directoria onde estamos trabalhar a esta lista. Tal é possível seleccionando *File* → *Set Path* na barra de ferramentas.

Exercício Defina as seguintes funções em MATLAB e experimenta-as:

1. $segundo(x) = (3x - 2)/(x + 1)$
2. $soma\_nat(n) = \sum_{i=1}^{n} i$.

Para a resolução do exercício 2, recorde a função definida nas aulas teóricas para calcular a soma dos primeiros n números naturais:

```
function r=soma_nat(n)
i=1;
r=0;
while (i<=n)
    r=r+i;
    i=i+1;
end
```

5 Gráficos

Vamos ver agora como podemos utilizar o MATLAB para desenhar o gráfico de uma função. Para desenhar o gráfico precisamos de dois vectores, uma para as abcissas e outro para as ordenadas. No final invocamos o comando `plot`.

O vector das *abcissas* é construído usando a operação `:` definida atrás. O vector das *ordenadas* é obtido aplicado a função de que se pretende desenhar o gráfico ao vector das abcissas. Para desenhar o gráfico da função *seno* no intervalo $[0, \pi]$ começamos por definir o vector das abcissas nesse intervalo com espaçamento entre pontos de 0.1:

```
>> x=0 : 0.1 : 2*pi;
```

Em seguida, calculamos o vector das ordenadas aplicando a função `sin` ao vector `x`:

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

Por fim, desenhamos o gráfico recorrendo ao comando `plot`:

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

Exercício Desenhe os seguintes gráficos:

1. função: *tangente*, intervalo: $[-\pi, \pi]$;
2. função: *logaritmo*, intervalo: $[0, 10]$. Neste caso comece por usar um espaçamento de 1 e depois repita o exercício com um espaçamento de 0.1.

O MATLAB permite também sobrepor gráficos de funções. Por exemplo, se pretendermos sobrepor o gráfico da função *seno* ao gráfico da função *coseno* no intervalo $[0, 2\pi]$ podemos proceder da seguinte forma:

```
>> x1=0:0.1:2*pi;
>> y1=sin(x1);
>> x2=0:0.1:2*pi;
>> y2=cos(x2);
>> plot(x1,y1,x2,y2)
```

Por fim, ilustra-se a visualização de gráficos de funções de duas variáveis $f : \mathbb{R}^2 \to \mathbb{R}$. Para tal, é necessário recorrer à função MATLAB `meshgrid` que gera duas matrizes, X e Y com os pontos x e y onde se pretende calcular cada valor $f(x, y)$. Por exemplo, para desenhar o gráfico da função f tal que

$$f(x, y) = \frac{sen(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$$

no intervalo $[-10, 10]$ utiliza-se a seguinte sequência de comandos:

```
>> [X,Y]=meshgrid(-10:0.5:10);
>> R=sqrt(X.^2+Y.^2)+eps;
>> Z=sin(R)./R;
>> mesh(X,Y,Z)
```

Em alternativa, poderá utilizar o comando `surf` em vez do comando `mesh` para gerar uma superfície plana em vez de uma grelha. Experimente

```
>> surf(X,Y,Z)
```