Aplicações Informáticas para Engenharia

Introdução ao Matlab

Sérgio Manuel Ângelo da Cruz

2007

Introdução

- **Ferramentas Informáticas de Apoio à Engenharia:**
  - **Matlab** (**Mat**rix **Lab**oratory) e **Simulink**
  - **Labview**
  - Mathematica e Maple
  - Mathcad
  - Maxwell 2D / 3D, Ansys, Flux 2D
  - ....e muitas outras.

Introdução

- **Porquê o estudo do Matlab/Simulink e Labview?**
  - Uso intensivo destas ferramentas quer nas universidades quer na indústria
  - O Matlab, com as suas caixas de ferramentas específicas (toolboxes), é uma ferramenta usada em todas as áreas da Engenharia Electrotécnica:
    - Controlo, Processamento de Sinal, Processamento de Voz e Imagem, Energia (Linhas de Transmissão, Máquinas Eléctricas, Electrónica de Potência, etc.), Redes Neuronais, Optimização de Sistemas, etc.

Matlab - Introdução

- Ferramenta de cálculo científico, vocacionada para o cálculo numérico

- Permite o desenvolvimento de programas ao nível da(o):
  - Análise numérica
  - Análise de dados
  - Cálculo matricial
  - Processamento de sinais
  - Construção de gráficos
  - etc.

Matlab - Introdução

- Nesta Unidade Curricular (UC) iremos apenas utilizar uma pequena percentagem das funcionalidades do Matlab

- Iremos abordar os conceitos básicos numa primeira fase e aprofundar os conhecimentos ao longo do semestre.

Matlab – Ambiente de Trabalho

- Quando o Matlab é iniciado, são visíveis três janelas distintas:
  - A janela de comando (onde aparece o símbolo “>>”)
  - Uma outra janela, com os submenus da directoria actual e do ambiente de trabalho. Aqui é exibida uma listagem dos ficheiros da directoria corrente bem como o ambiente de trabalho (*workspace*), onde consta uma listagem das variáveis que vão sendo geradas/introduzidas pelo matlab/utilizador
  - A janela do histórico (parte inferior do lado esquerdo do monitor, onde aparece o histórico dos comandos introduzidos na janela de comando.

Matlab – Operadores Aritméticos

Operador	Funcionalidade
^	Potenciação
/	Divisão à direita
\	Divisão à esquerda
*	Multiplicação
+	Adição
-	Subtracção

Nota: o Matlab respeita as prioridades habituais destes operadores.

Matlab – Operadores Relacionais

Operador	Funcionalidade
<	Menor
<=	Menor ou igual
>	Maior
>=	Maior ou igual
==	Igual
~=	Diferente

Matlab – Operadores Lógicos

Operador	Funcionalidade
~	NOT (negação)
&	AND (conjunção)
|	OR (ou inclusivo)
xor	ou exclusivo

Nota: Os operadores lógicos têm uma prioridade inferior à dos operadores aritméticos e relacionais.

- Exemplo:

```
>> 1==2 & 4>=3
ans =
     0
```

Tipos de dados no Matlab

- char (usar ‘ e não “ nas definições!!)
- numeric
  - single (4 bytes)
  - double (8 bytes, tipo de dado por defeito)
  - uint8 (inteiro de 1 byte, sem sinal)
  - int64 (inteiro de 8 bytes, com sinal)
  - ... e muitos outros ....
- struct
- cell
- Informação adicional: usar o comando **help datatypes**

Declarações, constantes e variáveis

- Definição de variáveis e atribuição de valores às mesmas:
  - **a=2** define uma variável *a* e atribui-lhe o valor 2
  - **b='isto é uma aula'**; define a variável *b* do tipo char
  - **c=3, d=23e-3** define e atribui valores a *c* e *d*
  - **d=int16(23)** define *d* como sendo do tipo *int16* e inicializa-o com o valor 23

- A colocação de **;** no fim da linha de comando instrui o matlab a não dar nenhuma resposta:
  - Ver a diferença entre introduzir **a=2** e **a=2;**

- A qualquer altura pode ver as variáveis que já foram criadas, com os comandos **who** e **whos**

- ATENÇÃO: o matlab faz distinção entre maiúsculas e minúsculas, por isso *y* e *Y* são duas variáveis distintas!!

Vectores e Matrizes

- Definição de um vector linha X:
  - **x=[1 20 45 1e-2 sin(pi)]** define um vector linha com 5 elementos
  - Em alternativa pode-se definir como **x=[1,20,45,1e-2,sin(pi)]**
  - Outras formas de definir vectores:

```
>>x=1:3
x =
    1    2    3
```

ou

```
>>x=2:3:11
x =
    2    5    8   11
```

Vector=valor mínimo:passo:valor máximo

Vectores e Matrizes

- A função **linspace** permite criar vectores de elementos com igual espaçamento entre si:

  Vector=linspace(valor mínimo, valor máximo, número de elementos)

```
>>x=linspace(2,8,4)
x =
    2    4    6    8

>> x=linspace(2,7,4)
x =
   2.0000   3.6667   5.3333   7.0000
```

Vectores e Matrizes

- Definição de uma matriz A, de dimensões 3×3:

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

- Em alternativa, pode-se definir a matriz A linha a linha, premindo **<enter>** após a introdução de cada linha

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

A =
     1     3     6
     4     2     6
     6     8     9
```

Vectores e Matrizes

- Para transpor um vector ou matriz, usa-se o operador ‘:

```
>> y=[1 34 56]’
y =
     1
    34
    56
```

- Pode-se aceder aos elementos das matrizes através do número da linha e coluna onde tais elementos estão localizados:
  - A(x,y) devolve o elemento da matriz A situado na linha x e coluna y. Por exemplo A(1,2)=3
  - Pode-se também extrair mais do que um elemento simultaneamente. Por exemplo A(2,:) devolve toda a segunda linha da matriz A

Vectores e Matrizes

- Pode-se ainda obter submatrizes usando comandos, para além dos índices. Por exemplo:

```
>> A(3,2:end)
ans =
     8     9
```

  - Neste caso, o comando **end** indica que se devem extrair os elementos da matriz, desde a segunda até à última coluna
- Outros exemplos:

```
>> B(7, 1:5);        % Lê as colunas 1-5 (de 1 a 5) na linha 7
>> B(4:2:8, 1:5);    % Lê as colunas 1-5 nas linhas 4, 6 e 8
>> B(:, 1:5);        % Lê as colunas 1-5 em todas as linhas
```

- **Notas importantes:** o matlab armazena a última resposta na variável **ans** (de *answer*); uma sequência de números pode ser gerada usando a sintaxe *início:passo:fim*

Vectores e Matrizes

- Para se conhecerem as dimensões de uma matriz ou vector, pode-se usar o comando **size**:

```
>> size(x)
ans =
     1     4
```

- Para aceder individualmente ao número de linhas e colunas, poder-se-á escrever

```
>> [linhas,colunas]=size(A);
```

  - Neste exemplo, *linhas* e *colunas* são variáveis que serão inicializadas com o número de linhas e colunas da matriz A

- No caso de vectores, pode-se ainda usar o comando **length** para obter o comprimento dos mesmos

Vectores e Matrizes

- Pode-se transformar uma matriz num vector coluna usando "**:**" da seguinte forma:

```
>> B=[1 34 45;4 5 6;100 200 300]
B =
     1    34    45
     4     5     6
   100   200   300
```

```
>> x=B(:)
x =
     1
     4
   100
    34
     5
   200
    45
     6
   300
```

Vectores e Matrizes

- Nas operações entre matrizes, utilizam-se os mesmos operadores aritméticos que entre escalares
- No entanto, pode-se preceder o operador aritmético de um “.”. Esse facto indica ao matlab que a operação deve ser efectuada elemento a elemento. Um exemplo:

```
>> A=[1 2;3 4]; B=[3 5;1 0.5];
>> C=A*B
C =                 % Produto habitual de duas matrizes
     5     6
    13    17
>> C=A.*B       % Neste caso C(i,j)=A(i,j)*B(i,j)
C =
     3    10
     3     2
```

Vectores e Matrizes

- Criação de matrizes especiais:

Comando	Funcionalidade
eye(m,n)	Cria uma matriz identidade de dimensões m×n
ones(m,n)	Cria uma matriz composta por elementos unitários
zeros(m,n)	Cria uma matriz composta por elementos nulos
rand(m,n) e randn(m,n)	Cria uma matriz composta por elementos aleatórios (entre 0 e 1 no caso de rand)
diag(X)	Cria uma matriz diagonal (se X é vector) ou extrai elementos da diagonal (se X é matriz)

- No caso de se fornecer apenas um argumento às funções anteriores, são criadas matrizes quadradas

Funções Sobre Escalares

Funções usadas frequentemente	
sin	sqrt
asin	abs
cos	ceil
acos	round
exp	floor
log e log10	rem
tan	sign
atan	rats

Nota: no caso de funções trigonométricas, o argumento terá de ser fornecido em radianos

Funções Sobre Vectores e Matrizes

Funções mais usuais	
median	sort
max	min
prod	sum
all	any
mean	std

Funções Sobre Matrizes

Funções mais usuais	
eig	chol
svd	inv
lu	qr
hess	schur
expm	sqrtm
size	norm
cond	rank
triu	tril
poly	det

Formatação da Saída

- O comando **format** permite alterar a forma como o matlab apresenta os números no monitor

Comando	Formato da saída
format short	0.6667
format short e	6.6667e-001
format long	0.66666666666667
format long e	6.66666666666666e-001
format hex	3fe5555555555555
format rat	2/3
format bank	0.67

Utilidades Diversas

- A função **clear** permite eliminar parte ou todas as variáveis criadas anteriormente:

```
>> clear x     % elimina o vector x
>> clear A B   % elimina as matrizes A e B
>> clear       % elimina todas as variáveis não permanentes
>> clear all   % idêntico a clear mas elimina também funções, atalhos MEX, etc., etc.
```

- Existem variáveis que o matlab já traz definidas por defeito e que não podem ser apagadas (variáveis permanentes):
  - **pi** % número pi
  - **inf** % infinito
  - **NaN** % não numérico, por exemplo 0/0
  - **i e j** % número complexo i=j=sqrt(-1)
  - **realmin** % menor número real positivo que o matlab consegue representar
  - **realmax** % maior número real positivo que o matlab consegue representar

Utilidades Diversas

- Quando se sai do matlab, perdem-se todas as variáveis. O comando **save** permite gravar todas as variáveis no ficheiro matlab.mat (defeito)
- O comando **load** permite restaurar as variáveis a partir desse ficheiro
- Pode-se gravar apenas algumas variáveis. Um exemplo:

```
>>save backup_aula A B x    % grava as matrizes A, B e o vector x
>>clear                     % elimina todas as variáveis
>>load backup_aula          % restaura as variáveis gravadas
```

- O matlab pode gravar todos os comandos introduzidos na janela de comando. Para esse efeito usa-se o comando **diary**:

```
>>diary arquivo        % grava todos os comandos (excepto gráficos)
```

- Para suspender este comando faz-se **diary off** e para voltar a habilitá-lo faz-se **diary on**

- O comando **clc** permite limpar a janela de comando