Introdução às Técnicas Computacionais utilizando o MATLAB

1) Introdução

MATLAB® é um sistema que compreende uma linguagem de programação e um ambiente de cálculo que utiliza matrizes de dimensões livres como estrutura básica de dados, sendo cada dia mais utilizado para cálculos científicos e de engenharia. Por apresentar uma linguagem de alto nível para análise numérica permite que os códigos dos algoritmos sejam escritos de forma extremamente simples e compacta. O MATLAB também possui excelentes ferramentas gráficas e de visualização, extremamente fáceis de se implementar. É um produto comercial desenvolvido e distribuído pela MathWorks®. O nome MATLAB® é uma abreviação de MATrix LABoratory.

Para iniciar o MATLAB no ambiente Windows basta clicar em seu atalho geralmente disposto na área de trabalho do PC por ocasião da instalação, ou ainda no menu de programas do Windows.

Ao iniciar o sistema aparecem três janelas:

- *Command Window*: onde todos os comandos são dados;
- *Command History*: onde são registrados os últimos comandos dados;
- *Launch Pad*: apresenta todos os dispositivos instalados.

A janela de trabalho é a *Command Window*, e é nela onde as instruções de cálculo devem ser passadas. Esses comandos podem ser digitados linha a linha, ou através de *scripts* editados no editor MEDIT do MATLAB, ou em qualquer outro editor de texto. Estes *scripts* devem ser salvos com a extensão (*.m*) e dispostos no diretório de trabalho, estabelecido na barra de rolagem *Current Directory* na janela principal do sistema. Uma vez salvo o *script* com a seqüência de comandos, o cálculo é realizado aos se digitar o nome do arquivo (sem a extensão) no *prompt* da *Command Window*.

Uma interessante demonstração das capacidades do MATLAB pode ser visualizada ao se digitar o comando ***demo*** no *prompt* da *Command Window*. Uma visão geral das funções pré-definidas do sistema pode ser obtida ao se digitar o comando ***helpwin***. A finalidade e sintaxe de uma função específica podem ser verificadas digitando-se ***help*** e em seguida o nome da função.

2) Definindo Variáveis

Para se definir uma variável, basta realizar uma atribuição simples, e apertar *Enter*:

```
>> x=4

x =
     4
```

Após a atribuição, o sistema apresenta o valor da variável definida. Para se evitar esta "repetição", basta digitar "ponto-e-vírgula" após a atribuição. A mesma atribuição pode ser feita em um arquivo de texto, seguida de um *line-feed*. Note-se que não é necessário o pré-estabelecimento do tipo e das dimensões da variável.

Um vetor ou matriz pode ser rapidamente definido a partir de suas linhas:

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

A =
     1     2     3
     4     5     6
     7     8     9
```

A matriz pode ser definida elemento a elemento também, por atribuição simples:

```
>> A(1,1)=1;
```

O uso do "dois pontos" permite a rápida definição de um vetor a partir do estabelecimento de um valor inicial, um passo e um valor final:

```
>> v=1:0.5:3

v =
    1.0000    1.5000    2.0000    2.5000    3.0000
```

O mesmo artifício pode ser utilizado para se definir submatrizes:

```
>> B=A(1,:)
B =
     1     2     3
```

```
>> C=A(:,1)
C =
     1
     4
     7
>> D=A(2:3,1:2)
```

```
D =
     4     5
     7     8
```

Existem também funções que constroem matrizes ou vetores a partir do estabelecimento de alguns parâmetros. Algumas delas são:

eye(n)	matriz identidade n×n;
zeros(n)	matriz de zeros n×n;
ones(n)	matriz n×n com coeficientes unitários;
diag(A)	retorna a diagonal principal da matriz A como um vetor;
rand(n)	matriz n×n formada com valores aleatórios;
linspace(x_1,x_2,n)	gera n pontos entre x_1 e x_2.

Exemplo:

```
E=eye(3)

E =
     1     0     0
     0     1     0
     0     0     1
```

3) Operações Básicas

As operações abaixo são disponíveis no MATLAB, podendo ser utilizadas diretamente nas matrizes:

```
+    adição
-    subtraçào
*    multiplicação
^    potência
'    transposta
/    divisão simples (pela direita)
\    divisão pela esquerda
```

```
>> A=[1 2;3 4];B=[1 0;0 0];
>> A*B

ans =
     1     0
     3     0
```

```
>> A+B

ans =
     2     2
     3     4
```

A operações de divisão à esquerda e à direita devem ser utilizadas nas seguintes situações:

- Sistema de equações lineares $[A]\cdot\{x\}=\{b\}$

  `>> x=A\b`

```
>> b=[1 2]'

b =
     1
     2

>> x=A\b

x =
          0
     0.5000
```

  Se matriz A for superdeterminada (retangular, com número de linhas maior que o número de colunas), a resposta compreende a solução pelo método dos mínimos quadrados (pseudo-inversa).

- Sistema de equações lineares $\{x\}^t\cdot[A]=\{c\}^t$

  `>> x=c/A`

As operações `*`, `^`, `\`, e `/` podem ainda ser realizadas elemento a elemento, desde que se digite o ponto antes das mesmas:

```
>> [1,2,3,4].*[1,2,3,4]

ans =
     1     4     9    16

>> [1,2,3,4].^2

ans =
     1     4     9    16
```

4) Gráficos 2D

O MATLAB dispõe de recursos gráficos elaborados, entretanto, neste momento iremos nos ater à impressão gráfica elementar em 2D. A função mais utilizada é:

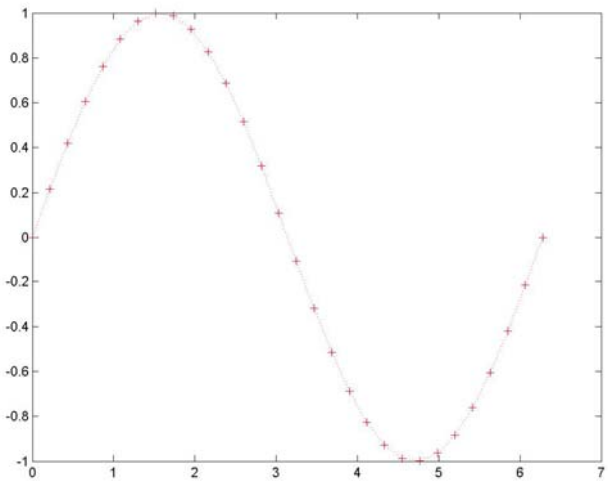
`plot` ex: `plot(y)` desenha uma série de valores armazenada em y;
`plot(x,y)` desenha o gráfico de uma função y=f(x);
`plot(x,y,s)` idem ao anterior, sendo s uma string com o formato do ponto:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Exemplo:

```
>> x=linspace(0,2*pi,30);
>> y=sin(x);
>> plot(x,y,'r+:')
```

Após o ultimo comando, é aberta uma janela “Figure No. 1” com o gráfico:



Para se incluir várias funções num mesmo gráfico, faz-se:

```
plot(x1,y1,s1,x2,y2,s2,x3,y3,s3)
```

Abaixo se apresentam algumas outras funções correlatas:

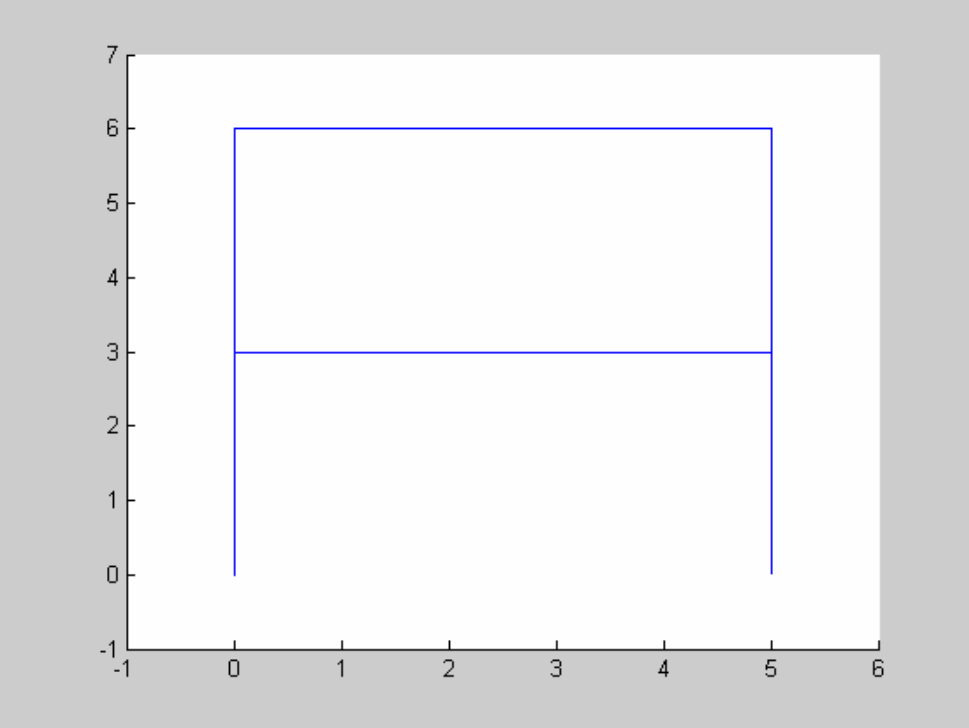
`grid`	acrescenta linhas de grid ao gráfico;
`xlabel ('texto')`	acrescenta rótulo ao eixo x;
`ylabel ('texto')`	acrescenta rótulo ao eixo y;
`title('texto')`	dá título ao gráfico;
`axis([xmin xmax ymin ymax])`	estabelece os limites do gráfico;
`axis equal`	estabelece a mesma escala em x e y;

Os gráficos podem ser exportados como imagens padrão bmp, jpg e outros. Outra função muito utilizada em gráficos 2D é a função `line`:

`line(X,Y)`	desenha uma linha contínua definida pelos pares ordenados formados pela relação X × Y ;

Por exemplo, para se desenhar um pórtico de dois pavimentos, faria-se:

```
>> clf
>> line([0,0,0,5,5,5],[0,3,6,6,3,0])
>> line([0,5],[3,3])
>> axis([-1,6,-1,7])
```



5) Polinômios

Um grande número de problemas de engenharia recaem em operações em polinômios. O MATLAB manipula polinômios com muita facilidade a partir da definição dos seus coeficientes em ordem decrescente de potência.

O polinômio $x^4 - 12x^3 + 25x + 116$ poderia então ser definido pelo vetor:

```
>> p=[1 -12 0 25 116]

p =
     1   -12     0    25   116
```

As funções pré-definidas mais importantes são:

`roots(p)`	calcula as raízes reais e imaginárias;
`poly(r)`	a partir das raízes, obtém os coeficientes do polinômio;
`y=polyval(p,x)`	calcula os valores y=p(x);
`polyder(p)`	obtém os coeficientes da função derivada do polinômio.

6) Séries

Dados dispostos segundo séries temporais podem ser facilmente analisados a partir das funções abaixo. Deve-se, entretanto, dispor dados diferentes em colunas.

`mean(X)`	fornece um vetor com a média das colunas da matriz X;
`max(X)`	fornece um vetor com os valores máximos das colunas da matriz X;
`min(X)`	fornece um vetor com os valores mínimos das colunas da matriz X;
`std(X)`	fornece um vetor com os valores de desvio padrão dos dados dispostos nas colunas da matriz X;
`corrcoef(v)`	fornece um vetor com os valores do coeficiente de correlação dos dados dispostos nas colunas da matriz X;

7) Entrada e saída de dados (*File I/O*)

Funções para abrir, carregar e salvar arquivos:

importdata	Load data from various types of files
load	Load all or specific data from MAT or ASCII file
open	Open files of various types using appropriate editor or program
save	Save all or specific data to MAT or ASCII file
uiimport	Open Import Wizard, the graphical user interface to import data
winopen	Open file in appropriate application (Windows only)

Funções de entrada e saída de arquivos texto:

csvread	Read numeric data from text file, using comma delimiter
csvwrite	Write numeric data to text file, using comma delimiter
dlmread	Read numeric data from text file, specifying your own delimiter
dlmwrite	Write numeric data to text file, specifying your own delimiter
textread	Read data from text file, write to multiple outputs
textscan	Read data from text file, convert and write to cell array

Funções de entrada e saída de baixo nível:

fclose	Close one or more open files
feof	Test for end-of-file
ferror	Query MATLAB about errors in file input or output
fgetl	Return next line of file as string without line terminator(s)
fgets	Return next line of file as string with line terminator(s)
fopen	Open file or obtain information about open files
fprintf	Write formatted data to file
fread	Read binary data from file
frewind	Rewind open file
fscanf	Read formatted data from file
fseek	Set file position indicator
ftell	Get file position indicator
fwrite	Write binary data to file

Funções de entrada e saída de dados em planilhas do Excel® (versão R13):

xlsfinfo	Determine if file contains Microsoft Excel (`.xls`) spreadsheet
xlsread	Read Microsoft Excel spreadsheet file (`.xls`)
xlswrite	Write Microsoft Excel spreadsheet file (`.xls`)

8) Expressões de Controle de Fluxo

Laço utilizando **for**:

```
for x=array
     comands
end
```

Laço utilizando **while**:

```
while expression
     comands
end
```

Estrutura **if-then-else**:

```
if expression
     commands evaluated if True
else
     commands evaluated if False
end
```

Relações das expressões:

```
<       less than
>       greater than
<=      less than or equal
>=      greater than or equal
==      equal
~=      not equal.
&       and
|       or
~       not.
```

9) *M-files*, Sub-rotinas e funções

Conforme já foi apresentado, os comandos do MATLAB podem ser escritos de forma seqüenciada em um arquivo texto, com extensão *.m*, salvos, e então executados a partir da digitação de seu nome (sem a extensão) no *prompt* da janela de comando. As variáveis declaradas ficam disponíveis na memória até que se finde a execução do MATLAB, ou se execute a função `clear`.

Outros arquivos *m-file* podem ser chamados dentro da execução de um desses *scripts*, os quais terão acesso direto a todas as variáveis disponibilizadas na memória, funcionando desta forma como sub-rotinas.

Podem ser definidos ainda arquivos do tipo função (*function m-file*). Eles se diferenciam dos demais por manipularem apenas as variáveis que lhes são passadas. Além disso, variáveis utilizadas internamente no cálculo da função não aparecem na memória da área de trabalho.

A primeira linha de um arquivo-função especifica seu nome (o mesmo do arquivo) e as variáveis de entrada e saída. A função deve por fim retornar os valores de suas variáveis de saída.

Exemplo:

```
function y=funcao(x)

y=x^3-11*x^2+38*x-40;
```

10) Resumo das Funções Matriciais mais utilizadas:

Category	Function	Description
Matrix analysis	norm	Matrix or vector norm
	normest	Estimate the matrix 2-norm
	rank	Matrix rank
	det	Determinant
	trace	Sum of diagonal elements
	null	Null space
	orth	Orthogonalization
	rref	Reduced row echelon form
	subspace	Angle between two subspaces
Linear equations	\ and /	Linear equation solution (slash)
	inv	Matrix inverse
	chol	Cholesky factorization
	cholinc	Incomplete Cholesky fact
	lu	LU factorization.
	luinc	Incomplete LU factorization
	qr	Orthogonal-triangular decomposition
	lsqnonneg	Nonnegative least-squares
	pinv	Pseudoinverse
	lscov	Least squares with known covariance
Eigenvalues and singular values	eig	Eigenvalues and eigenvectors
	svd	Singular value decomposition
	eigs	A few eigenvalues
	svds	A few singular values
	poly	Characteristic polynomial
	polyeig	Polynomial eigenvalue problem
	condeig	Condition number for eigenvalues
	hess	Hessenberg form
	qz	QZ factorization
	schur	Schur decomposition
Matrix functions	expm	Matrix exponential
	logm	Matrix logarithm
	sqrtm	Matrix square root
	funm	Evaluate general matrix function

http://www.mathworks.com/

11) Ferramentas de *Debugging*

12) Matemática Simbólica

13) Gráficos 3D

14) Programação orientada a eventos

15) Exercícios