PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Introdução ao Matlab
Parte I: Manipulação e visualização de dados

Introdução

O MatLab é um programa interativo para computação numérica e visualização de dados; é usado intensivamente na engenharia de controle para a análise e projeto. O MatLab contem uma coleção de ferramentas chamadas de *toolbox*. Cada *toolbox* possui muitas funções que expandem o ambiente do MatLab em áreas especificas tais como: processamento de sinais, sistemas de controle, comunicações, redes neurais artificiais, estatística, financeira, etc. O Simulink é um programa de simulação gráfica que acompanha o Matlab. Este programa possibilita a simulação de sistemas dinâmicos lineares e não-lineares em nível de diagrama de blocos, sendo empregado para análise e projeto de sistemas de controle.

Este roteiro será utilizado para acompanhar a primeira aula de laboratório de introdução ao MatLab. Nesta aula será utilizado um programa tutorial denominado "demat2" onde cada um dos itens apresentados aqui serão demonstrados.
Para utilizá-lo digite na linha de comando do matlab:
>> demat2(0,0)

O ambiente do MatLab não é gráfico e pode ser considerado como um editor de linha: digita-se uma linha de comando e no final da linha a tecla *enter* e o MatLab retorna com a informação processada ou uma mensagem de erro. O *prompt* >> indica que o MatLab está pronto para receber um comando. Uma linha sem o *prompt* indica o retorno do MatLab ao comando digitado em uma linha anterior.

Uma regra informal, mas útil é definir letras maiúsculas para variáveis vetoriais e matriciais e letras minúsculas para variáveis escalares (como a linguagem C o MatLab é sensível a letra maiúscula/minúscula: a ≠ A).

Vetores

Para começar vamos criar alguma coisa simples, como um vetor. Entre cada elemento do vetor (separado por um espaço) entre colchetes. Os operadores ([ ) e ( ]) criam *arrays* unidimensionais (vetores) e bidimensionais (matrizes). Por exemplo, para criar o vetor A, entre no MatLab com o seguinte comando de linha:

```
» A = [ 2  4  6  8  10  12  14 ]

A =
     2  4  6  8  10  12  14
```

Se tivéssemos digitado a seguinte linha, utilizando o operador (;):

```
» A = [ 2  4  6  8  10  12  14 ] ;
```

MatLab não retornaria nenhuma informação, mas a variável a armazenaria a informação digitada. Digitando:

```
» A
 A  =
       2  4  6  8  10  12  14
```

Existe uma outra maneira de criar um vetor utilizando o operador (:). Este método é geralmente criado para gerar um vetor de tempo. Na linha de comando, o primeiro elemento (depois do sinal =) é o valor inicial do vetor. O segundo elemento é o valor de incremento e o terceiro é o valor do último elemento.

```
» A  =  2 : 2 : 14
 A  =
       2  4  6  8  10  12  14
```

Cada elemento do vetor pode ser acessado individualmente, indexado pela sua posição dentro do vetor utilizando o operador ( ) . Por exemplo para acessar o terceiro elemento do vetor A :

```
» elemento = A( 3 )
elemento =
             6
```

MatLab retorna a variável ans quando na linha de comando não é definida uma variável para guardar a informação gerada pela linha de comando.

```
» A(3)
ans =
          6
```

Note que diferentemente da linguagem C o primeiro elemento do vetor é indexado com a posição 1 e não com a posição zero:

```
» A(0)
???  Index exceeds matrix dimensions.
```

O operador (:) é muito útil para manipular o vetor ou gerar um novo vetor:

```
» A( 1 : 3 )
ans =
       2  4  6
» A( 4 : 6 )
ans =
       8 10 12
» A( : )
 ans =
       2  4  6  8  10  12  14
» A( 1 : 2 : 7  )
 ans =
       2  6  10  14
```

Mas se indicarmos um valor final maior que o tamanho do vetor, resulta em erro:

```
» A( 1 : 10 )
???  Index exceeds matrix dimensions.
```

Para alterar algum elemento do vetor, basta atribuir um valor:

```
» A( 3 ) = 0
A =
     2  4  0  8  10  12  14
```

```
» A( 3 ) = 6;  % colocando o valor original na terceira posição do vetor A
```

A seguinte linha de comando retorna o tamanho do vetor A:

```
» length(A)
ans =
      7
```

Todas as operações aritméticas de vetores com escalares e vetores com vetores validas são possíveis:

```
» b =  2;
» C = b*A
C =
     4    8    12    16    20    24    28
```

```
» C = b + A
 C =
     4    6    8    10    12    14    16
```

```
» C-A
ans =
     2    2    2    2    2    2    2
```

Para multiplicar dois vetores, elemento a elemento, utiliza-se o operador (.) (ponto):

```
» C.*A
ans =
      8   24   48   80   120   168   224
```

Deve-se notar que a operação de multiplicação de vetores resulta em erro:

```
» C*A
??? Error using ==> *
 Inner matrix dimensions must agree.
```

Para transformar o vetor linha, em vetor coluna usa-se o operador (’) (apóstrofe):

```
» At = A'


At =
     2
     4
     6
     8
    10
    12
    14
```

Note que a posição dos elementos no vetor não se altera:

```
» At(3)
ans =
     6
```

Matrizes

Os vetores, na verdade são casos especiais de matrizes unidimensionais. Para se criar matrizes deve-se utilizar o operador (;), que dentro de um *array* indica o final de uma linha.

```
» M = [ 1  2  3 ; 4  5  6; 7  8  9 ]
M =
     1    2    3
     4    5    6
     7    8    9
```

Caso cada linha da matriz seja composta por vários elementos, pode-se usar o operador (...) , que interrompe a linha atual para continuar na próxima linha.

```
» M = [   1  2  3  ; ...
          4  5  6  ; ...
          7  8  9 ]
  M =
       1    2    3
       4    5    6
       7    8    9
```

Assim como os vetores, cada elemento, linha ou coluna da matriz pode ser chamada individualmente:

```
» i = 3;   % indica  o número da linha
» j = 2;   % indica  o número da coluna

» M( i , j)
ans =
     8
» M( 1 , :)
```

```
ans =
     1  2  3

» M(: , 3 )
ans=
     3
     6
     9

» M( 1:(i-1) , 1:j )  % equivalente a M(1:2,1:2) ou M([1 2],[1 2])
ans =
     1    2
     4    5
```

O operador (%) insere comentários que não são processados pelo MatLab.
Para alterar um elemento, linha ou coluna basta atribuir um valor:

```
» [ zeros(1,3) ; M(2,:); M(3,:)]
%zeros gera um array de elementos nulos, neste caso um vetor linha de três posições
 ans =
      0    0    0
      4    5    6
      7    8    9

» [ones(3,1)' ; M(:,2)' ; M(:,3)' ] '
% ones gera um array de elementos unitários, neste caso um vetor coluna de três posições.
 ans =
      1    2    3
      1    5    6
      1    8    9
```

A transposta da matriz pode ser obtida utilizando o operador (‘):

```
» M’
ans =
     1    4    7
     2    5    8
     3    6    9
```

Veja a atuação do operador (.) no exemplo a seguir:

```
» M*M
ans =
     30    36    42
     66    81    96
    102   126   150

» M.*M
ans =
      1     4     9
     16    25    36
     49    64    81
```

Funções

Para facilitar o usuário, MatLab inclui muitas funções *standard*. Cada função é um bloco com código que executa determinada tarefa. MatLab contem todas as funções matemáticas normalmente utilizadas, tais como sin, cos, log, exp, sqrt e muitas outras. Constantes normalmente utilizadas tais como pi, i (ou j) que representa a raiz quadrada de -1, são também, incorporadas ao MatLab.

```
» sin(pi/4)
 ans =
        0.7071
```

Por *default* mostra os resultados aritméticos em quatro dígitos porem as operações são realizadas com dupla precisão ou seja 16 dígitos decimais.

O MatLab possui um sistema de *help on-line*, basta digitar help e o nome da função que procuramos informação.

```
» help abs
 ABS    Absolute value and string to numeric conversion.
        ABS(X) is the absolute value of the elements of X. When
        X is complex, ABS(X) is the complex modulus (magnitude) of
        the elements of X.
        See also ANGLE, UNWRAP
        ABS(S), where S is a MATLAB string variable, returns the
        numeric values of the ASCII characters in the string.
        It does not change the internal representation, only the
        way it prints.
        See also SETSTR.
```

Existem dois tipos de funções do Matlab: as funções internas ou fechadas, implementadas de maneira mais eficientes possíveis (tempo de processamento) e as funções m, geradas a partir da linguagem de programação do MatLab. O comando type mostra a função m, mas não é habilitada para as funções internas.

```
» type abs
abs is a built-in function

» type sin
sin is a built-in function.

» type angle

function p = angle(h)
%ANGLE          Phase angle.
%       ANGLE(H) returns the phase angles, in radians, of a matrix with
%       complex elements.
%       See also ABS, UNWRAP.
%       Copyright (c) 1984-94 by The MathWorks, Inc.
% Clever way:
% p = imag(log(h));
% Way we'll do it:
p = atan2(imag(h), real(h));
```

Funções de uso geral:

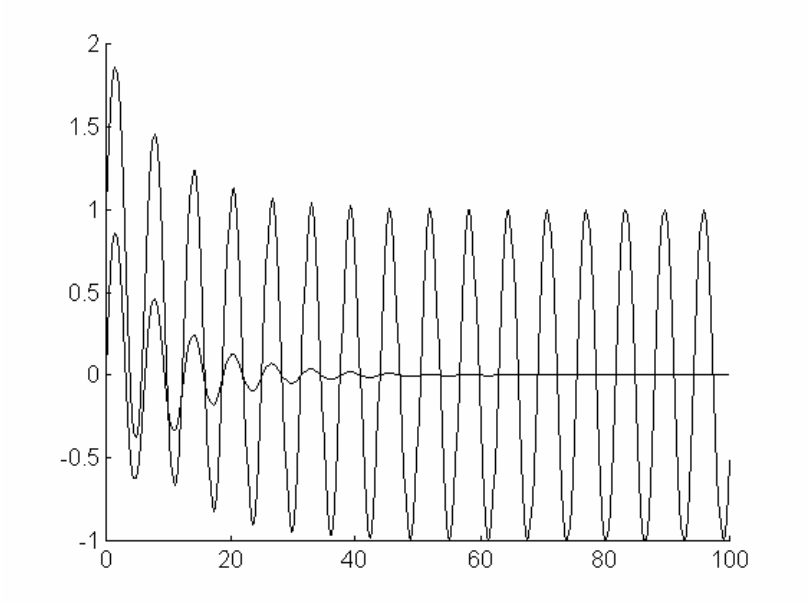
clear variável	apaga variável da memória
clear all	apaga todas as variáveis da memória
close(n)	apaga a figura de número n
close all	apaga todas as figuras
whos	relaciona as variáveis amarzenadas na memória
who	relaiona de forma simplificada as variaveis
which função	localiza o path de uma função
cd e dir	comandos dos
!comando do dos	outros comandos dos

Gráficos

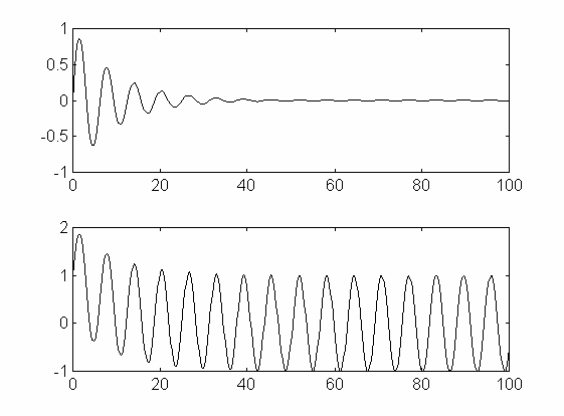
A principal função para gerar gráficos no MatLab é plot. Existem varias maneiras de utilizar este comando:

```
» a = 0.1;
» b = 1;
» t = 0:0.25:100;          %gera o vetor de tempo
» s1 = exp(-a*t);          %gera a função exponencial decrescente
» s2 = sin(b*t);           %gera a função seno
» y1 = s1.*s2;
» y2 = s1+s2;

» figure(1)                %dois gráficos superpostos gerados por comandos plot diferentes
» hold on
» plot(t,y1)
» plot(t,y2)
» hold off
```



```
» figure(2)                % dois gráficos superpostos gerados por um único comando plot
» plot(t', [y1' y2'] )
» figure(3)                % dois gráficos superpostos gerados por um único comando plot
» plot(t,y1,':r',t,y2,'-.b')  % com tempos independentes para cada gráfico
» figure(4)                % divide o gráfico em duas partes, uma para cada plot
» subplot(2,1,1)
» plot(t,y1)
» subplot(2,1,2)
» plot(t,y2)
```



Comandos auxiliares para gerar gráficos:

```
» close(4)     %fecha afigura 4
» close all    %fecha todas as figuras
» figure(1)
» plot(t,y1,':w',t,y2,'-.w')
» title('titulo do grafico')
» xlabel( 'referência eixo x')
» ylabel('referência eixo y')
» legend('y1','y2')
» text(60,1.5,'inserir texto')
```