**MATLAB**

Matlab é um programa interativo para cálculo numérico e científico; é extensivamente usado por engenheiros de controle para análise e projeto. As caixas de ferramentas (tools Box) foram incorporadas ao longo do tempo ao MatLab estendendo suas as funções em diversas áreas (controle, processamento de sinais...).

Foi inicialmente concebido como um programa que permitisse uma utilização mais facilitada dos pacotes de manipulação de matrizes. Atualmente é completamente escrito em C e constitui-se em um sistema integrado completo, incluindo funções gráficas e possibilitando o desenvolvimento de programas através de uma linguagem própria de fácil aprendizado. Para mais informação sobre Matlab, visite o site da Mathworks (www.mathworks.com).

**Vetores**

Criando um vetor linha:

```
um = [1 2 3 4 5 6 9 8 7]
```

Matlab devolve:

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

Para criar um vetor com elementos pares entre 0 e 20:

```
t = 0:2:20
```

```
t =
    0 2 4 6 8 10 12 14 16 18 20
```

**Manipulando vetores**

Somar o valor 2 a cada um dos elementos do vetor 'um':

```
b = um + 2

   b =
      3 4 5 6 7 8 11 10 9
```

Somando dois vetores. Se os dois vetores forem o mesmo comprimento, é fácil. Simplesmente some os dois como mostrado abaixo:

```
c = um + b
    c =
      4 6 8 10 12 14 20 18 16
```

Subtração de vetores dos mesmos trabalhos de comprimento exatamente o mesmo modo.

**Funções**

O Matlab inclui várias funções padrões. Cada função é um bloco de código que realiza uma tarefa específica. São funções padrões: sin, cos, log, exp, sqrt, entre outras. Constantes geralmente usadas como pi, e i ou j para a raiz quadrada de -1, também está incorporado em Matlab.

```
sin(pi/4)

ans =
    0.7071
```

Para determinar o uso de qualquer função, tecle help [nome de função] na janela de comando do Matlab. Matlab permite escrever suas próprias funções com o comando de function

**Polinômios**

Em Matlab, um polinômio é representado por um vetor. Para criar um polinômio em Matlab, simplesmente entre cada coeficiente do polinômio no vetor em ordem descendente. Por exemplo, digamos que você tem o seguinte polinômio:

$$s^4 + 3s^3 - 15s^2 - 2s + 9$$

Para entrar com este polinômio no Matlab, faça:

```
x = [1 3  -15 -2 9]
x =
   1 3 -15 -2 9
```

Matlab pode interpretar um vetor de n+1 de comprimento como um polinômio de ordem de nth. Assim, se houve coeficientes que não existam no seu polinômio você tem que entrar com zeros no lugar apropriado do vetor. Por exemplo,

$$s^4 + 1$$

seria representado em Matlab como:

```
y = [1 0 0 0 1]
```

Você pode achar o valor de um polinômio utilizando a função de polyval. Por exemplo, achar o valor do polinômio anterior em s = 2,

```
z = polyval([1 0 0 0 1],2)

    z =
        17
```

Você também pode extrair as raízes de um polinômio. Isto é útil quando você tiver um polinômio de alto-ordem como

$$s^4+3s^3-15s^2-2s+9$$

Achando as raízes seriam tão fáceis quanto entrando no comando seguinte;

```
                    roots([1 3 -15 -2 9])
    ans =
     -5.5745
      2.5836
     -0.7951
      0.7860
```

O produto de dois polinômios é realizado através da convolução de seus coeficientes deles. A função conv do Matlab realiza a convolução.

```
x = [1 2];
y = [1 4 8];
z = conv(x,y)

    z =
        1   6   16   16
```

A divisão de dois polinômios é realizada com a função deconv que devolve o resto e o resultado. Dividamos z por y e veja se nós adquirirmos x.

```
[xx, R] = deconv(z,y)

    xx =
        1 2
    R =
        0 0 0 0
```

**Matrizes**

Entrar com matriz no Matlab é igual a entrar com um vetor, sendo que cada linha de elementos será separada por um ponto-e-vírgula (;) ou um retorno:

```
B = [1 2 3 4;5 6 7 8;9 10 11 12]

    B =
        1 2 3 4
        5 6 7 8
        9 10 11 12
B = [1 2 3 4
     5 6 7 8
     9 10 11 12]

    B =
        1 2 3 4
        5 6 7 8
        9 10 11 12
```

Para achar a matriz transposta utilize o apóstrofo(‘):

```
C = B’

   C =
       1 5 9
       2 6 10
       3 7 11
       4 8 12
```

Deve-se notar que se a matriz C fosse complexa, a apóstrofe teria dado o complexo conjugado transposto. Para adquirir somente a transposta, use (.') (os dois comandos são o mesmo se a matriz não for complexa).

```
       Z = [1 2; 3 4] + [5 6; 7 8]*i
       Z1 = Z’
       Z2 = Z.’

Z =
  1.0000 + 5.0000i   2.0000 + 6.0000i
  3.0000 + 7.0000i   4.0000 + 8.0000i


Z1 =
  1.0000 - 5.0000i   3.0000 - 7.0000i
  2.0000 - 6.0000i   4.0000 - 8.0000i

Z2 =
  1.0000 + 5.0000i   3.0000 + 7.0000i
  2.0000 + 6.0000i   4.0000 + 8.0000i
```

Multiplicando as matrizes B * C e C * B.

D = B * C

D =

30 70 110

70 174 278

110 278 446

D = C * B

D =

107 122 137 152

122 140 158 176

137 158 179 200

152 176 200 224

Outra opção para manipulação de matrizes é multiplicação dos elementos correspondentes de duas matrizes. Para isso é utilizado o operador (.*) (as matrizes devem ser o mesmo tamanho para fazer isto).

E = [1 2;3 4]
F = [2 3;4 5]
G = E. * F

E =

1 2

3 4

F =

2 3

4 5

G =

2 6

12 20

Se você tem uma matriz quadrada, como E, que você também pode multiplicar por ela mesmo tantas vezes quanto se queira, bastando para isso elevar a uma determinada potência.

E^3

ans =

37 54

81 118

Se quiser elevar cada elemento ao cubo basta usar o operador (.^).

```
E. ^3

   ans =
       1 8
      27 64
```

Para achar o inverso de uma matriz:

```
X = inv(E)

   X =
     -2.0000 1.0000
      1.5000 -0.5000
```

Para achar seus auto-valores:

```
eig(E)

   ans =
     -0.3723
      5.3723
```

Há uma função até mesmo para achar os coeficientes do polinômio característico de uma matriz. A função de "poly" cria um vetor que inclui os coeficientes do polinômio característico.

```
p = poly(E)
p =
   1.0000 -5.0000 -2.0000
```

Lembre-se que os auto-valores de uma matriz são iguais às raízes de seu polinômio característico:

```
roots(p)

ans =
    5.3723
   -0.3723
```

**OBS**: Note que se alguma variável não for nomeada, o Matlab a armazena em uma variável temporária chamado "ans."

**Gráficos**

Também é fácil de criar gráficos em Matlab. Para desenhar a função seno em função de tempo. Primeiro crie o vetor de tempo (o ponto-e-vírgula depois que cada declaração faz com que o Matlab não mostre os valores) e então compute o valor do seno em cada momento e utilize a função plot para criar o gráfico.

***Gráficos Bidimensionais***

Comandos:

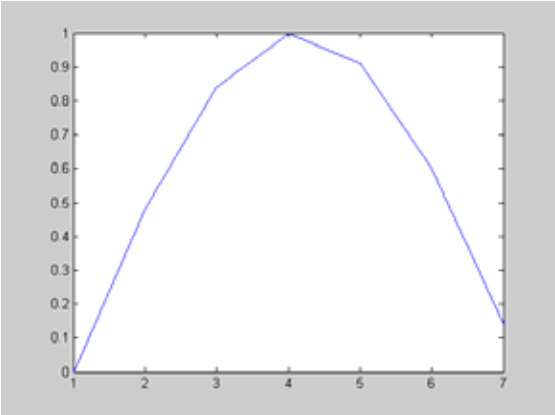
- plot - cria um gráfico de vetores ou de colunas de matrizes
- loglog - cria um gráfico utilizando escalas logarítmicas para ambos os eixos
- semilogx - cria um gráfico utilizando escala logarítmica no eixo x e escala linear no eixo y
- semilogy - cria um gráfico utilizando escala logarítmica no eixo y e escala linear no eixo x
- hist – plota histograma
- fplot – plota função
- fill – desenha polígono 2D
- bar – gráfico de barras
- polar – plota em coordenada polar

Pode-se adicionar título, nome aos eixos, linhas de grade e textos a qualquer gráfico utilizando

- title - adiciona um título ao gráfico
- xlabel - define um nome para a variável do eixo x
- ylabel - define um nome para a variável do eixo y
- text - adiciona um texto em lugar específico
- gtext - adiciona um texto ao gráfico utilizando o mouse
- grid - ativa as linhas de grade

Y = [0 0.48 0.84 1 0.91 0.6 0.14];

plot (Y)

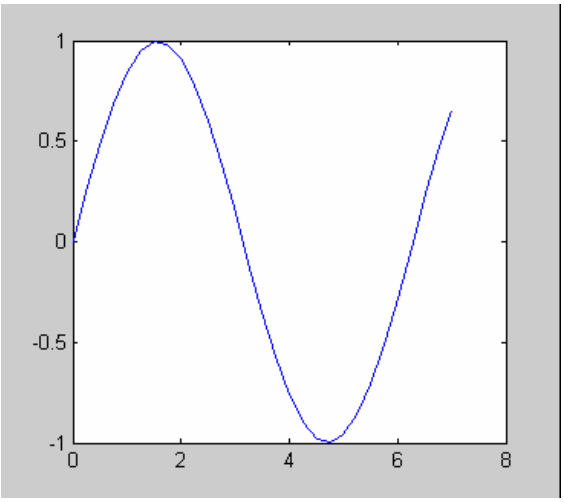



Se X e Y são vetores com dimensões iguais, o comando plot(X,Y) produz um gráfico bidimensional dos elementos de X versos os elementos de Y, por exemplo:

t=0:0.25:7;

y = sin(t);

plot(t,y)




***Estilos de linha e símbolo***

Pode ser passada uma string de caracteres como um argumento para a função plot de modo a especificar vários estilos de linhas, símbolos de traçado e cores. Na linha de comando,

plot(X,Y,S)

S é uma string envolvida por aspas e construída com os caracteres mostrados na tabela abaixo:

| **Símbolo** | **Cor** | **Símbolo** | **Estilo de Linha** |
|---|---|---|---|
| y | amarelo | . | ponto |
| m | magenta | o | círculo |
| c | ciano | x | x |
| r | vermelho | + | sinal positivo |
| g | verde | * | asterisco |
| b | azul | - | sólida |
| w | branco | : | pontilhada |
| k | preto | -. | traço e ponto |
| | | -- | tracejada |

Por exemplo, plot(X,Y,'b*') plota um asterisco azul em cada ponto de dado.

***Subplot(m, n,p)*** – divide a tela gráfica numa matriz mxn e seleciona a célula p para plotar o gráfico. Ex:

```
x = linspace(0,2*pi,50);
y = sin(x);
z = cos(x);
w = tan(x);

subplot(2,2,1)
plot(x,y)
subplot(2,2,2)
plot(x,z)
subplot(2,2,3)
plot(x,w)
```

**Plotando Gráficos Tridimensionais e Contornos**

| | |
|---|---|
| plot3 | plota em espaço 3D |
| fill3 | desenha polígono 3D |
| comet3 | plota em 3D com trajetória de cometa |
| contour | plota contorno 2D |
| contour3 | plota contorno 3D |
| clabel | plota contorno com valores |
| quiver | plota gradiente |
| mesh | plota malha 3D |
| meshc | plota malha 3D com contorno |
| surf | plota superfície 3D |
| surfc | plota superfície com contorno |
| slice | plota visualização volumétrica |
| cylinder | gera cilindro |
| sphere | gera esfera |

```
x = [-2:.2:2]
y = [-2:.2:2]
[x,y] = meshgrid(x,y)
z = x .* exp(-x.^2 - y.^2);
mesh(x,y,z)

contour3(z)

title('titulo do grafico')
Xlabel('eixo x')
Ylabel('eixo y')
```

**Armazenamento e recuperação de dados**

O espaço de trabalho pode ser salvo através do comando save. A execução deste comando cria um arquivo com extensão .mat no disco rígido. Os dados deste arquivo podem ser recuperados através do comando load. Exemplo:

```
who             % mostra todas variáveis utilizadas
save arq1       % salva as variáveis no arquivo arq1.mat
dir *.mat       % mostra os arquivos das variáveis salvas
clear           % limpa todas as variáveis
who             % não deve haver nenhuma variável depois do comando clear
load arq1       % carrega as variáveis do arquivo arq1.mat
whos            % mostra as variáveis e seus respectivos tamanho
```

**Instruções de Seleção e Controle de Fluxo**

Os loops sempre precisam de um end para indicar seu fim

**i) Loop For**

```
for i =1:5, x(i) = 0, end

for i = 1:5
    for j = 1:4
        A(i, j) = 1/(i+j-1);
    end
end
```

***ii) Loop While***

O loop while repete o ciclo enquanto a condição for verdadeira.

```
n = 1;
while prod(1:n) <1e100          % prod → realiza o produto dos elementos
      n = n +1;
end
disp('fim do laco')             % escreve mensagem na tela
n
```

***iii) Instruções If e Break***

Se a condição do If for verdadeira, seus comandos são processados, caso contrário são ignorados.

```
EPS = 1
for num = 1:100
    EPS = EPS/2;
    if(1+EPS) <= 1
        EPS = EPS*2;
        break                    % comando usado para sair antecipadamente do loop for
    end                          % comando usado para sair do if
end                              % comando usado para sair do for
EPS, num
```

**ARQUIVOS**

O matlab é capaz de executar seqüências de comandos armazenadas em arquivos de extensão .m (M-files). Um arquivo M-file é um arquivo texto ASCII, constituído de uma seqüência de instruções do MATLAB.

Crie o seguinte arquivo e salve como teste.m:

```
% Arquivo que traça superfícies de seno
t = -8:.3:8;
x  = sin(t)./t;
xx = [x;x;x;x;x;x;x;x;x;x;x];
mesh(xx);
```

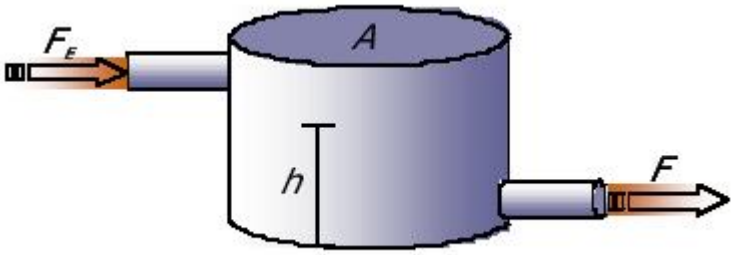
**Arquivos de funções**

Um arquivo de função é um arquivo que contém instruções do MATLAB com extensão .m que permite a passagem de argumentos. Exemplo:

```
function y = media(x)           % cálculo de média aritmética
% Para vetores, media(x) retorna a média aritmética dos elementos de x
% Para matrizes, media(x) é um vetor linha contendo o valor médio de cada coluna
[m n] = size(x);
if m == 1
        m = n;
end
y = sum(x)/m;
```

Exercícios:

1) Considere o tanque de nível da figura abaixo tenha a vazão $F_E$, a densidade ρ e a temperatura T1 constantes e que no instante inicial h(t=0) = h0:




Para o balanço das massas:

$$\frac{dm(t)}{dt} = \rho(F_E - F) \qquad \text{Ou} \qquad \frac{dm(t)}{dt} = \rho A \frac{dh(t)}{dt}$$

Portanto: $$\frac{dh(t)}{dt} = \frac{1}{A}(F_E - F)$$

Como a razão de saída do tanque é proporcional a altura de saída do líquido (h) e inversamente proporcional a resistência de escoamento (R):

F = h/R

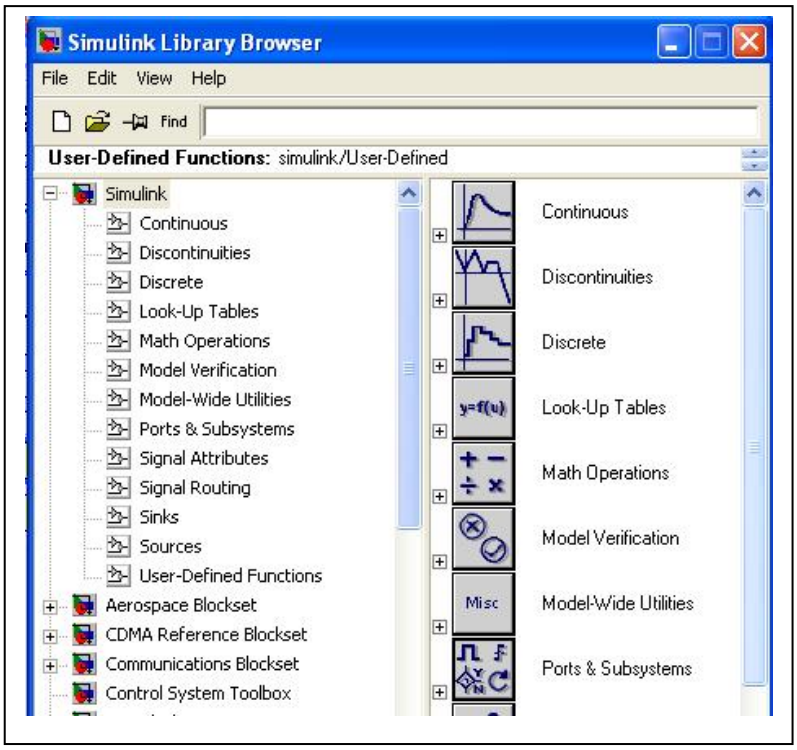
Tem-se: $$\frac{dh(t)}{dt} = \frac{1}{A}(F_E - \frac{h}{R})$$

Faça um arquivo para modelar o tanque que plote o gráfico de h x t, usando R = 1, A = 2 e $F_E$ = 10.

**SIMULINK**

O Simulink é uma opção em Matlab para simulação de sistemas dinâmicos. Estes sistemas poderiam ser implementados computacionalmente em Matlab, mas podem ser resolvidos de forma mais simples através desta opção.

Abra o simulink digitando “simulink” na janela de comando do Matlab. Será aberta a tela do simulink:




Pela figura acima, nota-se que o simulink possui diversos blocos separados por categorias. Por exemplo, entre outros, na categoria:

- Continuous: tem os blocos que descrevem funções lineares
- Discontinuous: blocos que descrevem funções não lineares
- Math: bloco que descreve funções matemáticas
- Sources: blocos para gerar sinais
- Sinks: blocos para mostrar ou arquivar as saídas

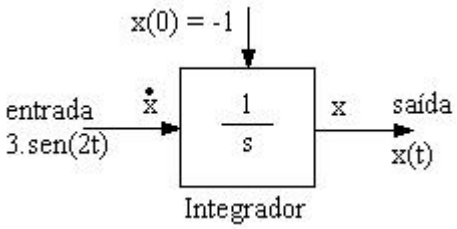
Construindo um modelo simples

Construindo um modelo para resolver a equação diferencial:

x’ = 2.sin(2t)

Condição inicial: x(0) = -1.

- Forçamos que já exista a derivada x’, então para calcular x(t) necessitaria apenas de integrar esse valor:




- Trazer os blocos da fonte senoidal, do integrador e do osciloscópio, fazendo a seguinte montagem:

- Dê dois cliques no bloco da função senoidal para configurar o seno para a amplitude de 3 e freqüência igual 2.

- Dê dois cliques no integrador para configurar a condição inicial = -1.

- Após isso é só simular o modelo.