Laboratório de Controle
Prof. Celso J. Munaro - Departamento de Engenharia Elétrica - UFES

Matlab: comandos básicos, modelos e simulações

Parte 1 – Comandos básicos

2.1 Operações básicas

Matlab como uma calculadora

```
>> 3*4*(2+5^2)
ans =

   324

>> exp(3/4)

ans =

    2.1170
```

Definir uma matriz:

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

A =

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

Definir um vetor:

```
>> r = [10 11 12];
```

Juntar um vetor e uma matriz:

```
>> B=[A r’ ]
B =

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

Acessar a última linha da matriz B

```
B(3,:);
```

Transposto

Basta usar apóstrofe

```
>> x=[1 2 3]

x =

     1     2     3
```

```
>> x'

ans =

     1
     2
     3
```

Criando um vetor com 10 números aleatórios usando distribuição normal
x=randn(10,1);

Números complexos:

```
>> z=2+4*i

z =

   2.0000 + 4.0000

>> imag(z)

ans =

     4
```

Infinito

```
>> c=1/0

c =

   Inf
```

NaN

```
>> c/c

ans =

   NaN
```

2.2 Comandos lógicos

```
>> 5==4

ans =

  logical

   0

>> 2<3

ans =
```

```
  logical

   1
```

Definindo a variável lógica:

```
>> p=true

p =

  logical

   1
```

Outro exemplo:

```
>> x=[1 2 3 4 5]

x =

     1     2     3     4     5

>> x<4

ans =

  1×5 logical array

   1   1   1   0   0
```

2.3 Operações com vetores e matrizes

Limpar todas variáveis
>> clear

Salvar todas as variáveis em um arquivo
>> save arquivo1.mat

Salvar apenas X e Y no arquivo
>> save arquivo2.mat X Y

Para recuperar os dados de um arquivo .mat
>> load arquivo2.mat

Outros comandos úteis: who, whos, importdata

Tabela 1: Comandos muito utilizados em matrizes e vetores

eig(a)	Autovalores de A
norm(a)	Norma
svd(a)	Valores singulares
inv(a)	Inversa
pinv(a)	Pseudoinversa

poly(a)	Polinômio característico
det(a)	Determinante
bsxfun(@minus, A, b)	Subtrai cada elemento de b de cada coluna de A. O vetor b deve ter o mesmo número de colunas de A
size(a)	Dimensões de A
length(x)	Comprimento do vetor x
sum(x)	Soma dos elementos do vetor x
mean(x)	Média de x
std(x)	Desvio padrão de x
roots(p)	Raízes de um polinômio definido pelo vetor p
x=linspace(a,b,N)	Gera um vetor x de a até b com N elementos uniformemente espaçados
A.^2	Eleva cada elemento de A ao quadrado
find(A>k)	Encontra os elementos da matriz (ou vetor) A que são maiores que k, fornecendo seus índices.
abs(x)	Valor absoluto de x

2.4 Gerando matrizes especiais

Tabela 2. Comandos para gerar matrizes especiais

zeros(N)	Matriz quadrada de zeros com dimensão N
zeros(N,M)	Matriz NxM de zeros
ones(N,M)	Matriz NxM de uns
eye(N)	Matriz identidade de dimensão N
randn(N,M)	Matriz com a dimensão de NxM com valores aleatórios de uma distribuição normal
rand(N,M)	Matriz com a dimensão de NxM com valores aleatórios de uma distribuição uniforme
magic(N)	Retorna uma matriz NxN construída a partir dos inteiros de 1 a N^2 com somas iguais de linha e coluna
randi(P,N,M)	Matriz de dimensão NxM com valores inteiros pseudoaleatórios variando de 1 a P.
triu(A)	Retorna a matriz triangular superior de A

2.5 Comandos para plotar sinais e matrizes

Tabela 3: Comandos

stairs(t,x);	Gráfico em escada (sistemas discretos)
title('Fig. 2');	Define o título
bar(x)	Gráfico de barras
xlabel('atraso');ylabel('erro');	Rótulos da abcissa e ordenada
legend('x','y')	Legendas de duas curvas x e y
subplot(3,1,1)	Gera 3 gráficos empilhados
hold on /off	Plota sobre o mesmo gráfico ou não
errorbar	Plota uma barra com o erro em cada sinal
histogram(x)	Histograma de x
axis	Define limites nos eixos x e y

imagesc(C)	Exibe os dados da matriz C como uma imagem que usa toda a gama de cores no mapa de cores (colormap)
imshow('arquivo')	Abre figura e mostra a imagem do arquivo
xlim e ylim	Define limites para eixo x ou y
scatter(x,y)	Gráfico de dispersão de x versus y

Exemplo:

```
>> x = (0:1/2000:1)';
>> plot(x,cos(tan(pi*x)))
```

Tabela 4: Formatando figuras

plot(x,'LineWidth',3)	Escolhe a espessura da linha plotada
title('Titulo', 'FontSize',18)	Define o tamanho da fonte do texto
title('y=x^2')	Escreve $y = x^2$ na figura
xlabel(('y=\alpha^3');	Escreve a letra grega α^3 na abcissa
text(n,m,'Texto'))	Escreve 'Texto' nas coordenadas (x,y) da figura
[xg,yg]=ginput(N)	Armazena nos vetores xg e yg as coordenadas dos N pontos do gráfico clicados.
gtext('Texto')	Coloca Texto no ponto do gráfico onde se clica

2.6 Fluxo de controle

For

```
>> for i=1:10 x(i)=i; end
```

While

```
i =
    1
>> while(i<10) x(i)=i; i=i+1; end;
```

2.7 Arquivos e funções

Scripts são uma sequência de comandos, usando variáveis do workspace . São úteis para evitar a repetição de comandos, sendo salvos em arquivos no formato *.m

Funções começam com a palavra function, e têm argumentos de entrada e saída. Exemplo:

```
function y = mean(x)
% MEAN Average or mean value
% For vectors, Mean(x) returns the mean value
% For matrices, MEAN(x) is a row vector
% containing the mean value of each column.

[m,n] = size(x);
if m == 1
    m = n;
end
y = sum(x)/m;
```

As linhas comentadas com % são mostradas ao dar o comando help ou doc.

2.8 Tipos de variáveis

Variáveis tipo char:

```
>> s1='Bom '

s1 =

   'Bom '

>> s2='dia!'

s2 =

   'dia!'

>> [s1 s2]

ans =

   'Bom dia!'

>> idade=18

idade =

   18

>> ss=sprintf('Minha idade é %d',idade)

ss =

   'Minha idade é 18'
```

Definindo variáveis tipo struct

Criaremos a variável estudante com 3 campos (fields)

```
>> estudante.nome='Pedro';
>> estudante.ID=201920357
>> estudante.notas=[9 8 9.5 10]

estudante =

  struct with fields:

     nome: 'Pedro'
       ID: 201920357
    notas: [9 8 9.5000 10]
```

Definindo uma função de transferência $G(s) = \frac{10}{s^2+2s+3}$

```
>> g=tf( 10, [1 2 3])
g =
      10
  -------------
  s^2 + 2 s + 3
```

Veja os campos da variável tipo g struct criada:

```
>> get(g)
      Numerator: {[0 0 10]}
    Denominator: {[1 2 3]}
       Variable: 's'
        IODelay: 0
     InputDelay: 0
    OutputDelay: 0
             Ts: 0
Etc,……
```

Acessando os campos:

```
>> den=g.Denominator{1}

den =

    1    2    3

>> num=g.Numerator{1}

num =

    0    0   10
```

Usando o comando stepinfo para obter os parâmetros da resposta de g

A função stepinfo retorna uma variável tipo struct

```
>> S=stepinfo(g)

S =

  struct with fields:

    RiseTime: 1.0402
    SettlingTime: 3.4043
    SettlingMin: 3.0185
    SettlingMax: 3.6948
    Overshoot: 10.8433
    Undershoot: 0
    Peak: 3.6948
    PeakTime: 2.2105
```

```
>> ts=S.SettlingTime (tempo de estabelecimento)

ts =

   3.4043
```

S. Overshoot = sobreelevação

Definindo um atraso para uma função de transferência

```
g.InputDelay=2
g =

                10
  exp(-2*s) * -------------

              s^2 + 2 s + 3
```

O campo atraso é um dos campos da variável g (FT), sendo nulo se não for definido.

2.9 Criando tabelas no Matlab

```
>> g=tf(1,[1 1 10]);
>> S=stepinfo(g);
>> Tipo={'Especificacoes'};
>> UP=S.Overshoot;
>> ts=S.SettlingTime;
>> tr=S.RiseTime;
>> Tabela1=table(Tipo, UP, ts, tr)

Tabela1 =

  1×4 table

          Tipo             UP        ts        tr
    __________________   ______    ______    ______

    {'Especificacoes'}   60.453    7.3148    0.3738
```

Parte 2 - Simulações usando o Matlab

2.1) Modelos: função de transferência e variáveis de estado

Seja como exemplo, a equação diferencial de segunda ordem

$$\ddot{x}(t) + a\dot{x}(t) + bx(t) = u(t)$$

Sua função de transferência é obtida aplicando a transformada de Laplace,

$$\frac{X(s)}{U(s)} = \frac{1}{s^2 + as + b}$$

Para o modelo em variáveis de estado, escolhendo $x_1(t) = x(t)$ e $x_2(t) = \dot{x}(t)$ resultam nas equações diferenciais de primeira ordem para os dois estados:

$\dot{x_1}(t) = \dot{x}(t) = x_2(t)$

$\dot{x}_2(t) = \ddot{x}(t) = u(t) - ax_2(t) - bx_1(t)$

Colocando na forma matricial, resulta

$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$

$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$

ou

$\dot{x} = Ax + Bu$

$y = Cx$

com $A = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$

2.2) Entrada dos Modelos no Matlab:

Para definir um modelo dado por sua função de transferência, usa-se o comando tf,

G=tf(num,den)

onde num contém os coeficientes do numerador de G(s) e den os coeficientes do denominador.

Para o exemplo dado,

G=tf(1,[1 a b])

Para definir um modelo em variáveis de estado, usa-se o comando ss,

sys1=ss(A,B,C,D)

onde A,B,C,D são as matrizes que definem o modelo. Caso D não exista, basta fazer D=0.

Para obter a função de transferência a partir do modelo em variáveis de estado,

[num,den]=ss2tf(A,B,C,D)

Para obter o modelo em variáveis de estado a partir da função de transferência,

[A,B,C,D]=tf2ss(num,den)

2.3) Simulação dos modelos no Matlab:

Considerando-se os modelos dados, G e sys1:

2.3.1 Resposta ao degrau:

step(G) ou [y,t]=step(G); (; é para ele não mostrar as variáveis y,t na tela)

step(sys1) ou [y,t,x]=step(sys1);

Para o caso de variáveis de estado, as variáveis y(saída), t(tempo), x (estados) são geradas.

2.3.2 Resposta ao impulso:

Trocar impulse por step nos comandos acima.

2.3.3 Resposta a um sinal qualquer u:

Definir u e t, e para variáveis de estado, o estado inicial x0, caso seja diferente de zero.

Em todos estes casos, pode-se escolher o tempo final de simulação.
Exemplos:
step(g,10);
[y,t,x]=step(sys1,10);

Exemplo 1:
Sinal degrau com 100 elementos: u=ones(100,1);
Vetor de tempo com 100 instantes de tempo: t=0:0.1:(99*0.1). Com este vetor de tempo, a simulação é feita para instantes de amostragem de 0.1s e até 9.9s. O vetor depende do modelo a ser simulado. Sugiro dar o comando [y,t]=step(G) para que o próprio Matlab escolha um bom vetor t.

Os vetores u e t devem ter o mesmo comprimento (comando size(u))

[y,t]=lsim(G,u,t);
*[y,t]=lsim(sys1,u,t); ou [y,t]=lsim(sys1,u*0,t,x0);*
sendo x0=[1;2], por exemplo

Exemplo 2:

Vetor de tempo com 100 instantes de tempo: t=0:0.1:(99*0.1)
Sinal senoidal com 100 elementos: u=sin(3*t);

Simular com o comando lsim como acima.

Ao gerar y,t,x, veja exemplos de comandos abaixo para plotar:

a) *plot(t,y);xlabel('Tempo(s)'); ylabel('Saída');title('Simulação');*
b) *plot(t,x);xlabel('Tempo(s)'); ylabel('Saída');legend('x1','x2');*

Outros comandos:
text(0.6,0.4,' y = exp(x)');
hold on;hold off;
subplot(221);

2.4) Simulando modelos discretos
O modelo discreto pode ser obtido discretizando o sistema com o comando c2d, especificando o tempo de amostragem Ts, conforme abaixo
sysd=c2d(sys,Ts) variáveis de estado
gd=c2d(g,Ts); função de transferência

Na simulação discreta não é necessário especificar o vetor de tempo t ao usar o comando lsim.
Exemplos:
[y,t,x]=lsim(sysd,ones(40,1));

2.5) Obtendo os parâmetros do modelo
Para uma ft definida como *g=tf(1,[1 2 1]),* o comando *get(g)* fornece todos parâmetros associados ao modelo g:

```
>> get(g)
      Numerator: {[0 0 1]}
    Denominator: {[1 2 1]}
       Variable: 's'
        IODelay: 0
     InputDelay: 0
    OutputDelay: 0
             Ts: 0
       TimeUnit: 'seconds'
      InputName: {''}
      InputUnit: {''}
     InputGroup: [1×1 struct]
     OutputName: {''}
     OutputUnit: {''}
    OutputGroup: [1×1 struct]
          Notes: [0×1 string]
       UserData: []
           Name: ''
   SamplingGrid: [1×1 struct]
```

O denominador de g pode ser obtido via g.Denominator{1}. Por exemplo, roots(via g.Denominator{1}) fornece os polos de g(s). Embora seja mais simples usar o comando pole(g).

2.6) Simulação de sistemas não lineares

Seja o sistema não linear dado pelas equações

$$\dot{x}_1(t) = x_1(t) + x_2^2(t)$$
$$\dot{x}_2(t) = x_1(t) + u(u)$$

Define-se a função s1nl.m abaixo

```
function dx=s1nl(t,x)

u=2;

dx(1,1)=x(1)+x(2)^2;

dx(2,1)=x(1)+u;
```

e executa-se no Matlab o comando

```
>> [t,x]=ode45('s1nl',[0 1],[-1 1]);

>> plot(t,x)
```

A função ode45 (ou ode23, etc) integra as equações diferenciais descritas por s1nl.m

Passando parâmetros para a simulação:

```
[t,y] = ode45(@(t,y) nivel1(t,y,A,a1,q1), [0 1000], h0);
```

```
function dx = nivel1(t,y,A,a1,qi)
%Simulacao de um sistema de nivel
dx=(1/A)*(qi*100/6-a1*sqrt(2*981*y));
end
```

2.7 Obtendo funções de transferência de malha fechada

Seja o digrama de blocos mostrado na figura 1, e as FTs C(s), G(s), e H(s)

Figura 1. Diagrama de blocos

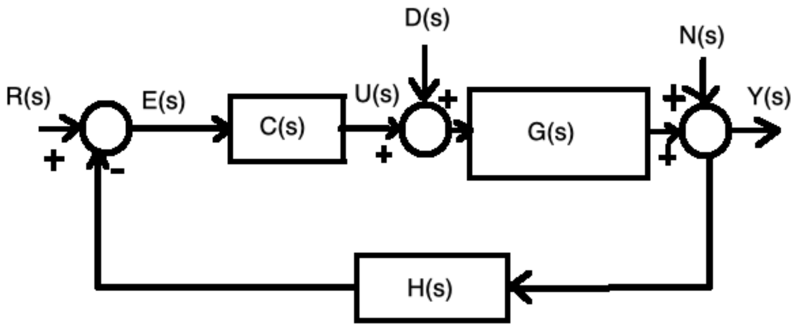


Tabela 3. FTs do sistema em malha fechada

FT	Comando
$M(s) = \frac{Y(s)}{R(s)}$	M=feedback(C*G,H)
$\frac{Y(s)}{D(s)}$	feedback(G,C*H)
$\frac{Y(s)}{N(s)}$	feedback(1,C*G*H)
$\frac{E(s)}{R(s)}$	feedback(1,C*G*H)

2.8 Estabilidade, polos e zeros

Seja G uma FT definida com o comando tf e sys um sistema no espaço de estados definido com o comando ss.

Polos do sistema: pole(g) ou pole(sys)
Zeros do sistema: zero(g) ou zero(sys)
Autovalores de sys: eig(sys)

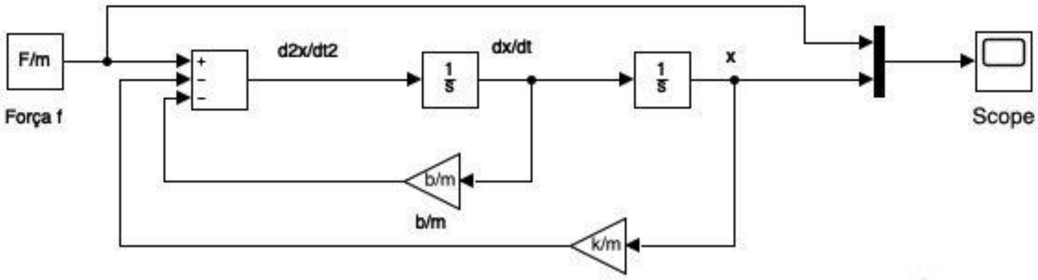
Para verificar a **estabilidade** basta calcular os polos. No caso contínuo, a parte real dos polos deve ser negativa. No caso discreto, o valor absoluto dos polos deve ser menor que 1 (polos dentro do círculo unitário)

2.9 Simulação no Simulink

Na figura 2 é mostrado um diagrama do Simulink. Para construí-lo basta arrastar os objetos da biblioteca para o modelo e conectá-los. O bloco scope mostra o resultado da simulação. Ele também foi configurado para salvar no workspace do Matlab o resultado da simulação. Basta clicar este bloco para configurar.

As constantes F,b,m,k usadas na simulação devem estar definidas no workspace, ou nos blocos onde estão.
O diagrama pode ser simulado com o comando sim do Matlab, na forma sim('arquivo', T), onde T é o tempo total de simulação. Se omitido, usa-se T definido no diagrama.

Figura 2. Diagrama do Simulink

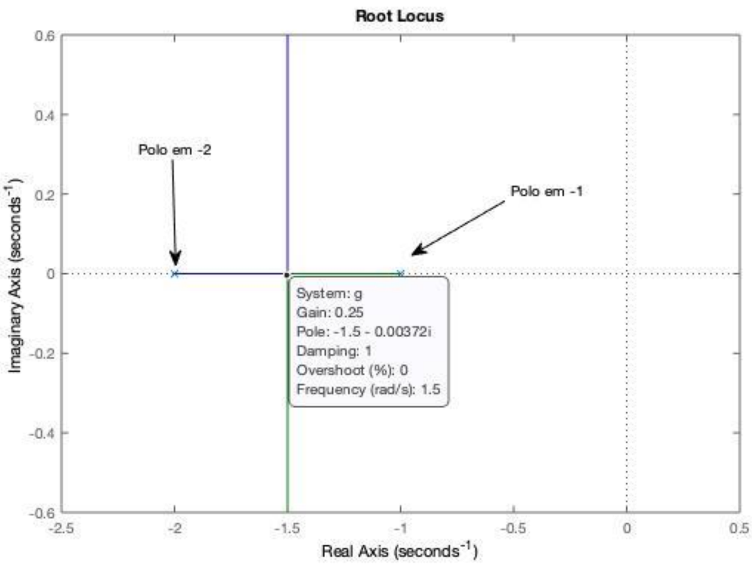


2.10 O método do lugar das raízes no Matlab

Dada uma de FT de malha aberta G, o método do lugar das raízes permite obter os polos malha fechada quando um parâmetro K varia, $M(s) = \frac{KG(s)}{1+KG(s)}$. Eles nada mais são do que as raízes do polinômio característico $p(s) = 1 + KG(s) = 0$.

No Matlab, calcula-se as raízes de $p(s)$ numericamente e plota-se. Uma forma de fazer isto é usar o comando rlocus. Por exemplo, rlocus(g). Neste caso, o Matlab escolhe os ganhos de K automaticamente. Eventualmente seja melhor escolher os ganhos, por exemplo, K=0:0.1:100. E então dar o comando rlocus (g,K), usando estes ganhos, mostrado na figura 3 para g=tf(1,[1 3 2]); Ao clicar sobre um ponto do gráfico pode-se obter 6 informações (ver figura 3).

Figura 3. Lugar das raízes de 1+KG(s)=0.



Fim !