Capítulo 1

Pré-Requisitos Básicos

1-1 INTRODUÇÃO*

Este capítulo apresenta o material básico para o projeto de sistemas de controle usando o pacote MATLAB. O MATLAB (uma abreviação para MATrix LABoratory) é um sistema baseado em matrizes, empregado em cálculos matemáticos e de engenharia. Pode-se imaginar o MATLAB como uma espécie de linguagem desenvolvida com o intuito de manipular matrizes. Todas as variáveis tratadas pelo MATLAB são matrizes. Isto quer dizer que o MATLAB só tem um tipo de dado, a matriz, que nada mais é que um arranjo retangular de números. O MATLAB tem um extenso conjunto de rotinas para obtenção de saídas gráficas.

Comandos e funções matriciais comumente empregados na solução de problemas de engenharia de controle

O pacote MATLAB tem diversas funções predefinidas que poderão ser chamadas pelo usuário para resolver uma ampla gama de problemas de engenharia de controle.

Tais comandos e funções estão listados na Tabela 1.1.

Entrando e saindo do pacote MATLAB

Na maioria dos sistemas, o MATLAB pode ser executado digitando-se MATLAB. Para sair do pacote, basta executar o comando exit ou o quit.

Como o MATLAB é usado

O pacote MATLAB normalmente é usado no modo comando. Quando são digitados comandos em uma linha, o MATLAB os processa imediatamente e mostra os resultados. O MATLAB também é capaz de executar seqüências de comandos armazenadas em arquivos.

Os comandos que são digitados na linha de comandos podem ser acessados posteriormente através da tecla *seta-para-cima*. Através desta tecla é possível passar pelos últimos comandos digitados, e recuperar uma linha específica.

Variáveis no MATLAB

Uma característica interessante do MATLAB é que suas variáveis não precisam ser dimensionadas antes de serem usadas. No MATLAB, as variáveis são geradas automaticamente por ocasião de sua utilização. (As dimensões das variáveis poderão ser alteradas mais tarde, se necessário.) Tais variáveis permanecem armazenadas na memória até a execução do comando exit ou do comando quit.

---

*O material apresentado neste capítulo é um resumo dos capítulos 1 e 2 do livro *Solving Control Engineering Problems with MATLAB* (*Solução de Problemas de Engenharia de Controle usando o MATLAB*), escrito por Katsuhiko Ogata, publicado em versão original pela Prentice Hall, Englewood Cliffs, New Jersey, e em versão traduzida para o português pela Prentice Hall do Brasil.

Tabela 1-1 COMANDOS E FUNÇÕES MATRICIAIS DO MATLAB

Comandos e funções matriciais usadas na solução de problemas de engenharia de controle	Explicações sobre o que o comando faz, ou sobre o que a função matricial significa, ou sobre o que a declaração significa
abs	Valor absoluto, magnitude complexa
angle	Ângulo da fase
ans	Resposta obtida quando a expressão não possuir atribuição
atan	Arco tangente
axis	Escala manual do eixo
bode	Gráfico do Diagrama de Bode
clear	Limpar a área de trabalho
clg	Limpar a tela gráfica
computer	Tipo de computador
conj	Conjugado complexo
conv	Convolução, multiplicação
corrcoef	Coeficientes de correlação
cos	Co-seno
cosh	Co-seno hiperbólico
cov	Covariante
deconv	Deconvolução, divisão
det	Determinante
diag	Matriz diagonal
eig	Autovalores e autovetores
exit	Término de programa
exp	Exponenciação, base e
expm	Exponencial matricial
eye	Matriz identidade
filter	Implementação de filtro direto
format long	Número real de 15 dígitos exemplo: 1.33333333333333
format long e	Número real de 15 dígitos na notação científica exemplo: 1.33333333333333E+000
format short	Número real de 5 dígitos exemplo: 1.3333
format short e	Número real de 5 dígitos em notação científica: exemplo: 1.3333E+000
freq	Resposta de freqüência da transformada de Laplace
freqz	Resposta de freqüência da transformada z
grid	Desenhar linhas de grade
hold	Manter na tela o gráfico corrente
i	$\sqrt{-1}$
imag	Parte imaginária de um número complexo
inf	Infinito
inv	Inverso
j	$\sqrt{-1}$
length	Tamanho do vetor
linspace	Vetores linearmente espaçados
log	Logaritmo natural
loglog	Gráfico loglog nos eixos $x - y$
logm	Logaritmo matricial
logspace	Vetores logaritmicamente espaçados
log10	Logaritmo na base 10
lqe	Projeto do estimador quadrático linear
lqr	Projeto do regulador quadrático linear

Tabela 1-1 COMANDOS E FUNÇÕES MATRICIAIS DO MATLAB (Cont.)

Comandos e funções matriciais usadas na solução de problemas de engenharia de controle	Explicações sobre o que o comando faz, ou sobre o que a função matricial significa, ou sobre o que a declaração significa
max	Valor máximo
mean	Valor médio
median	Valor mediano
min	Valor mínimo
NaN	"Não-numérico"
nyquist	Gráfico da resposta de freqüência em coordenadas de Nyquist
ones	Constante
pi	Número pi (π)
plot	Gráfico linear de coordenadas cartesianas x-y
polar	Gráfico de coordenadas polares
poly	Característica polinomial
polyfit	Traçado de curva polinomial
polyval	Cálculo polinomial
polyvalm	Cálculo de polinômio matricial
prod	Produto de elementos
quit	Término de programa
rand	Geração de números e matrizes randômicas
rank	Cálculo do posto (rank) de uma matriz
real	Parte real de um número imaginário
rem	Resto ou módulo
residue	Expansão em frações-parciais
rlocus	Traçar o lugar-das-raízes
roots	Raízes polinomiais
semilogx	Gráfico semilog x-y (eixo x logarítmico)
semilogy	Gráfico semilog x-y (eixo y logarítmico)
sign	
sin	Seno
sinh	Seno hiperbólico
size	Dimensão da linha e da coluna de uma matriz
sqrt	Raiz quadrada
sqrtm	Raiz quadrada de uma matriz
std	Desvio padrão
step	Gráfico de resposta ao degrau unitário
sum	Soma de elementos
tan	Tangente
tanh	Tangente hiperbólica
text	Texto posicionado arbitrariamente
title	Título do gráfico
trace	Traço da matriz
who	Lista de todas as variáveis atualmente na memória
xlabel	Título do eixo dos x
ylabel	Título do eixo dos y
zeros	Zero

Para obter uma lista das variáveis da área de trabalho corrente basta digitar o comando who. Todas as variáveis da área de trabalho corrente aparecerão na tela.

O comando clear elimina todas as variáveis não-permanentes eventualmente existentes na área de trabalho corrente. Se for preciso eliminar somente uma variável, digamos x, basta digitar o comando clear x.

Linha de programa começando com %

Ao longo deste livro serão apresentados inúmeros programas MATLAB, todos com comentários para tornar claros cada um dos passos executados por cada um dos programas. As linhas de programas MATLAB iniciadas por % são linhas de comentário. A notação % é similar ao REM da linguagem BASIC. As linhas de comentários não são executadas, ou seja, tudo o que aparecer depois de um % em uma linha de programa MATLAB é ignorado na execução. Se os comentários precisarem de mais de uma linha, cada uma delas deve começar com o símbolo %.

Uso do operador ponto-e-vírgula

O ponto-e-vírgula é usado para suprimir impressão. Se o último caractere de um comando for um ponto-e-vírgula, o comando é executado mas sem a impressão dos resultados gerados. Esta é uma característica muito útil na medida em que a impressão de resultados intermediários pode não ser necessária. O ponto-e-vírgula também é usado como indicador de final de linha quando se estiver dando entrada a uma matriz, exceção feita obviamente à última linha.

Uso do operador dois pontos

Os dois pontos têm uma função muito importante no MATLAB. Este operador pode ser usado para criar vetores, subscrever matrizes, e para especificar as iterações do comando *for*. Por exemplo, j:k é a mesma coisa que [*j* *j* + 1 ... *k*], A(:, j) é a *j*-ésima coluna da matriz **A**, e A(i,:) é a *i*-ésima linha da matriz **A.** Para as iterações *for*, ver, por exemplo, o Programa MATLAB 4-9.

Entrada de um comando longo que não cabe em uma linha

Um comando termina normalmente ao se pressionar a tecla de retorno de carro ou a tecla enter. Se o comando que estiver sendo digitado for muito grande para caber em uma única linha, pode-se digitar um sinal que consiste em três ou mais pontos, ..., seguidos do retorno do carro ou do enter, para indicar que o comando continua na outra linha. Um exemplo deste caso é apresentado a seguir:

```
x = 1.234 + 2.345 + 3.456 + 4.567 + 5.678 + 6.789...
    + 7.890 + 8.901 - 9.012;
```

Observe-se que os espaços em branco seguindo os sinais de =, de + e de − são opcionais. Tais espaços são colocados para tornar o comando mais legível.

Entrada de vários comandos em uma única linha

Pode-se colocar mais de um comando em uma única linha desde que eles sejam separados por vírgulas ou por ponto-e-vírgulas. Exemplos disto são mostrados a seguir:

```
plot(x,y,'o'), text(1,20,'System 1'), text(1,15,'System 2')
```

e

```
plot(x,y,'o'); text(1,20,'System 1'); text(1,15,'System 2')
```

Seleção do formato da saída

Todos os cálculos do MATLAB são realizados em precisão dupla. No entanto, a saída que aparece na tela pode ter quatro casas decimais. Por exemplo, no caso do vetor

```
x = [1/3  0.00002]
```

o MATLAB exibe a seguinte saída na tela:

```
x =

    0.3333    0.0000
```

Se pelo menos um dos elementos de uma matriz não for um inteiro exato, existem quatro possibilidades para o formato da saída. Tais possibilidades são as seguintes:

```
format short
format long
format short e
format long e
```

Uma vez referenciado, um determinado formato permanece válido até que venha a ser modificado por um comando explícito.

Para a análise de sistemas de controle, os formatos short e long são os mais usados. O formato default do MATLAB é o short. Portanto, sempre que não for executado nenhum comando explícito de mudança de formato, os resultados dos programas e comandos do MATLAB serão apresentados no formato short. Se todos os elementos de uma matriz ou de um vetor forem inteiros, então o formato short e o formato long vão gerar o mesmo resultado.

Como salvar variáveis ao término da execução do pacote MATLAB

Quando os comandos exit ou quit são digitados, todas as variáveis eventualmente criadas quando da execução do pacote serão perdidas. Porém, se o comando save for digitado antes do quit ou do exit, tais variáveis poderão ser mantidas em um arquivo em disco denominado matlab.mat. Em execuções posteriores do pacote, o comando load faz com que a área de trabalho adquira o estado anterior ao término da execução.

1-2 OBTENÇÃO DE GRÁFICOS DE CURVAS DE RESPOSTA

O MATLAB tem um extenso conjunto de rotinas para obtenção de saídas gráficas. O comando plot cria gráficos lineares com eixos $x-y$. Os gráficos logarítmicos e polares são obtidos substituindo a palavra plot por loglog, semilogx, semilogy ou polar. Todos estes comandos são usados com o mesmo objetivo: desenhar um gráfico. A diferença é na escala dos eixos e na forma como os dados plotados aparecem no gráfico.

Gráfico $x-y$

Se **x** e **y** são vetores de mesma dimensão, o comando

```
plot(x,y)
```

plota os valores de y contra os valores de x.

Plotando várias curvas

Para plotar várias curvas em um mesmo gráfico, devemos utilizar o comando plot com vários argumentos.

```
plot(X1, Y1, X2, Y2, . . . , Xn, Yn)
```

As variáveis X1,Y1,X2,Y2, e assim por diante, são pares de vetores. Cada par x-y é plotado, gerando várias curvas no mesmo gráfico. O uso de vários argumentos tem a vantagem de permitir que vetores de dimensões diferentes apareçam no mesmo gráfico. Cada par usa um tipo diferente de linha no desenho do gráfico.

Para plotar mais de uma curva em um mesmo gráfico pode-se também usar o comando *hold*, que congela a curva corrente e inibe o seu apagamento. Portanto, é possível plotar curvas subseqüentes a uma curva original e enxergar o efeito conjunto das mesmas. Uma nova execução do comando hold libera o gráfico corrente.

Adição de linhas de grade, título do gráfico e identificação dos eixos x e y

Uma vez que um gráfico esteja na tela, é possível desenhar linhas de grade, dar um título ao gráfico e identificar os eixos x e y. Os comandos MATLAB para tais ações são:

```
grid     (linhas de grade)
title    (título do gráfico)
xlabel   (identificação do eixo dos x)
ylabel   (identificação do eixo dos y)
```

Observe-se que, com o gráfico na tela, as linhas de grade, o título do gráfico e as identificações dos eixos podem ser sucessivamente adicionadas a ele, através da execução em seqüência dos comandos acima mencionados.

Escrita de texto em um gráfico

Para escrever um texto começando em um ponto de coordenadas (*X, Y*) de um gráfico na tela, deve-se usar o comando

```
text(X,Y,'text')
```

Por exemplo,

```
text(3,0.45,'seno t')
```

escreve horizontalmente seno *t*, começando no ponto de coordenadas (3,0.45). Os comandos

```
plot(x1,y1,x2,y2), text(x1,y1,'1'), text(x2,y2,'2')
```

marcam duas curvas, de maneira a torná-las facilmente distinguíveis. (Veja Exemplos 2-1 e 2-2 para verificar como se escreve texto em gráficos na tela.)

Tipo de gráfico

O comando

```
plot(X, Y,'x')
```

desenha um gráfico de pontos usando × para marcar os pontos, ao passo que

```
plot(X1, Y1, ':', X2, Y2, '+')
```

usa uma linha pontilhada para a primeira curva e o símbolo + para a segunda curva. Outros tipos de linhas e de pontos são mostrados a seguir:

Tipos de linha		Tipos de ponto	
sólida	–	ponto	•
tracejada	– –	mais	+
pontilhada	:	asterisco	*
traço-ponto	–·	círculo	o
		marca-x	×

Cores

Os comandos

```
plot(X,Y,'r')
plot(X,Y,'+g')
```

indicam o uso de uma linha vermelha no primeiro gráfico, e de sinais de + verdes no segundo. As cores disponíveis são:

vermelha	r
verde	g
azul	b
branca	w
invisível	i

Algoritmos para plotagem automática

No MATLAB a escala dos gráficos é ajustada automaticamente. Um gráfico permanece como gráfico corrente até que outro gráfico seja plotado, resultando no apagamento do gráfico antigo, fazendo com que os eixos adqui-

ram novas escalas. Os algoritmos de plotagem automática para curvas de resposta a transitórios, de lugar das raízes, diagramas de Bode, gráficos de Nyquist são projetados para trabalhar com uma grande variedade de sistemas, mas não são perfeitos. Então, em certas situações, é desejável passar por cima da característica do comando plot de fazer com que os gráficos adquiram suas escalas automaticamente, selecionando manualmente os limites da plotagem.

Seleção manual da escala dos eixos

Se for necessário plotar uma curva em uma região especificada por

```
v = [x-min   x-max   y-min   y-max]
```

deve-se digitar o comando axis(v). O comando axis(v), onde v é um vetor de quatro elementos, ajusta a escala do eixo para os limites prescritos. No caso de gráficos logarítmicos, os elementos de v são $\log_{10}$ de mínimos e de máximos.

A execução de axis(v) congela a escala atual, fazendo-a valer para os próximos gráficos. Digitando novamente axis, o sistema readquire a característica de auto-escala.

O comando axis('square') faz com que a região da tela onde fica o gráfico seja um quadrado. Neste caso, uma linha com inclinação 1 está exatamente a 45°, não sendo afetada pelo formato irregular da tela do computador. O comando axis('normal') faz com que o aspecto volte ao normal.

1-3 CÁLCULO DE FUNÇÕES MATRICIAIS

Nesta seção será discutido o cálculo de normas, autovetores, autovalores, autovalores generalizados, autovetores generalizados, além de tratar da avaliação de polinômios, entre outros assuntos.

Normas

A norma de uma matriz é um escalar que dá uma medida do tamanho da matriz. Existem várias definições diferentes usadas com certa freqüência. Uma delas é a seguinte:

$$\text{norma}(\mathbf{A}) = \text{maior valor individual de } \mathbf{A}$$

Da mesma maneira existem várias definições para a norma de um vetor. Uma delas é a seguinte:

```
norm(x) =  sum(abs(x).^2)^0.5
```

Observe o seguinte exemplo:

```
x = [2   3   6];
norm(x)
ans =

    7
```

Autovalores e autovetores

Se **A** é uma matriz $n \times n$, então os n números λ que satisfizerem à relação

$$\mathbf{Ax} = \lambda\mathbf{x}$$

são os autovalores de **A**. Eles podem ser encontrados usando o comando

```
eig(A)
```

que retorna os autovalores em um vetor coluna.

Se **A** for real e simétrico, seus autovalores serão reais. Porém, se **A** não for simétrico, seus autovalores muito provavelmente serão números complexos.

Por exemplo, com

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

o comando

```
eig(A)
```

produz

```
ans =

   0 + 1.0000i
   0 - 1.0000i
```

As funções do MATLAB podem ter um ou vários argumentos de saída. Por exemplo, conforme visto acima, eig(A) produz um vetor coluna, cujos elementos são os autovalores de **A**. Já o comando de dupla atribuição

```
[X,D] = eig(A)
```

produz autovalores e autovetores. Os elementos da diagonal da matriz diagonal **D** são os autovalores, e as colunas de **X** são os autovetores correspondentes, de forma que

$$\mathbf{AX} = \mathbf{XD}$$

Por exemplo, se

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}$$

então a declaração

```
[X,D] = eig(A)
```

gera o resultado abaixo:

```
[X,D] = eig(A)

X =

   -0.5774    0.2182   -0.1048
    0.5774   -0.4364    0.3145
   -0.5774    0.8729   -0.9435


D =

   -1.0000         0         0
         0   -2.0000         0
         0         0   -3.0000
```

Os autovetores são calculados de maneira que a norma de cada um deles seja igual a 1.

Se os autovalores de uma determinada matriz forem diferentes, os autovetores serão sempre independentes, e o autovetor da matriz **X** vai diagonalizar a matriz original **A**. No entanto, se a matriz possuir autovalores repetidos, ela não poderá ser diagonalizada a não ser que ela tenha um conjunto completo (independente) de autovetores. Se os autovetores não forem independentes, a matriz original é dita ser defeituosa. Mesmo neste caso, a solução obtida a partir de eig satisfaz à relação $\mathbf{AX} = \mathbf{XD}$.

Autovetores e autovalores generalizados

Se **A** e **B** forem matrizes quadradas, então o comando

eig(A,B)

retorna um vetor contendo os autovalores generalizados que resolvem a equação

$$\mathbf{Ax} = \lambda \mathbf{Bx}$$

onde λ é um escalar. Os valores de λ que satisfazem à relação são os autovalores generalizados, e os valores correspondentes de **x** são os autovetores generalizados.

Para obtenção dos autovetores, deve ser utilizado o comando de atribuição dupla, como se segue:

[X,D] = eig(A,B)

Isto produz uma matriz diagonal **D** de autovalores generalizados e uma matriz quadrada **X** cujas colunas são autovetores correspondentes de forma que

$$\mathbf{AX} = \mathbf{BXD}$$

Por exemplo, se

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -6 & -4 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

então eig(A,B) dá como resultado

```
eig(A,B)

ans =

   0.3129 - 2.5087i
   0.3129 + 2.5087i
  -0.6258 - 0.0000i
```

e [X,D] = eig(A,B) resulta em

```
[X,D] = eig(A,B)

X =

   0.7309 + 0.0144i  -0.6720 + 0.2880i  -0.2390 + 0.5893i
  -0.0178 - 0.2503i  -0.0776 - 0.2387i   0.2459 - 0.6062i
  -0.6336 - 0.0336i   0.5745 - 0.2693i  -0.1539 + 0.3794i


D =

   0.3129 - 2.5087i        0                       0
         0           0.3129 + 2.5087i              0
         0                 0             -0.6258 - 0.0000i
```

Os autovetores são postos em escala, de maneira que a norma de cada um seja igual a 1,0.

Equação característica

As raízes da equação característica são idênticas aos autovalores da matriz **A**. A equação característica da matriz **A** é calculada através de

```
p = poly(A)
```

Por exemplo, se uma matriz **A** for igual a

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}$$

então o comando poly(A) resulta em

```
p = poly(A)

p =

    1.0000    6.0000   11.0000    6.0000
```

Esta é a representação no MATLAB do polinômio

$$s^3 + 6s^2 + 11s + 6 = 0$$

As raízes da equação característica $p = 0$ podem ser obtidas através do comando r = roots(p):

```
r = roots(p)

r =

   -3.0000
   -2.0000
   -1.0000
```

Estas raízes podem ser rearrumadas de novo no polinômio original com o comando q = poly(r).

```
q = poly(r)

q =

    1.0000    6.0000   11.0000    6.0000
```

Produto de polinômios

Sejam:

$$a(s) = s^2 - 20.6$$
$$b(s) = s^2 + 19.6s + 151.2$$

O produto dos polinômios é a convolução de seus coeficientes. O produto dos polinômios *a(s)* e *b(s)* pode ser obtido entrando com o comando c = conv(a,b).

```
a = [1   0   −20.6]; b = [1   19.6   151.2];
c = conv(a,b)

c =

  1.0e + 003 *

    0.0010    0.0196    0.1306   −0.4038   −3.1147
```

A representação do polinômio c(s) no MATLAB é mostrada a seguir:

$$c(s) = s^4 + 19.6s^3 + 130.6s^2 - 403.8s - 3114.7$$

Deconvolução (divisão de polinômios)

Para dividir o polinômio *c(s)* pelo polinômio *a(s)* pode-se empregar o comando de deconvolução [q,r] = deconv(c,a).

```
[q,r] = deconv(c,a)

q =

     1.0000   19.6000   151.2000

r =

     0     0     0     0     0
```

Cálculo de polinômios

Se *p* é um vetor cujos elementos são os coeficientes de um polinômio, organizados na ordem das potências decrescentes, então polyval(p,s) fornece o valor do polinômio avaliado em *s*. Por exemplo, para calcular o polinômio

$$p(s) = 3s^2 + 2s + 1$$

para $s = 5$, digite o comando

```
p = [3   2   1];

polyval(p,5)
```

que vai resultar em

```
ans =

    86
```

O comando polyvalm(p,A) avalia o polinômio *p* no âmbito matricial. Considere-se a matriz **J** seguinte:

$$\mathbf{J} = \begin{bmatrix} -2 + j2\sqrt{3} & 0 & 0 \\ 0 & -2 - j2\sqrt{3} & 0 \\ 0 & 0 & -10 \end{bmatrix}$$

O comando poly(J) dá a característica polinomial para a matriz **J**.

```
p = poly(J)

p =

    1.0000    14.0000    56.0000   160.0000
```

A expressão MATLAB para a característica polinomial de **J** é apresentada a seguir.

$$\text{poly}(\mathbf{J}) = \phi(\mathbf{J}) = \mathbf{J}^3 + 14\mathbf{J}^2 + 56\mathbf{J} + 160\mathbf{I}$$

onde **I** é a matriz identidade. Para a matriz

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}$$

o comando polyvalm(poly(J),A) avalia a função $\phi(\mathbf{A})$:

$$\phi(\mathbf{A}) = \mathbf{A}^3 + 14\mathbf{A}^2 + 56\mathbf{A} + 160\mathbf{I} = \begin{bmatrix} 154 & 45 & 8 \\ -48 & 66 & -3 \\ 18 & -15 & 84 \end{bmatrix}$$

A avaliação da expressão acima aparece na seguinte saída do MATLAB:

```
polyvalm(poly(J),A)

ans =

   154.0000    45.0000     8.0000
   -48.0000    66.0000    -3.0000
    18.0000   -15.0000    84.0000
```

Obtenção dos quadrados das entradas de um vetor *x*

Dado um vetor **x**, o comando x.^2 fornece um novo vetor onde cada elemento é o quadrado do correspondente em **x**. Por exemplo para

$$\mathbf{x} = [\,1 \quad 2 \quad 3\,]$$

o comando x.^2 resulta na seguinte saída MATLAB

```
x = [1   2   3];
x.^2

ans =

     1     4     9
```

Da mesma forma para o vetor **y**,

$$\mathbf{y} = [2 + 5j \quad 3 + 4j \quad 1 - j]$$

y.^2 produz o seguinte resultado:

```
y = [2+5*i   3+4*i   1-i];
y.^2

ans =

  -21.0000 + 20.0000i   -7.0000 + 24.0000i         0 - 2.0000i
```

Obtenção dos quadrados das entradas de uma matriz A

Para uma matriz **A**, o comando A.^2 produz uma nova matriz, cujos elementos são o quadrado de seus elementos correspondentes em **A** e **B**. Por exemplo, se tivermos as seguintes matrizes

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1+j & 2-2j \\ 3+4j & 5-j \end{bmatrix}$$

a execução de A.^2 e B.^2 vai resultar em

```
A = [1   2;3   4];
A.^2

ans =

      1     4
      9    16

B = [1+i   2-2*i;3+4*i   5-i];
B.^2

ans =

         0 +  2.0000i         0 -  8.0000i
  -7.0000 + 24.0000i   24.0000 - 10.0000i
```

Valores absolutos

A função abs(A) resulta em uma matriz composta pelos valores absolutos de cada elemento de A. Se **A** for uma matriz complexa, abs(A) retorna seu módulo complexo (magnitude):

$$\text{abs(A)} = \text{sqrt(real(A).}\hat{}\,2 + \text{imag(A).}\hat{}\,2)$$

A função angle(A) retorna os ângulos das fases, em radianos, dos elementos da matriz complexa **A**. Tais ângulos variam entre $-\pi$ e $+\pi$ radianos. Veja o exemplo abaixo.

```
A = [2+2*i   1+3*i;4+5*i   6-i];
abs(A)

ans =

    2.8284    3.1623

    6.4031    6.0828

angle(A)

ans =

    0.7854    1.2490
    0.8961   -0.1651
```

Magnitude e ângulo da fase de um número complexo

A magnitude e o ângulo da fase de um número complexo $z = x + iy = re^{i\theta}$ são dados por

```
r = abs(z)
theta = angle(z)
```

e o comando

```
z = r*exp(i*theta)
```

converte o número complexo z de volta ao seu formato original.

Exponencial de uma matriz

A função expm(A) gera a exponencial de uma matriz **A**, de dimensão $n \times n$. Isto é

$$\mathbf{expm(A)} = \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \frac{\mathbf{A}^3}{3!} + \cdots$$

Observe-se que a função transcendente é interpretada como a função matricial, se um "m" aparecer no final do nome da função, como em expm(A) ou em sqrtm(A).

Matrizes notáveis

Em MATLAB, as funções

```
ones(n)
ones(m,n)
ones(A)
zeros
```

geram matrizes especiais. A função ones(n) produz uma matriz de dimensão $n \times n$, cujos elementos são todos unitários. Já a função ones(m,n) produz uma matriz de dimensão $m \times n$, também com todos os seus elementos unitários. A função zeros(*m,n*) tem como resultado uma matriz de dimensão $m \times n$, com todos os elementos nulos, e zeros(A) produz uma matriz de elementos nulos com a mesma dimensão de A, exceto quando A for um escalar.

Matriz identidade

Muitas vezes é preciso entrar com uma matriz identidade **I** em programas MATLAB. O comando eye(n) fornece uma matriz identidade de dimensão $n \times n$. Isto é,

```
eye(5)

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Matriz diagonal

Sendo **x** um vetor, o comando diag(x) produz uma matriz diagonal com **x** na linha da diagonal principal. Por exemplo, para o vetor

```
x = [ones(1,n)]
```

`diag([ones(1,n)])` produz uma matriz identidade $n \times n$, conforme mostrado a seguir

```
diag([ones(1,5)])

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Se **A** for uma matriz quadrada, então diag(A) é um vetor cujos elementos são os da diagonal principal de **A**, e diag(diag(A)) é a matriz diagonal com os elementos de diag(A) aparecendo na linha da diagonal principal. Veja a saída MATLAB apresentada abaixo.

```
A = [1  2  3;4  5  6;7  8  9];
diag(A)

ans =

     1
     5
     9

diag(diag(A))

ans =

     1     0     0
     0     5     0
     0     0     9
```

Observe-se que diag(1:5) resulta em

```
diag(1:5)

ans =

     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
```

Já diag(0:4) tem como resultado

```
diag(0:4)

ans =

     0     0     0     0     0
     0     1     0     0     0
     0     0     2     0     0
     0     0     0     3     0
     0     0     0     0     4
```

Portanto diag(1:5) — diag(0:4) tem como resultado a matriz identidade.

É importante observar que diag(0,n) é completamente diferente de diag(0:n). A função diag(0,n) produz uma matriz de dimensão $(n+1)\times(n+1)$ com todos os seus elementos nulos. Veja como exemplo a seguinte saída MATLAB.

```
diag(0,4)

ans =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

1-4 MODELOS MATEMÁTICOS DE SISTEMAS LINEARES

O MATLAB tem comandos para transformar um modelo matemático de um sistema linear em outro modelo. Tais transformações, úteis para a solução de problemas de engenharia de controle, serão discutidas a seguir.

Função de transferência para espaço de estados

O comando

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

converte o sistema na forma de função de transferência

$$\frac{Y(s)}{U(s)} = \frac{\text{num}}{\text{den}} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + D$$

para a forma de espaço de estados.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x} + Du$$

De espaço de estados para função de transferência

Se o sistema tiver uma entrada e uma saída, o comando

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

produz a função de transferência *Y(s)/U(s)*.

Se o sistema possuir mais de uma entrada, deve-se usar o comando seguinte:

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

Tal comando converte o sistema representado em espaço de estados

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$y = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

para função de transferência

$$\frac{Y(s)}{U_i(s)} = i\text{th element of } [\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}]$$

Observe-se que o escalar 'iu' é um índice para as entradas do sistema e especifica qual das entradas deve ser usada para resposta.

Considere-se, por exemplo, o seguinte sistema, com duas entradas, u_1 e u_2.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0 \quad 0] \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Podem ser obtidas duas funções de transferência para este sistema. Uma relaciona a saída y com a entrada u_1, e a outra relaciona a saída y com a entrada u_2. (Ao considerar a entrada u_1, admite-se que u_2 é zero, e vice-versa.) A seguinte saída de um programa MATLAB ilustra o exposto.

```
A = [0  1;-2  -3];
B = [1  0;0  1];
C = [1  0];
D = [0  0];

[num,den] = ss2tf(A,B,C,D,1)

num =

     0     1     3


den =

     1     3     2

[num,den] = ss2tf(A,B,C,D,2)

num =

     0     0     1


den =

     1     3     2
```

Da saída apresentada pode-se tirar o seguinte:

$$\frac{Y(s)}{U_1(s)} = \frac{s + 3}{s^2 + 3s + 2}$$

e

$$\frac{Y(s)}{U_2(s)} = \frac{1}{s^2 + 3s + 2}$$

Expansão da função de transferência em frações parciais

Considere-se a função de transferência abaixo

$$\frac{B(s)}{A(s)} = \frac{\text{num}}{\text{den}} = \frac{b(1)s^n + b(2)s^{n-1} + \cdots + b(n)}{a(1)s^n + a(2)s^{n-1} + \cdots + a(n)}$$

onde $a(1)$ é diferente de 0, mas com alguns dos coeficientes $a(i)$ e $b(j)$ podendo ser zero.

Os vetores linha num e den especificam os coeficientes do numerador e do denominador da função de transferência. Isto é,

$$\text{num} = [b(1) \quad b(2) \quad \cdots \quad b(n)]$$

$$\text{den} = [a(1) \quad a(2) \quad \cdots \quad a(n)]$$

O comando

```
[r,p,k] = residue(num,den)
```

encontra os resíduos, os pólos, e os termos diretos da expansão em frações parciais da relação entre dois polinômios, $B(S)$ e $A(s)$. A expansão em frações parciais de $B(s)/A(s)$ é dada por

$$\frac{B(s)}{A(s)} = \frac{r(1)}{s - p(1)} + \frac{r(2)}{s - p(2)} + \cdots + \frac{r(n)}{s - p(n)} + k(s)$$

Como exemplo, considere-se a seguinte função de transferência:

$$\frac{B(s)}{A(s)} = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}$$

Para esta função,

```
num = [2 5 3 6]

den = [1 6 11 6]
```

O comando

```
[r,p,k] = residue(num,den)
```

fornece o seguinte resultado:

```
[r,p,k] = residue(num,den)

r =

   -6.0000
   -4.0000
    3.0000


p =

   -3.0000
   -2.0000
   -1.0000


k =

    2
```

Observe-se que os resíduos retornam em um vetor coluna r, os pólos em um vetor coluna p, e o termo direto em um vetor linha k. Esta é a representação em MATLAB da seguinte expansão em frações parciais de $B(s)/A(s)$:

$$\frac{B(s)}{A(s)} = \frac{2s^3 + 5s^2 + 3s + 6}{(s + 1)(s + 2)(s + 3)}$$

$$= \frac{-6}{s + 3} + \frac{-4}{s + 2} + \frac{3}{s + 1} + 2$$

O comando

[num,den] = residue(r,p,k)

onde r,p e k são dados na saída MATLAB mostrada anteriormente, converte a expansão em frações parciais de novo na razão polinomial $B(s)/A(s)$, conforme abaixo:

```
[num,den] = residue(r,p,k)

num =

    2.0000    5.0000    3.0000    6.0000


den =

    1.0000    6.0000   11.0000    6.0000
```

Conversão do modelo contínuo no tempo para o discreto

O comando

[G,H] = c2d(A,B,Ts)

onde Ts é o período de amostragem em segundos, converte o modelo em espaço de estados de contínuo no tempo para discreto, admitindo-se a existência de um extrapolador de ordem zero ZOH-zero-order-hold nas entradas. Isto é, com o comando acima,

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

é convertido para

$$\mathbf{x}(k+1) = \mathbf{Gx}(k) + \mathbf{Hu}(k)$$

Considere-se, por exemplo, o seguinte sistema contínuo no tempo:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -25 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

Um sistema equivalente, discreto no tempo, pode ser obtido por meio da execução do comando [G,H]=c2d(A,B,Ts). O período de amostragem Ts vale 0,05 s. Observe a seguinte saída MATLAB:

```
A = [0   1;-25   -4];
B = [0;1];
format long
[G,H] = c2d(A,B,0.05)

G =

   0.97088325381929   0.04484704238264
  -1.12117605956599   0.79149508428874


H =

   0.00116466984723
   0.04484704238264
```

A equação no espaço de estados do sistema equivalente é:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9709 & 0.04485 \\ -1.1212 & 0.7915 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.001165 \\ 0.04485 \end{bmatrix} u(k)$$

1-5 RESUMO DO LIVRO

Este livro trata do projeto de sistemas de controle utilizando o MATLAB. Em resumo, os assuntos abordados serão os seguintes: O Cap. 1 apresentou o material introdutório. Os Caps. 2 e 3 discutem a alocação de pólos e dos problemas do projeto de observadores com auxílio do MATLAB. Especificamente o Cap. 2 trata dos sistemas contínuos no tempo, enquanto que o Cap. 3 trata dos sistemas discretos no tempo. O Cap. 4 é sobre o projeto de sistemas de controle ótimo. Nele são resolvidos problemas de controle quadrático ótimo com o MATLAB. Tanto os sistemas contínuos no tempo quanto os discretos serão abordados. O controle de energia-mínima dos sistemas discretos também é discutido neste capítulo.