Apêndice A

Tutorial do Matlab

A.1 Introdução

O Matlab (de **Mat**rix **Lab**oratory) é um pacote matemático capaz de tratar matrizes, vetores e outras estruturas, com números reais ou complexos, da mesma forma que uma calculadora trata com valores escalares. Além disso, o Matlab possui recursos de programação estruturada que permitem a implementação de procedimentos mais complexos, além de visualização gráfica e simulação de sistemas dinâmicos (através do pacote integrado Simulink).

A janela básica para o uso do Matlab é a janela de comandos, identificada pelo *prompt* ">>". A partir dessa janela é possível acessar todos os recursos do programa.

Este tutorial apresenta apenas os conceitos mais básicos. A consulta à documentação do sistema ou ao *site* do fabricante (http://www.mathworks.com), bem como aos diversos livros já publicados sobre o pacote é indispensável para um aprendizado mais aprofundado.

Este tutorial considera a versão 6.5 e posteriores. Todos os recursos apresentados são suportados em versões posteriores, mas alguns recursos não estão disponíveis para versões mais antigas.

A.2 Operações básicas

A.2.1 Criando matrizes e vetores

Ao se digitar

```
>> a = 2
```

obtém-se a seguinte resposta:

```
a =


     2
```

Este comando cria a variável "`a`" com o valor 2. Para se criar uma matriz, deve-se colocar os valores entre colchetes, separados por espaço (para separar colunas em uma mesma linha) ou por "`;`" (para separar linhas), assim

```
>> A = [1 2 3;3 2 1;5 0 0]
```

produz

```
A =


     1     2     3

     3     2     1

     5     0     0
```

A variável "A" agora possui o valor matricial acima. Para evitar que o resultado do comando apareça na tela, basta terminá-lo com um ";".

```
>> b = [10;5;3];
```

cria um vetor coluna, mas não imprime o resultado na janela de comandos. Isso é bastante útil caso o vetor ou matriz seja muito grande. Um exemplo de como criar vetores muito grandes é apresentado a seguir.

```
>> t = 0:0.01:10;
```

O comando acima cria um vetor em "t" da forma

```
[ 0 0.01 0.02 0.03 . . . 9.99 10 ].
```

Vetores e matrizes podem também ser facilmente criados a partir de outros vetores e matrizes. Por exemplo,

```
>> y = sin(t);
```

cria um vetor "y" contendo o seno dos elementos (supostos em radianos) do vetor "t". Note que a função "sin" (seno trigonométrico) de um vetor resulta num novo vetor com o seno de cada elemento do vetor original.

VEJA TAMBÉM

Os comandos "who" e "whos" fornecem uma lista das variáveis criadas. Variáveis podem ser apagadas com o comando "clear".
Os comandos "linspace" e "logspace" são também bastante úteis para a criação de vetores.
Para saber mais sobre um comando, basta digitar "help <*comando*>" ou "doc <*comando*>".
O Matlab também trabalha com *strings*.
`>> s = 'abc'`
cria uma variável *string*. O sinal " ' " delimita *strings*.
As variáveis criadas podem ser salvas em um arquivo e recuperadas posteriormente com o uso dos comandos "save" e "load".
Para recuperar comandos já digitados, use as setas para cima e para baixo do teclado (ou o histórico de comandos).

A.2.2 Cálculos básicos

Os operadores básicos "+", "–", "*", "^", etc. operam matricialmente, portanto

a) somente é possível adicionar e subtrair variáveis de mesmas dimensões;

b) somente é possível multiplicar duas variáveis se o número de colunas da primeira for igual ao número de linhas da segunda;

c) somente é possível exponenciar matrizes quadradas. (com a função "expm").

Uma exceção é s soma de uma matriz com um número escalar, assim

```
>> r = [1 2 3] + 10
```

é uma operação permitida e resulta em

```
r =


    11    12    13
```

Os operadores "/" e "\" são ligeiramente mais complexos e podem ser entendidos como sendo divisão à direita e divisão à esquerda.

`x = A\b` é a solução do sistema `A*x = b`

`x = A/B` é a solução do sistema `x*A = B`

de modo que

```
>> x = A\b
```

fornece

```
x =


    0.6000
    0.0500
    3.1000
```

De fato

```
>> A*x-b
```

fornece o resultado esperado:

```
ans =


     0
     0
     0
```

Note que como não atribuímos o resultado desta operação a nenhuma variável, o Matlab automaticamente criou a variável "`ans`" (de *answer*).

Há também operadores que operam sobre cada elemento de um vetor ou matriz, por exemplo: "`.*`", "`./`", "`.^`" e boa parte das funções, como "`sin`", "`cos`", "`log`", "`sqrt`" (raiz quadrada), "`exp`", etc.

```
>>( [1 2 3] ).^2
```

resulta em

```
ans =


     1     4     9
```

.

VEJA TAMBÉM
Algumas funções que operam matricialmente são "`det`" (determinante), "`inv`" (matriz inversa), "`sqrtm`", "`logm`", "`expm`" (versões matriciais das funções já vistas).
Note que "`log`" é o logaritmo neperiano. O logaritmo base 10 é denotado por "`log10`".

A.2.3 Números complexos e aritmética IEEE

O Matlab opera naturalmente com números complexos. O comando

```
>> f = sqrt(-2)
```

fornece

```
f =


         0 + 1.4142i
```

Para facilitar a entrada de valores complexos, o Matlab fornece duas funções: "`i`" e "`j`", ambas iguais a $\sqrt{-1}$. Assim

```
>> g = 2 + 3*i
```

resulta em (poderíamos também ter usado "`j`")

```
g =


   2.0000 + 3.0000i
```

IMPORTANTE: Caso sejam criadas variáveis "`i`" ou "`j`" pelo usuário, as funções originais deixam de ser acessíveis.

O Matlab utiliza o modelo aritmético dado pelo padrão ANSI/IEEE 754-1985. Uma característica importante desse padrão é a introdução dos conceitos de "infinito" e de "não-número" (para valores indefinidos). O comando

```
>> w = 1/0
```

não produz uma mensagem de erro, apenas um aviso (ou não, conforme a versão do Matlab) e o resultado "infinito".

```
Warning: Divide by zero.


w =


   Inf
```

Ao passo que o comando abaixo produz o resultado "não-número".

```
>> q = 0/0
Warning: Divide by zero.

q =

   NaN
```

O uso desses conceitos é bastante conveniente, especialmente em seqüências complexas de cálculos, pois o Matlab não interrompe operações que podem eventualmente resultar válidas mesmo que um resultado intermediário divirja, como por exemplo:

```
>> 1/w + 10

ans =

    10
```

(lembrando que $1/\infty \to 0$)..

> **VEJA TAMBÉM**
> As funções "`inf`" e "`nan`" permitem criar diretamente valores na situação "infinito" ou "não-número".

A.3 Manipulação de matrizes e vetores

A.3.1 Operações com elementos de vetores ou matrizes

É possível acessar diretamente valores de matrizes ou vetores, bem como alterá-los ou eliminar submatrizes de uma matriz. A forma básica de se referir a um elemento de uma matriz é colocando os índices (linha e coluna) entre parênteses, separados por vírgula, ou seja:

```
>> A(2,3)
```

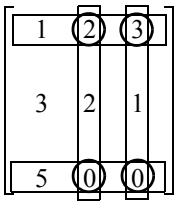
como a matriz "`A`" é dada por $\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & \textcircled{1} \\ 5 & 0 & 0 \end{bmatrix}$ — coluna 3, linha 2

temos

```
ans =

    1
```

Podemos também extrair submatrizes da matriz "`A`". Basta colocar vetores nos índices.

```
>> A([1 3],[2 3])
```

ou seja 

, resultando em

```
ans =


     2     3
     0     0
```

```
>> A(:,2)
```

O símbolo "`:`" nesse caso pode ser entendido como "todos os elementos", resultando em

```
ans =


     2
     2
     0
```

Poderíamos substituir "`:`" por `1:3`

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

gerando o mesmo resultado.

Para eliminar uma linha, coluna ou submatriz, podemos proceder da seguinte forma:

```
>> A(3,:) = []
```

O símbolo "`[]`" pode ser lido como "matriz vazia", tendo como efeito eliminar a terceira linha da matriz, resultando em

```
A =


     1     2     3
     3     2     1
```

A.3.2 Concatenação

É possível também concatenar matrizes e vetores, de modo que para acrescentar uma nova linha à matriz "A", basta o seguinte comando.

```
>> A = [ A ; [10 1 11] ]
A =


     1     2     3
     3     2     1
    10     1    11
```

e

```
>> B = [A A]
```

resulta em

```
B =


     1     2     3     1     2     3
     3     2     1     3     2     1
    10     1    11    10     1    11
```

.

VEJA TAMBÉM

Todos os exemplos deste tutorial usam matrizes bidimensionais, porém o Matlab também tem capacidade de tratar matrizes *n*-dimensionais com comandos específicos.
No caso mais simples em que a matriz se resume a um vetor, isto é, possui uma única linha ou coluna, podemos referenciá-lo com um único índice, ou seja

`A(1,n)` → `A(n)`

`A(m,1)` → `A(m)`

Um comando extremamente útil para se localizar valores em um vetor ou matriz é o comando "`find`". Procure saber mais sobre este comando.

A.4 Polinômios

Uma série de comandos e funções do Matlab, particularmente úteis, tratam com polinômios, conforme veremos a seguir.

A.4.1 Codificando polinômios em vetores

A codificação é extremamente simples, basta colocar os coeficientes de cada grau do polinômio em ordem decrescente (incluindo todos os graus, mesmo que tenham coeficientes nulos) em um vetor. Por exemplo:

```
>> poli1 = [1 4 2 0 20];
```

equivale a $x^4 + 4x^3 + 2x^2 + 20$.

```
>> poli2 = [5 7 0 0];
```

equivale a $5x^3 + 7x^2$.

A.4.2 Operações com polinômios

Para somar ou subtrair polinômios é necessário ajustar as dimensões e então somá-los ou subtraí-los como vetores comuns.

```
>> soma = poli1 + [ 0 poli2 ]
```

É necessário concatenar um zero ao polinômio "`poli2`" para que ambos tenham a mesma dimensão, resultando em

```
soma =


     1     9     9     0    20
```

que é equivalente a $x^4 + 9x^3 + 9x^2 + 20$.

Para realizar o produto de polinômios (ou convolução), basta usar a função "conv".

```
>> prod = conv(poli1,poli2)


prod =


     5    27    38    14   100   140     0     0
```

equivalente a $5x^7 + 27x^6 + 38x^5 + 14x^4 + 100x^3 + 140x^2$.

Para extrair as raízes do polinômio acima, usamos a função "roots".

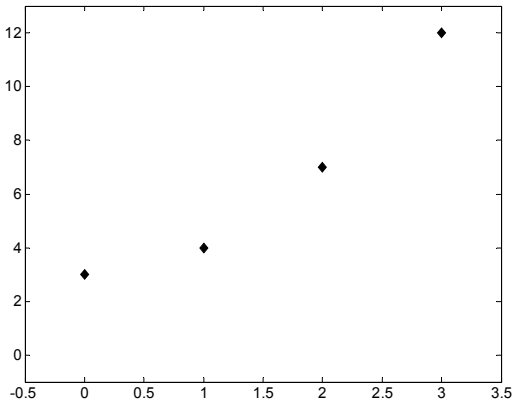
```
>> roots(prod)


ans =


         0
         0
   0.7881 + 1.3254i
   0.7881 - 1.3254i
  -2.7881 + 0.7988i
  -2.7881 - 0.7988i
  -1.4000
```

O comando apresenta todas as raízes, incluindo as raízes múltiplas e as complexas..

> **VEJA TAMBÉM**
> Divisão polinomial pode ser feita com o uso da função "deconv".

A.4.3 Ajuste de curva com polinômios

Suponha que queiramos ajustar os pontos do gráfico ao lado (feito com o Matlab, como veremos no próximo item) a um polinômio de determinada ordem.



Inicialmente criamos os vetores com os dados.

```
>> x = [0 1 2 3];
>> y = [3 4 7 12];
```

Usamos então o comando "polyfit" para ajustar um polinômio de 3ª. ordem.

```
>> p3 = polyfit(x,y,3)
p3 =


    0.0000    1.0000    0.0000    3.0000
```

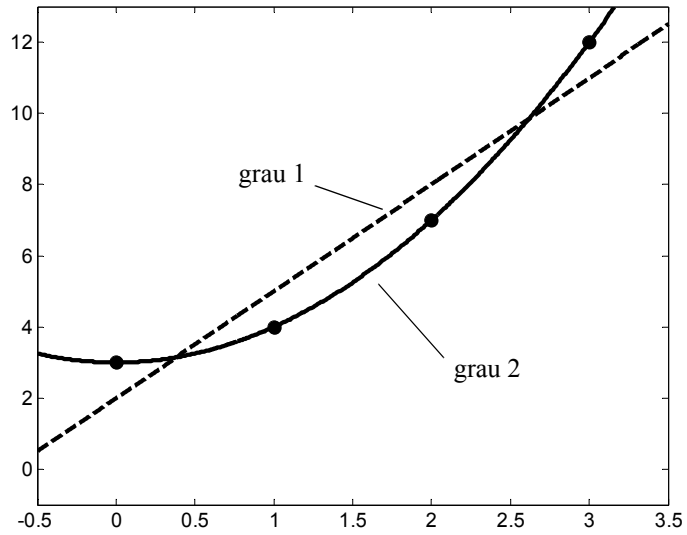
equivalente a $x^2 + 3$. O resultado nos sugere que um polinômio de grau 2 seria suficiente, assim

```
>> p2 = polyfit(x,y,2)
p2 =


    1.0000    0.0000    3.0000
```

Caso quiséssemos ajustar uma reta aos pontos dados, bastaria usar um polinômio de grau 1.

```
>> p1 = polyfit(x,y,1)
p1 =


    3.0000    2.0000
```

Os polinômios obtidos ajustam os dados conforme a figura abaixo.



.

VEJA TAMBÉM

O comando "`polyval`" permite calcular um dado polinômio em pontos escolhidos.

O Matlab também possui recursos de interpolação. Veja as funções "`interp1`" e "`interp2`".

A.5 Gráficos

O Matlab possui recursos gráficos extremamente sofisticados, porém neste tutorial somente os recursos mais simples serão explorados.

A.5.1 Criando um gráfico simples

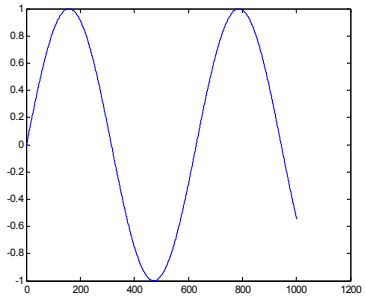
O comando mais simples para se gerar um gráfico é o comando "plot". Inicialmente vamos gerar alguns vetores.

```
>> t = 0:0.01:10;
>> y = sin(t);
>> z = 0.5*cos(t);
>> w = y.*z;
```

O comando

```
>> plot(y)
```

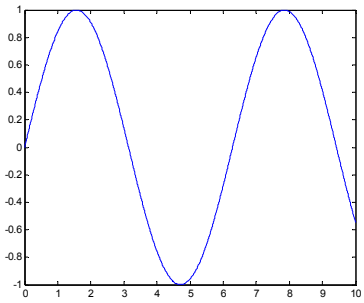
traça um gráfico com o vetor "y" na ordenada e os índices do vetor (indo de 1 até 1001), conforme a figura ao lado. Esta é a forma mais simples de se gerar um gráfico. Mais interessante seria traçar um gráfico do tipo $t \times y$.



Isso pode ser feito utilizando-se a forma

```
>> plot(t,y)
```

que gera um gráfico como este ao lado.



É possível traçar gráficos com diversas curvas. O comando

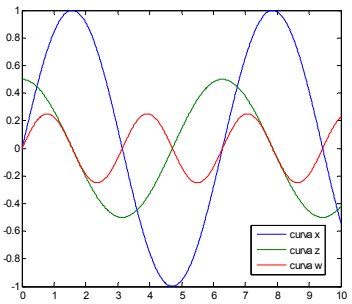
```
>> plot(t,y,t,z,t,w)
```

traça três curvas no mesmo gráfico. Para identificá-las, o comando "`legend`" é bastante útil.

```
>> legend('curva x','curva z','curva w')
```

identifica as curvas com uma legenda. Para mover a caixa de legendas para um lugar mais conveniente basta arrastá-la com o *mouse*.

O resultado final está no gráfico ao lado.



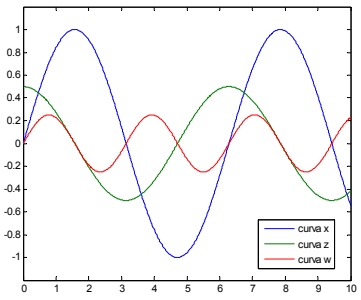
Para melhorar a aparência do gráfico podemos alterar a escala do gráfico, usando a função "axis". O argumento da função é um vetor de quatro elementos, do tipo

$$\left[x_{min}\ x_{max}\ y_{min}\ y_{max}\right]$$

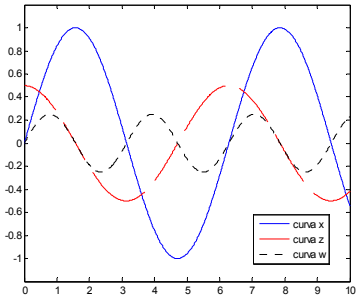
onde basta escolher os valores desejados ou "`-inf`" ou "`inf`" para deixar isso a cargo do Matlab. Assim

```
>> axis([-inf inf -1.2 1.2])
```

deixa o gráfico mais apresentável (ao lado).



Podemos alterar o tipo ou a cor das linhas diretamente no momento de traçar o gráfico. O Matlab possui diversas opções, para saber quais são consulte a documentação do programa. O comando a seguir produz uma linha azul (**b**lue) continua para a primeira curva, uma linha tracejada vermelha (**r**ed) para a segunda curva e uma linha pontilhada preta (blac**k**) para a terceira.

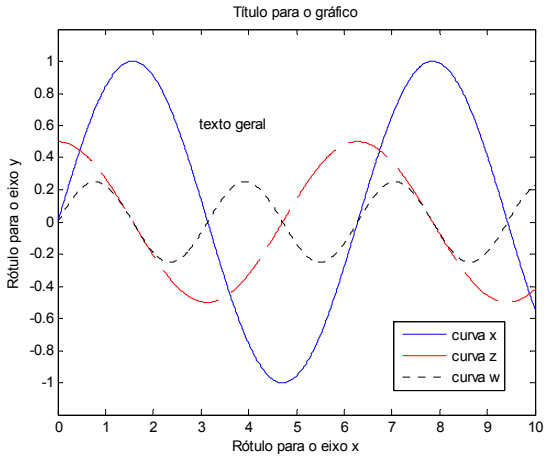



```
>> plot(t,y,'b-',t,z,'r--',t,w,'k:')
```

Podemos também incluir títulos, rótulos e textos diversos, utilizando os comandos

```
>> title('Título para o gráfico')
>> xlabel('Rótulo para o eixo x')
>> ylabel('Rótulo para o eixo y')
>> gtext('texto geral')
```

resultando em



.

VEJA TAMBÉM

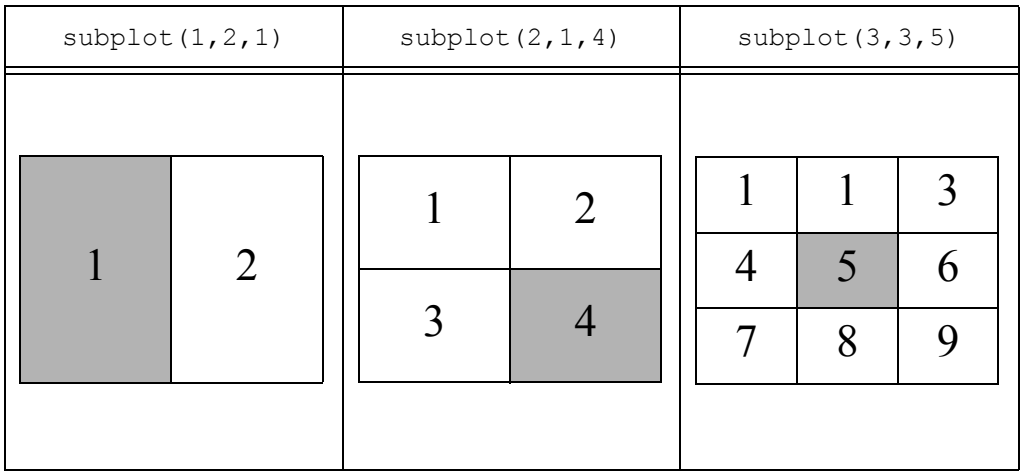
Gráficos com escalas logarítmicas podem ser produzidos com os comandos "`semilogx`", "`semilogy`" e "`loglog`".
Para sobrepor gráficos, basta utilizar o comando "`hold on`". Os comandos a seguir farão com que os novos gráficos sejam traçados sobre os anteriores. "`hold off`" desativa a função.
Da mesma forma, "`grid on`" e "`grid off`" servem para colocar grades nos gráficos.
Também é possível traçar os pontos sem ligá-los, usando símbolos como "o" ou "+".
Novas janelas gráficas podem ser criadas com o comando "`figure`".

A.5.2 Dividindo os gráficos

Os gráficos podem ser subdivididos com o uso do comando "`subplot`". O comando possui a seguinte sintaxe.

`subplot(l,c,n)`

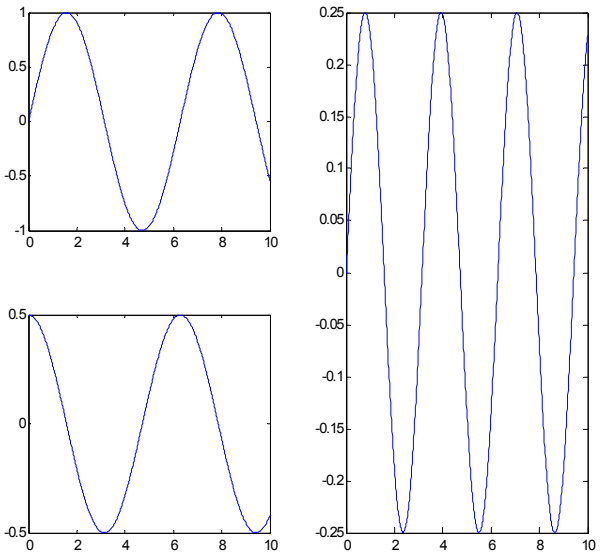
Este comando divide a janela em uma matriz de `l` linhas e `c` colunas, e ativa o `n`-ésimo elemento dessa matriz. Por exemplo:



Divisões diferentes podem coexistir. Os comandos

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

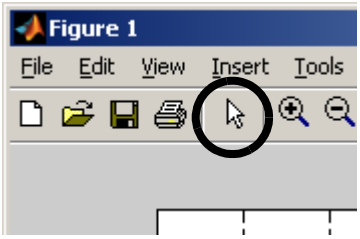
produzem o resultado do gráfico ao lado. Todos os comandos que se aplicam aos gráficos (como "`title`", "`xlabel`", etc. podem ser usados nos subgráficos.



A.5.3 Propriedades avançadas dos gráficos

Todos os comandos vistos até o momento podem ser executados a partir da interface gráfica através de formulários e menus. Além disso, tarefas mais complexas, tais como alterar um tipo de linha, sua cor ou espessura podem ser realizados sem a necessidade de se refazer o gráfico. Outros recursos como alterar o tipo ou o tamanho dos caracteres, etc. também podem ser realizados através desta interface.

Para manipular os elementos de um gráfico, basta colocá-lo no modo de edição, o que é feito ativando o ícone de edição, localizado na janela gráfica.



.

VEJA TAMBÉM

O Matlab define a janela gráfica como um objeto, que possui propriedades que podem ser manipuladas pelo usuário. Não faz parte do escopo deste tutorial a manipulação de gráficos neste nível, porém o leitor interessado poderá consultar a documentação do Matlab a este respeito.

A.6 O toolbox de controle

O Matlab possui uma biblioteca de funções específicas para projeto e análise de sistemas de controle lineares, o "Control System Toolbox". Para mais informações basta digitar

```
>> help control
```

ou

```
>> doc control
```

Dentre as principais funções, pode-se citar "`impulse`", "`step`", "`bode`", "`rlocus`", entre outras.

Um aspecto importante do toolbox de controle é o objeto "Linear Invariante no Tempo" (LTI). Ele permite armazenar o modelo completo de um sistema linear invariante no tempo, assim como diversas de suas propriedades, em uma única variável. Além disso, diversas operações envolvendo sistemas podem ser realizadas diretamente com objetos LTI. Este é um tópico complexo demais para este tutorial, mas para sistemas descritos por funções de transferência (que são a regra neste curso), podemos criar um objeto LTI com o seguinte comando

```
>> sis = tf(num,den)
```

onde `num` e `den` são o numerador e denominador da função de transferência, descritos no padrão de polinômios, como visto acima. Definido `sis`, as diversas funções do toolbox podem ser utilizadas, tais como

```
>> step(sis); bode(sis)
```

dentre outras.

Com esses objetos pode-se fazer diversas operações. Por exemplo

```
>> sisp = sis1 + sis2
```

cria um sistema `sisp` com a soma das saídas dos sistemas `sis1` e `sis2` para uma mesma entrada (isso equivale a implementar `sis1` e `sis2` em paralelo num diagrama de blocos).

```
>> siss = sis1*sis2
```

cria um sistema `siss` com composição em série dos sistemas `sis1` e `sis2` (isso equivale a implementar `sis1` e `sis2` em sequência num diagrama de blocos)..

VEJA TAMBÉM

O toolbox de sistemas de controle é uma ferramenta extremamente poderosa para projeto e análise de sistemas de controle como os considerados neste curso. Recomenda-se um estudo detalhado deste toolbox e de sua documentação.

A.7 Programação com o Matlab

Diversos comandos e funções não são codficados internamente no próprio Matlab, e estão definidos em arquivos de texto denominados *M-files,* escritos em uma linguagem de programação própria, denominada de "M". A função "bode" por exemplo, é uma rotina cujo conteúdo pode ser visto com o comando

```
>> type bode
```

Os *M-files* podem ser de dois tipos: *functions* e *scripts*. *Scripts* são simplesmente seqüências de comandos, como acima, dispostos em um arquivo texto com extensão ".m". *Functions* são estruturas mais complexas, que apresentam certas características básicas, que são as seguintes: i) Existe passagem de parâmetros entre o *workspace* do Matlab (*workspace* ou espaço de trabalho pode ser entendido a grosso modo como o conjunto de variáveis que pode ser observado com o comando `whos` na janela de comandos) e esta é a única troca de informações. Por exemplo, o comando `y=cos(x)` passa à função `cos` um argumento `x` e recebe como resposta `y`. Não há além disso nenhuma troca de informações. ii) O ambiente de cada função é independente, isto é, uma função não reconhece variáveis definidas fora dela, e outras funções

ou o *workspace* não reconhecem variáveis definidas nela. Programadores de Pascal perceberão uma analogia clara entre *scripts* e *functions* do Matlab com *procedures* e *functions* de Pascal.

Um exemplo simples de função é o arquivo log10.m, que é uma função escrita para calcular logaritmos base 10 a partir de logaritmos naturais. O comando

```
>> type log10
```

mostra o conteúdo deste arquivo. Analise-o e entenda o seu funcionamento.

Espera-se que você seja capaz de escrever seus próprios *scripts* e *functions* para automatizar tarefas repetitivas e eventualmente realizar algumas mais complexas. A linguagem de programação do Matlab é muito semelhante a outras linguagens estruturadas, e se você estiver familiarizado com qualquer uma delas não terá problemas em fazer programas para o Matlab. Lembre de usar somente o seu diretório para armazenar programas e dados e de documentar bem os seus programas. É importante que você crie um cabeçalho inicial de comentários, pois o que o comando `help` faz nada mais é do que apresentar na tela o conteúdo deste cabeçalho. O comando

```
>> help lang
```

ou

```
>> doc lang
```

apresenta basicamente tudo o que é necessário saber para se iniciar em programação Matlab.

A.8 Comentário final

Tenha em mente que o número de comandos e funções do Matlab, numa instalação normal, é usualmente da ordem de milhares, portanto é quase impossível memorizá-los todos, e é óbvio que este tutorial apresenta somente o mínimo necessário para se ter alguma desenvoltura com o Matlab. Use os comandos `doc`, `help` e `demo` sempre que estiver em dúvida sobre como fazer algo.