1

Visão geral do MATLAB®[1]

Este é o capítulo mais importante do livro. Assim que você terminá-lo, será capaz de utilizar o MATLAB para resolver muitos tipos de problemas. A Seção 1.1 apresenta uma introdução ao MATLAB como uma calculadora interativa. A Seção 1.2 aborda os principais menus e a barra de ferramentas. A Seção 1.3 introduz arranjos, arquivos e plotagens. A Seção 1.4 discute como criar, editar e salvar programas no MATLAB. A Seção 1.5 introduz o extensivo Sistema de Ajuda do MATLAB e a Seção 1.6 introduz a metodologia de resolução de problemas de engenharia.

Como utilizar este livro

A organização de capítulos do livro é flexível o suficiente para acomodar uma diversidade de usuários. Entretanto, é importante que se abordem os primeiros capítulos na ordem em que estão dispostos. O Capítulo 2 aborda *arranjos*, que são os blocos de construção básicos no MATLAB. O Capítulo 3 aborda utilização de arquivos, funções internas do MATLAB e funções definidas pelo usuário. O Capítulo 4 aborda programação utilizando operadores relacionais e lógicos, sentenças condicionais e laços.

Os capítulos de 5 a 11 são independentes e podem ser abordados em qualquer ordem. Eles contêm discussões aprofundadas sobre como utilizar o MATLAB para solucionar diversos tipos comuns de problemas. O Capítulo 5 aborda plotagens bidimensionais e tridimensionais em mais detalhes. O Capítulo 6 mostra como utilizar plotagens para construir modelos matemáticos a partir de dados. O Capítulo 7 aborda aplicações de probabilidade, estatística e interpolação. O Capítulo 8 trata de equações algébricas lineares de maneira mais aprofundada por meio do desenvolvimento de métodos para os casos indeterminado e sobredeterminado. O Capítulo 9 introduz métodos

[1] MATLAB é uma marca registrada da empresa The MathWorks, Inc.

numéricos para cálculo e equações diferenciais ordinárias. O Simulink®[2], que é o tópico do Capítulo 10, é uma interface gráfica do usuário para resolver modelos de equações diferenciais. O Capítulo 11 aborda processamento simbólico com o MuPAD®[2], uma nova funcionalidade do toolbox de Matemática Simbólica do MATLAB, com aplicações em álgebra, cálculo, equações diferenciais, transformadas e funções especiais.

Subsídios de referência e aprendizado

Este livro foi idealizado para ser tanto uma referência quanto uma ferramenta de aprendizado. As características especiais úteis para esses propósitos são apresentadas a seguir.

- Notas à margem identificam onde os novos termos foram introduzidos.
- Ao longo de cada capítulo, aparecem pequenos exercícios do tipo Teste seus conhecimentos. Quando apropriado, as respostas são apresentadas imediatamente após os exercícios, de modo que você possa mensurar o seu domínio do assunto.
- Exercícios de dever de casa são encontrados ao final de cada capítulo. Eles normalmente requerem um esforço maior do que os exercícios do tipo Teste seus conhecimentos.
- Cada capítulo contém tabelas que resumem os comandos do MATLAB nele introduzidos.
- No final de cada capítulo encontra-se
  - Um resumo do que você deve ser capaz de fazer após terminar o capítulo.
  - Uma lista de termos-chave que você deve conhecer.
- O Apêndice A contém tabelas dos comandos do MATLAB, agrupados por categorias, com as páginas de referência apropriadas.
- O índice apresenta quatro partes: símbolos do MATLAB, comandos do MATLAB, blocos do Simulink e tópicos.

1.1 Sessões interativas do MATLAB

Agora mostraremos como iniciar o MATLAB, como fazer alguns cálculos básicos e como sair do MATLAB.

[2] Simulink e MuPAD são marcas registradas da empresa The MathWorks, Inc.

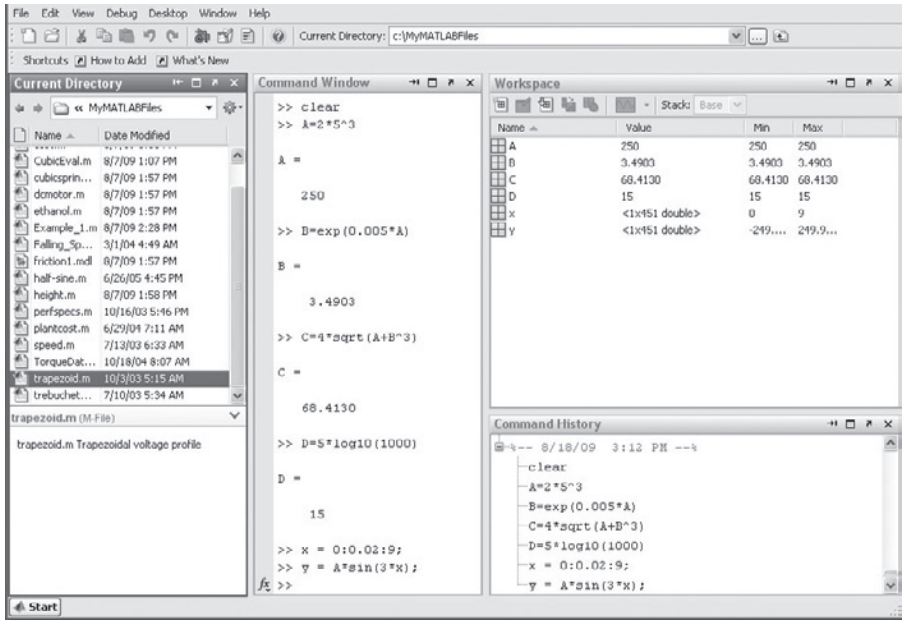

FIGURA 1.1-1 A Área de Trabalho padrão do MATLAB.

Convenções†

Neste livro utilizamos `fonte de máquina de escrever` para representar comandos do MATLAB, qualquer texto que você digitar no computador e qualquer resposta do MATLAB que aparecer na tela, por exemplo, `y = 6*x`. Variáveis no texto matemático normal aparecem em itálico, por exemplo, $y = 6x$. Utilizamos negrito para três finalidades: representar vetores e matrizes em texto matemático normal (por exemplo, $\mathbf{Ax = b}$); representar uma tecla do teclado (por exemplo, **Enter**); e representar o nome de um menu ou de um item que aparecer em tal menu (por exemplo, **File**). Supomos que você sempre aperta **Enter** após digitar um comando. Não representaremos essa ação com um símbolo separado.

Iniciando o MATLAB

ÁREA DE TRABALHO

Para iniciar o MATLAB em um sistema MS Windows, clique duas vezes no ícone do MATLAB. Você então verá a *Área de Trabalho* (*Desktop*) do MATLAB. A Área de Trabalho gerencia a janela de Comandos (Command window), o Navegador de

† N. de R. T.: No Brasil, a separação de casas decimais é feita por vírgula. Entretanto, o padrão utilizado no MATLAB é o inglês (ou seja, separação de casas decimais por ponto). Assim, quando ocorrerem casas decimais, elas devem ser separadas por um ponto na digitação de números no MATLAB (seja na Command Window, seja no desenvolvimento de programas). Nas explicações ao longo do texto, será utilizada a notação brasileira, ou seja, a separação de casas decimais por vírgulas. Obviamente, nos códigos apresentados ao longo do texto, os números estarão separados por ponto, concordando com a notação do MATLAB. O aluno deve ficar atento a isso.

Ajuda (Help Browser) e outras ferramentas. A aparência padrão da Área de Trabalho é apresentada na Figura 1.1-1. Aparecem cinco janelas: a janela de Comandos ao centro, a janela de Histórico de Comandos (Command History) à direita inferior, a janela do Espaço de Trabalho (Workspace) à direita superior, a janela de Detalhes (Details window) à esquerda inferior e a janela de Diretório Atual (Current Directory) à esquerda superior. Na parte superior da Área de Trabalho há uma linha de nomes de menu e uma linha de ícones chamada de *barra de ferramentas* (*toolbar*). À direita da barra de ferramentas há uma caixa que mostra o diretório onde o MATLAB busca e salva arquivos. Nós descreveremos os menus, a barra de ferramentas e os diretórios mais adiante neste capítulo.

JANELA DE COMANDOS

Você utiliza a janela de Comandos para se comunicar com o MATLAB digitando diferentes tipos de instruções: *comandos*, *funções* e *sentenças*. Mais adiante, discutiremos as diferenças entre eles, mas por enquanto, para simplificar a discussão, de maneira genérica chamaremos as instruções de *comandos*. O MATLAB exibe o prompt (≫) para indicar que ele está pronto para receber instruções. Antes de dar instruções ao MATLAB, certifique-se de que o cursor esteja localizado logo após o prompt. Se não estiver, utilize o mouse para mover o cursor. O prompt na Edição de Estudante (*Student Edition*) tem o seguinte aspecto: `EDU ≫`. Utilizaremos o símbolo normal do prompt (≫) para ilustrar os comandos neste texto. A janela de Comandos na Figura 1.1-1 mostra alguns comandos e os resultados dos cálculos. Nós abordaremos esses comandos mais adiante neste capítulo.

Quatro outras janelas aparecem na Área de Trabalho padrão. A janela de Diretório Atual se parece com uma janela de gerenciador de arquivos; você pode usá-la para acessar arquivos. Ao clicar duas vezes no nome de um arquivo com a extensão .m, aquele arquivo será aberto no Editor do MATLAB. O Editor será discutido na Seção 1.4. A Figura 1.1-1 mostra os arquivos no diretório do autor C:\MyMATLABFiles.

Abaixo da janela de Diretório Atual está a janela . . . . Ela exibe quaisquer comentários existentes no arquivo. Note que dois tipos de arquivo são mostrados no Diretório Atual. Eles possuem as extensões .m e .mdl. Abordaremos arquivos M neste capítulo. O Capítulo 10 abordará o Simulink, que utiliza arquivos MDL. Você pode ter outros tipos de arquivo no diretório.

A janela do Espaço de Trabalho aparece à direita superior e exibe as variáveis criadas na janela de Comandos. Clique duas vezes no nome de uma variável para abrir o Editor de Arranjos (Array Editor), que será discutido no Capítulo 2.

A quinta janela na Área de Trabalho padrão é janela de Histórico de Comandos. Essa janela mostra todos os caracteres que você inseriu previamente na janela de Comandos. Ela é útil para rastrear tudo o que você digitou. Você pode clicar em algum comando que já digitou e arrastá-lo para a janela de Comandos ou do Editor para evitar digitá-la novamente. O duplo clique em algo já digitado o faz ser executado na janela de Comandos.

Você pode alterar a aparência da Área de Trabalho, se desejar. Por exemplo, para eliminar uma janela, apenas clique no botão fecha janela (×) no canto direito superior. Para desencaixar ou separar uma janela da Área de Trabalho, clique no botão que contém uma seta curva. Uma janela desencaixada pode ser levada para qualquer lugar na tela. Você pode manipular outras janelas do mesmo modo. Para restaurar a configuração padrão, clique no menu **Desktop**, depois clique em **Desktop Layout** e selecione **Default**.

Inserindo comandos e expressões

Para ver como é simples utilizar o MATLAB, tente inserir alguns comandos no seu computador. Se você cometer algum erro de digitação, apenas pressione **Enter** até aparecer o prompt e redigite a linha. Ou, uma vez que o MATLAB retém seus toques de tecla anteriores em um arquivo de comandos, você pode usar a tecla direcional superior (↑) para rever os comandos já inseridos. Aperte a tecla uma vez para ver a última inserção, duas para ver a penúltima inserção, e assim por diante. Utilize a tecla direcional inferior (↓) para caminhar para frente ao longo da lista de comandos já inseridos. Quando você encontrar a linha desejada, você pode editá-la utilizando as teclas direcionais esquerda e direita (← e →) e as teclas **Backspace** e **Delete**. Pressione **Enter** para executar o comando. Essa técnica permite corrigir rapidamente erros de digitação.

Note que você pode visualizar seus toques de tecla anteriores exibidos na janela de Histórico de Comandos. Você pode copiar uma linha dessa janela para a janela de Comandos selecionando a linha com o mouse, mantendo o botão esquerdo do mouse pressionado e arrastando a linha para a janela de Comandos.

Certifique-se de que o cursor esteja no prompt da janela de Comandos. Para dividir 8 por 10, digite `8/10` e pressione **Enter** (o símbolo / é o símbolo do MATLAB para divisão). Sua inserção e a resposta do MATLAB aparecerão da seguinte maneira na tela (chamamos essa interação entre você e o MATLAB de *sessão interativa*, ou simplesmente de *sessão*). Lembre-se de que o símbolo >> aparece automaticamente na tela; você não o digita.

SESSÃO

```
>> 8/10
ans =
     0.8000
```

O MATLAB indenta o resultado numérico. Ele utiliza uma precisão alta para os seus cálculos, mas por padrão ele normalmente exibe quatro casas decimais, exceto quando o resultado é um inteiro.

O MATLAB atribui a resposta mais recente a uma variável chamada `ans`, que é uma abreviação de *answer* (resposta). Uma *variável* no MATLAB é um símbolo utilizado para conter um valor. Você pode utilizar a variável `ans` para cálculos posteriores; por exemplo, utilizando o símbolo do MATLAB para multiplicação (*), obtemos

VARIÁVEL

```
>> 5*ans
ans =
      4
```

Note que a variável `ans` agora possui o valor 4.

Você pode utilizar variáveis para escrever expressões matemáticas. Em vez de utilizar a variável padrão `ans`, você pode atribuir o resultado a uma variável de sua própria escolha, por exemplo, a variável `r`, como apresentado a seguir:

```
>> r=8/10
r =
   0.8000
```

Espaços na linha melhoram a legibilidade; por exemplo, você pode colocar um espaço antes e depois do sinal =, se você quiser. O MATLAB ignora esses espaços quando realiza os seus cálculos. Ele também ignora espaços em torno dos sinais + e −.

Se você agora digitar `r` no prompt e pressionar **Enter**, você verá

```
>> r
r =
    0.8000
```

e verificará, portanto, que a variável `r` contém o valor 0,8. Você pode utilizar essa variável em cálculos posteriores. Por exemplo,

```
>> s=20*r
s =
    16
```

Um erro comum é esquecer o símbolo de multiplicação * e digitar a expressão como se faria em álgebra, como $s = 20r$. Se você fizer isso no MATLAB, ele retornará uma mensagem de erro.

O MATLAB possui centenas de funções disponíveis. Uma delas é a função *raiz quadrada*, `sqrt`. Um par de parênteses é utilizado após o nome da função para envolver o valor (chamado de *argumento* da função) que é processado pela função. Por exemplo, para calcular a raiz quadrada de 9 e atribuir o seu valor a uma variável *r*, você digita `r = sqrt(9)`. Note que o valor anterior de `r` foi substituído por 3.

ARGUMENTO

Ordem de precedência

ESCALAR

Um *escalar* é um número único. Uma *variável escalar* é uma variável que contém um número único. O MATLAB utiliza os símbolos + − * / ^ para adição, subtração, multiplicação, divisão e exponenciação (potência) de escalares. Eles estão listados na Tabela 1.1-1. Por exemplo, se você digitar `x = 8 + 3*5`, retornará a resposta `x = 23`. Se você digitar `2^3-10`, retornará a resposta `ans = -2`. A *barra* (/) representa a *divisão à direita*, que é o operador de divisão normal com o qual você é familiarizado. Se você digitar `15/3`, é retornado o resultado `ans = 5`.

O MATLAB possui outro operador de divisão, chamado de *divisão à esquerda*, que é indicado pela *barra invertida* (\). O operador de divisão à esquerda é útil para resolver conjuntos de equações algébricas lineares, como veremos. Uma boa maneira de lembrar a diferença entre os operadores de divisão à direita e à esquerda consiste em notar que a barra se inclina em direção ao denominador. Por exemplo, 7/2 = 2\7 = 3,5.

TABELA 1.1-1 Operações aritméticas escalares

Símbolo	Operação	Forma no MATLAB
^	exponenciação: a^b	`a^b`
*	multiplicação: ab	`a*b`
/	divisão à direita: $a/b = \frac{a}{b}$	`a/b`
\	divisão à esquerda: $a\backslash b = \frac{b}{a}$	`a\b`
+	adição: $a + b$	`a+b`
−	subtração: $a - b$	`a-b`

TABELA 1.1-2 Ordem de precedência

Precedência	Operação
Primeiro	Parênteses, a começar pelo par mais interno.
Segundo	Exponenciação, da esquerda para a direita.
Terceiro	Multiplicação e divisão com igual precedência, da esquerda para a direita.
Quarto	Adição e subtração com igual precedência, da esquerda para a direita.

PRECEDÊNCIA

As operações matemáticas representadas pelos símbolos + − * / \ e ^ obedecem a um conjunto de regras chamado de *precedência*. As expressões matemáticas são avaliadas a partir da esquerda, com a operação de exponenciação tendo a maior ordem de precedência, seguida pela multiplicação e pela divisão com igual precedência, seguidas, por sua vez, pela adição e pela subtração com igual precedência. Os parênteses podem ser utilizados para alterar essa ordem. A avaliação começa com o par de parênteses mais interno. A Tabela 1.1-2 resume essas regras. Por exemplo, observe o efeito da precedência na seguinte sessão.

```
>>8 + 3*5
ans =
     23
>>(8 + 3)*5
ans =
     55
>>4^2 - 12 - 8/4*2
ans =
     0
>>4^2 - 12 - 8/(4*2)
ans =
     3
>>3*4^2 + 5
ans =
     53
>>(3*4)^2 + 5
ans =
     149
>>27^(1/3) + 32^(0.2)
ans =
     5
>>27^(1/3) + 32^0.2
ans =
     5
>>27^1/3 + 32^0.2
ans =
     11
```

Para evitar erros, sinta-se livre para inserir parênteses sempre que estiver inseguro acerca do efeito que a precedência terá sobre o cálculo. Utilize os parênteses também para melhorar a legibilidade das suas expressões no MATLAB. Por exemplo, os parênteses não são necessários na expressão `8 + (3*5)`, mas eles deixam clara a sua intenção de multiplicar 3 por 5 antes de adicionar 8 ao resultado.

Teste seus conhecimentos

T1.1-1 Utilize o MATLAB para calcular as seguintes expressões

a. $6\left(\frac{10}{13}\right) + \frac{18}{5(7)} + 5(9^2)$

b. $6(35^{1/4}) + 14^{0,35}$

(Respostas: a. 410,1297 b. 17,1123)

O operador de atribuição

O sinal = no MATLAB é chamado de operador de *atribuição* ou de *substituição*. Ele trabalha de maneira diferente do sinal de igualdade que você conhece da matemática. Quando você digita `x = 3`, você está dizendo ao MATLAB para atribuir 3 à variável `x`. Isto não é diferente da que acontece na matemática. Entretanto, no MATLAB também podemos digitar algo como: `x = x + 2`. Isso diz ao MATLAB para adicionar 2 ao valor atual de `x` e substituir o valor atual de `x` com esse novo valor. Se `x` originalmente tivesse o valor 3, o seu novo valor seria 5. Esse uso do operador = é diferente do uso na matemática. Por exemplo, a equação matemática $x = x + 2$ é inválida porque implica que $0 = 2$.

No MATLAB a variável do *lado esquerdo* do operador = é substituída pelo valor gerado pela conta do *lado direito*. Portanto, uma variável, e apenas uma variável, deve estar do lado esquerdo do operador =. Dessa forma, no MATLAB você não pode digitar `6 = x`. Outra consequência dessa restrição é que você não pode escrever no MATLAB expressões como a seguinte:

```
>>x+2=20
```

A equação correspondente $x + 2 = 20$ é aceitável em álgebra e tem como solução $x = 18$, mas o MATLAB não pode resolver tal equação sem comandos adicionais (esses comandos estão disponíveis no toolbox de Matemática Simbólica, o qual será descrito no Capítulo 11).

Outra restrição é que o lado direito do operador = deve ter um valor calculável. Por exemplo, se a variável *y* não tiver um valor a ela atribuído, então o comando seguinte gerará uma mensagem de erro no MATLAB.

```
>>x = 5 + y
```

Além de associar valores conhecidos a variáveis, o operador de atribuição é muito útil para atribuir valores que não são conhecidos com antecedência, ou para mudar o valor de uma variável utilizando um procedimento prescrito. O exemplo seguinte mostra como isso é feito.

EXEMPLO 1.1-1 Volume de um cilindro circular

O volume de um cilindro circular de altura h e raio r é dado por $V = \pi r^2 h$. Um dado tanque cilíndrico possui altura de 15 m e um raio de 8 m. Queremos construir outro tanque cilíndrico com um volume 20% maior, mas com a mesma altura. Qual deve ser o valor do raio?

■ **Solução**

Primeiro resolvemos a equação do cilindro para o raio r. Isso resulta em

$$r = \sqrt{\frac{V}{\pi h}}$$

A sessão é mostrada abaixo. Primeiro atribuímos valores para as variáveis `r` e `h` que representam o raio e a altura. Depois, calculamos o volume do cilindro original e o aumentamos em 20%. Finalmente, resolvemos para o valor de raio requerido. Para este problema, podemos utilizar a constante interna do MATLAB `pi`.

```
>>r = 8;
>>h = 15;
>>V = pi*r^2*h;
>>V = V + 0.2*V;
>>r = sqrt(V/(pi*h))
r =
   8.7636
```

Assim, o novo cilindro deve ter um raio de 8,7636 m. Note que os valores originais das variáveis `r` e `V` são substituídos por dois valores novos. Isso é aceitável, uma vez que não desejamos utilizar os valores originais novamente. Note como a precedência se aplica à linha `V = pi*r^2*h;`. Ela é equivalente a `V = pi*(r^2)*h;`.

Nomes de variáveis

ESPAÇO DE TRABALHO

O termo *espaço de trabalho* (*workspace*) se refere aos nomes e valores de qualquer variável em uso na sessão atual de trabalho. Os nomes de variáveis devem começar com uma letra; o restante do nome pode conter letras, dígitos e traços inferiores (*underscore*). O MATLAB é sensível a maiúsculas e minúsculas (*case sensitive*). Portanto, os seguintes nomes representam cinco variáveis diferentes: `speed`, `Speed`, `SPEED`, `Speed_1` e `Speed_2`. No MATLAB 7, os nomes de variáveis não podem ter mais do que 63 caracteres.

Gerenciando a sessão de trabalho

A Tabela 1.1-3 resume alguns comandos e símbolos especiais para gerenciar a sessão de trabalho. Um sinal de ponto e vírgula no final da linha suprime a impressão dos resultados na tela. Se um sinal de ponto e vírgula não for inserido no final de uma linha, o MATLAB exibe os resultados da linha na tela. Mesmo que você suprima a exibição com o ponto e vírgula, o MATLAB continua armazenando o valor da variável.

TABELA 1.1-3 Comandos para gerenciar a sessão de trabalho

Comando	Descrição
`clc`	Limpa a janela de comandos.
`clear`	Remove todas as variáveis da memória.
`clear var1 var2`	Remove as variáveis `var1` e `var2` da memória.
`exist('name')`	Determina se existe um arquivo ou variável com o nome 'name'.
`quit`	Fecha o MATLAB.
`who`	Lista todas as variáveis na memória.
`whos`	Lista todas as variáveis e tamanhos e indica se elas possuem parte imaginária.
`:`	Dois pontos; gera um arranjo com elementos igualmente espaçados.
`,`	Vírgula; separa elementos de um arranjo.
`;`	Ponto e vírgula; suprime impressão na tela; também indica uma nova linha em um arranjo.
`...`	Reticências; continua uma linha.

Você pode inserir vários comandos na mesma linha separando-os com uma vírgula se quiser ver o resultado do comando anterior, ou com um ponto e vírgula se quiser suprimir a exibição. Por exemplo,

```
>>x=2;y=6+x,x=y+7
y =
    8
x =
    15
```

Note que o primeiro valor de `x` não foi exibido. Note também que o valor de `x` mudou de 2 para 15.

Se você precisa digitar uma linha longa, você pode utilizar o sinal de *reticências*, digitando três pontos, para atrasar a execução. Por exemplo,

```
>>NumberOfApples = 10; NumberOfOranges = 25;
>>NumberOfPears = 12;
>>FruitPurchased = NumberOfApples + NumberOfOranges ...
+NumberOfPears
FruitPurchased =
     47
```

Utilize as teclas direcionais **Tab** e **Ctrl** para recuperar, editar e reutilizar funções e variáveis que você digitou anteriormente. Por exemplo, suponhamos que você insira a linha

```
>>volume = 1 + sqr(5)
```

O MATLAB responde com uma mensagem de erro porque você não digitou o `t` de `sqrt`. Em vez de redigitar toda a linha, pressione a tecla direcional superior (↑) uma vez para exibir a última linha digitada. Aperte a tecla direcional esquerda (←) algumas vezes para mover o cursor e adicionar o `t` que está faltando, e então pressione **Enter**. O uso repetido da tecla direcional superior recupera linhas digitadas anteriormente.

Teclas Tab e direcionais

Você pode utilizar a funcionalidade de *recuperação inteligente* para recuperar uma função ou variável digitada anteriormente cujos primeiros caracteres você especifique. Por exemplo, após ter inserido a linha que começa com `volume`, ao digitar `vol` e pressionar a tecla direcional superior (↑) uma vez, você recupera a última linha digitada que começa com a função ou variável cujo nome se inicia com `vol`. Essa funcionalidade é sensível a maiúsculas e minúsculas.

Você pode utilizar a funcionalidade *tab completion* para reduzir a quantidade de digitação. O MATLAB automaticamente completa o nome de uma função, variável ou arquivo se você digitar as primeiras letras do nome e pressionar a tecla **Tab**. Se o nome for único, ele é completado automaticamente. Por exemplo, na sessão listada anteriormente, se você digitar `Fruit` e pressionar **Tab**, o MATLAB completará o nome e exibirá `FruitPurchased`. Pressione **Enter** para exibir o valor da variável, ou continue editando para criar uma nova linha executável que utiliza a variável `FruitPurchased`.

Se houver mais do que um nome que começa com as letras que você digitou, o MATLAB exibirá esses nomes quando você pressionar a tecla **Tab**. Utilize o mouse para selecionar o nome desejado na lista.

As teclas direcionais esquerda (←) e direita (→) movem o cursor para a esquerda e para a direita um *caractere* por vez ao longo de uma linha. Para mover o cursor uma *palavra* por vez, pressione **Ctrl** e → ao mesmo tempo para movê-lo para a *direita*; e **Ctrl** e ← ao mesmo tempo para movê-lo para a *esquerda*. Pressione **Home** para mover o cursor para o começo de uma linha; pressione **End** para mover o cursor para o final de uma linha.

Deletando e removendo

Pressione **Del** para deletar o caractere no cursor; pressione **Backspace** para deletar o caractere antes do cursor. Pressione **Esc** para apagar toda a linha; pressione **Ctrl** e **k** simultaneamente para deletar (*kill*) os caracteres desde o cursor até o final da linha.

O MATLAB armazena o último valor de uma variável até que você feche o MATLAB ou apague esse valor. Negligenciar esse fato comumente causa erros. Por exemplo, você pode preferir utilizar a variável `x` em vários cálculos diferentes. Se você se esquecer de atribuir a `x` o valor correto, o MATLAB utilizará o último valor, e você obterá um resultado incorreto. Você pode utilizar a função `clear` para remover os valores de *todas* as variáveis da memória, ou pode utilizar a forma `clear var1 var2` para remover as variáveis `var1` e `var2`. O resultado do comando `clc` é diferente; ele remove tudo que está exibido na janela de Comandos, mas os valores das variáveis são mantidos.

Você pode digitar o nome de uma variável e pressionar **Enter** para ver o seu valor atual. Se a variável não tiver um valor (isto é, se ela não existir), você verá uma mensagem de erro. Você também pode utilizar a função `exist`. Digite `exist('x')` para ver se a variável `x` está em uso. Se retornar o valor 1, a variável existe; um 0 indica que ela não existe. A função `who` lista os nomes de todas as variáveis na memória, mas não mostra os seus valores. A forma `who var1 var2` restringe a exibição aos valores especificados. O caractere curinga * pode ser utilizado para exibir variáveis que atendem a um padrão. Por exemplo, `who A*` encontra todas as variáveis no espaço de trabalho atual que comecem com A. A função `whos` lista os nomes das variáveis e os seus tamanhos, e indica se elas possuem parte imaginária não nula.

A diferença entre uma função, um comando e uma sentença é que a função tem os seus argumentos evolvidos por parênteses. Comandos, como `clear`, não precisam ter argumentos; mas se tiverem, eles não são envolvidos por parênteses, como em `clear x`, por exemplo. Sentenças não podem ter argumentos; por exemplo, `clc` e `quit` são sentenças.

Pressione **Ctrl-C** para cancelar uma computação longa sem terminar a sessão. Você pode sair do MATLAB digitando `quit`. Você também pode clicar no menu **File** e então clicar em **Exit MATLAB**.

Constantes predefinidas

O MATLAB possui várias constantes especiais predefinidas, tais como a constante interna `pi` que utilizamos no Exemplo 1.1-1. O símbolo `Inf` representa ∞, o que, na prática, significa um número tão grande que o MATLAB não consegue representá-lo. Por exemplo, ao digitar `5/0`, obtemos a resposta `Inf`. O símbolo `NaN`† representa "não é um número". Ele indica um resultado numérico indefinido tal como o obtido ao digitarmos `0/0`. O símbolo `eps` é o menor número que, quando somado a 1 pelo computador, resulta em um número maior do que 1. Nós o utilizamos como um indicador da precisão dos cálculos.

Os símbolos `i` e `j` indicam a unidade imaginária, em que $i = j = \sqrt{-1}$. Nós os utilizamos para criar e representar números complexos, tais como `x = 5 + 8i`.

Tente não utilizar os nomes de constantes especiais como nomes de variáveis. Apesar de o MATLAB permitir que você atribua um valor diferente a essas constantes, não é uma boa prática fazer isso.

Operações com números complexos

O MATLAB manipula álgebra de números complexos automaticamente. Por exemplo, o número $c_1 = 1 - 2i$ é inserido da seguinte maneira: `c1 = 1 - 2i`. Você também pode digitar `c1 = Complex(1, -2)`.

> **Atenção:** Note que não é necessário um asterisco entre `i` ou `j` e um número, apesar de ele ser necessário com uma variável, tal como `c2 = 5 - i*c1`. Essa convenção pode causar erros se você não for cuidadoso. Por exemplo, as expressões `y = 7/2*i` e `x = 7/2i` geram dois resultados diferentes: $y = (7/2)i = 3{,}5i$ e $x = 7/(2i) = -3{,}5i$.

TABELA 1.1-4 Variáveis e constantes especiais

Comando	Descrição
`ans`	Variável temporária que contém a resposta mais recente.
`eps`	Especifica a acurácia da precisão de ponto flutuante.
`i,j`	A unidade imaginária $\sqrt{-1}$.
`Inf`	Infinito.
`NaN`	Indica um resultado numérico indefinido.
`pi`	O número π.

† N. de T.: NaN é a sigla de *not a number*.

A adição, a subtração, a multiplicação e a divisão de números complexos são realizadas facilmente. Por exemplo,

```
>>s = 3+7i;w = 5-9i;
>>w+s
ans =
     8.0000 - 2.0000i
>>w*s
ans =
     78.0000 + 8.0000i
>>w/s
ans =
     -0.8276 - 1.0690i
```

Teste seus conhecimentos

T1.1-2 Dados $x = -5 + 9i$ e $y = 6 - 2i$, utilize o MATLAB para mostrar que $x + y = 1 + 7i$, $xy = -12 + 64i$ e $x/y = -1{,}2 + 1{,}1i$.

Comandos de formatação

O comando `format` controla como os números aparecem na tela. A Tabela 1.1-5 apresenta as variantes desse comando. O MATLAB utiliza muitos algarismos significativos em seus cálculos, mas raramente precisamos ver todos eles. O formato de exibição padrão do MATLAB é o formato `short`, que utiliza quatro dígitos decimais. Você pode exibir mais digitando `format long`, o que resulta na exibição de 16 dígitos. Para retornar ao modo padrão, digite `format short`.

Você pode forçar para que a saída esteja em notação científica digitando `format short e`, ou `format long e`, em que `e` representa o número 10. Assim, a saída `6.3792e+03` representa o número $6{,}3792 \times 10^3$. A saída `6.3792e-03` representa o número $6{,}3792 \times 10^{-3}$. Note que nesse contexto `e` não representa o número e, que é a base do logaritmo natural. Aqui, `e` representa "expoente". Essa é uma

TABELA 1.1-5 Formatos de exibição numérica

Comando	Descrição e exemplo
`format short`	Quatro dígitos decimais (o padrão); 13.6745.
`format long`	16 dígitos; 17.27484029463547.
`format short e`	Cinco dígitos (quatro decimais) mais o expoente; 6.3792e+03.
`format long e`	16 dígitos (15 decimais) mais o expoente; 6.379243784781294e–04.
`format bank`	Dois dígitos decimais; 126.73.
`format +`	Positivo, negativo ou zero; +.
`format rat`	Aproximação racional; 43/7.
`format compact`	Suprime algumas linha em branco.
`format loose`	Restabelece o modo de exibição menos compacto.

FIGURA 1.2-1 A parte superior da Área de Trabalho do MATLAB.

escolha pobre de notação, mas o MATLAB segue os padrões convencionais de programação de computadores que foram estabelecidos há muitos anos.

Utilize `format bank` apenas para operações monetárias; esse modo não reconhece parte imaginária.

1.2 Menus e barra de ferramentas

A Área de Trabalho gerencia a janela de Comandos e outras ferramentas do MATLAB. A aparência padrão da Área de Trabalho é apresentada na Figura 1.1-1. Ao longo da parte superior da Área de Trabalho estão uma linha de nomes de menu e uma linha de ícones chamada de *barra de ferramentas*. À direita da barra de ferramentas está uma caixa que mostra o *diretório atual*, onde o MATLAB busca por arquivos. Veja a Figura 1.2-1.

DIRETÓRIO ATUAL

Outras janelas aparecem em uma sessão do MATLAB, a depender do que você fizer. Por exemplo, uma janela gráfica contendo uma plotagem aparece quando você utiliza as funções de plotagem; uma janela do Editor/Debugador aparece para ser utilizada na criação de arquivos de programa. Cada janela possui a sua própria barra de menus, com um ou mais menus, na parte superior. *Assim, a barra de menus mudará quando você mudar de janela*. Para ativar ou selecionar um menu, clique nele. Cada menu possui alguns itens. Clique em um item para selecioná-lo. *Mantenha em mente que os menus são sensíveis a contexto. Assim, seus conteúdos mudam, dependendo de quais funcionalidades você estiver utilizando*.

Os menus da área de trabalho

A maior parte da sua interação se dará na janela de Comandos. Quando a janela de Comandos está ativa, a Área de Trabalho padrão do MATLAB 7 (mostrada na Figura 1.1-1) apresenta seis menus: **File**, **Edit**, **Debug**, **Desktop**, **Window** e **Help**. Note que esses menus mudam dependendo da janela que estiver ativa. Cada item em um menu pode ser selecionado com o menu aberto tanto clicando no item quanto digitando sua letra sublinhada. Alguns itens podem ser selecionados, sem o menu estar aberto, por meio da tecla de atalho listada à direita do item. Aqueles itens seguidos por três pontos (**...**) abrem um submenu ou outra janela contendo uma caixa de diálogo.

Os três menus mais úteis são os menus **File**, **Edit** e **Help**. O menu **Help** está descrito na Seção 1.5. O menu **File** no MATLAB 7 contém os seguintes itens, que realizam as ações indicadas quando você os seleciona.

O Menu File no MATLAB 7

New Abre uma caixa de diálogo que permite criar um novo arquivo de programa, chamado de arquivo M (M-file), utilizando um editor de texto chamado de Editor/Debugador, uma nova Figura, uma variável na janela do Es-

paço de Trabalho, um arquivo do tipo Model (utilizado pelo Simulink) ou uma nova GUI (Interface Gráfica do Usuário)†.

Open... Abre uma caixa de diálogo que permite selecionar um arquivo para edição.

Close Command Window (ou **Current Folder**) Fecha a janela de Comandos ou o arquivo atual, se algum estiver aberto.

Import Data... Inicia o Import Wizard, que permite importar dados facilmente.

Save Workspace As... Abre uma caixa de diálogo que permite salvar um arquivo.

Set Path... Abre uma caixa de diálogo que permite definir o caminho de busca do MATLAB.

Preferences... Abre uma caixa de diálogo que permite definir as preferências para itens como fonte, cor, espaçamento da tecla **Tab**, e assim por diante.

Page Setup... Abre uma caixa de diálogo que permite formatar a saída impressa.

Print... Abre uma caixa de diálogo que permite imprimir toda a janela de Comandos.

Print Selection... Abre uma caixa de diálogo que permite imprimir partes selecionadas da janela de Comandos.

File List Contém uma lista dos arquivos utilizados anteriormente, em ordem de utilização mais recente.

Exit MATLAB Fecha o MATLAB.

A opção **New** no menu **File** permite selecionar o tipo de arquivo M a ser criado: um arquivo M em branco, um arquivo M do tipo função ou um arquivo M do tipo classe. Selecione arquivo M em branco para criar um arquivo M do tipo discutido na Seção 1.4. Arquivos M do tipo função são discutidos no Capítulo 3, mas arquivos M do tipo classe estão além do escopo deste texto.

O menu **Edit** contém os seguintes itens.

O Menu Edit no MATLAB 7

Undo Reverte a operação de edição anterior.

Redo Reverte a operação Undo anterior.

Cut Remove o texto selecionado e o armazena para colagem posterior.

Copy Copia o texto selecionado para colagem posterior, sem removê-lo.

Paste Insere qualquer texto do clipboard na localização atual do cursor.

Paste to Workspace... Insere os conteúdos do clipboard no espaço de trabalho (workspace) como uma ou mais variáveis.

Select All Seleciona todo o texto na janela de Comandos.

Delete Apaga a variável selecionada no Navegador do Espaço de Trabalho (Browser Workspace).

† N. de T.: GUI é a sigla de *Graphical User Interface*.

Find... Encontra e substitui frases.

Find Files... Encontra arquivos.

Clear Command Window Remove todo o texto da janela de Comandos.

Clear Command History Remove todo o texto da janela de Histórico de Comandos.

Clear Workspace Remove os valores de todas as variáveis do Espaço de Trabalho.

Você pode utilizar **Copy** e **Paste** para copiar e colar comandos que aparecem na janela de Comandos. Entretanto, uma maneira mais fácil é utilizar a tecla direcional superior para percorrer a lista de comandos anteriores e pressionar **Enter** quando você vir o comando que deseja recuperar.

Utilize o menu **Debug** para acessar o Debugador, que é discutido no Capítulo 4. Utilize o menu **Desktop** para controlar a configuração da Área de Trabalho e para exibir barras de ferramentas. O menu **Window** apresenta um ou mais itens, dependendo do que você tiver feito até então na sua sessão. Clique no nome de uma janela que aparece no menu para abri-la. Por exemplo, se você tiver plotado uma figura e não tiver fechado a sua janela, a janela da plotagem aparecerá nesse menu como **Figure 1**. Entretanto, há outras maneiras de ir de uma janela para outra (tal como pressionando as teclas **Alt** e **Tab** simultaneamente se as janelas não estiverem ancoradas).

O menu **View** aparecerá à direita do menu **Edit** se você tiver selecionado um arquivo em uma pasta na janela de Diretório Atual. Esse menu fornece informação sobre o arquivo selecionado.

A barra de ferramentas, que está abaixo da barra de menus, fornece botões como atalhos para algumas das funcionalidades nos menus. Clicar no botão é equivalente a clicar no menu e em seguida clicar no item do menu; assim, o botão elimina um clique do mouse. Os primeiros sete botões correspondem a **New**, **M-File**, **Open File**, **Cut**, **Copy**, **Paste**, **Undo** e **Redo**. O oitavo botão ativa o Simulink. O nono botão ativa o GUIDE Quick Start, que é utilizado para criar e editar interfaces gráficas do usuário (GUIs). O décimo botão ativa o Profiler, que pode ser utilizado para otimizar o desempenho do programa. O décimo primeiro botão (com um ponto de interrogação) acessa o Sistema de Ajuda (Help System).

Abaixo da barra de ferramentas está um botão que acessa a ajuda para adicionar atalhos à barra de ferramentas e um botão que acessa uma lista das funcionalidades adicionadas desde o lançamento (*release*) anterior.

1.3 Arranjos, arquivos e plotagens

Esta sessão introduz arranjos, que são os blocos de construção básicos no MATLAB, e mostra como manipular arquivos e gerar plotagens.

Arranjos

Ao longo do texto, discutiremos centenas de funções no MATLAB. Por exemplo, para calcular sen x, em que x tem o seu valor em radianos, você deve digitar `sin(x)`. Para calcular cos x, digite `cos(x)`. A função exponencial e^x é calculada a partir de `exp(x)`. O logaritmo natural, ln x, é calculado digitando-se `log(x)` (note a

diferença entre o texto matemático, ln, e a sintaxe do MATLAB, `log`). Você calcula o logaritmo na base 10 digitando `log10(x)`. O inverso do seno, ou o arco seno, é obtido ao se digitar `asin(x)`. A resposta é retornada em radianos, não em graus. A função `asind(x)` retorna o valor em graus.

ARRANJO

Um dos pontos fortes do MATLAB é a sua habilidade em manipular coleções de números, chamadas *arranjos*, como se elas fossem uma única variável. Um arranjo numérico é uma coleção ordenada de números (um conjunto de números arranjados em uma ordem específica). Um exemplo é uma variável do tipo arranjo que contém os números 0, 4, 3 e 6, nesta ordem. Nós utilizamos colchetes para definir a variável `x` que contém essa coleção digitando `x = [0, 4, 3, 6]`. Os elementos do arranjo também podem ser separados por espaços, mas vírgulas são preferidas para melhorar a legibilidade e evitar erros. Note que a variável `y` definida como `y = [6, 3, 4, 0]` não é igual a `x`, porque a ordem dos números é diferente. A razão para se utilizar colchetes é a seguinte: se você digitasse `x = 0, 4, 3, 6,` o MATLAB iria tratar as quatro entradas separadamente e atribuiria o valor 0 a `x`. Pode-se considerar que arranjo [0, 4, 3, 6] possui uma linha e quatro colunas, e é um subcaso de uma *matriz*, que possui múltiplas linhas e colunas. Como veremos mais adiante, as matrizes também são indicadas por colchetes.

Podemos somar os dois arranjos `x` e `y` para produzir outro arranjo `z` digitando a linha `z = x + y`. Para calcular `z`, o MATLAB soma todos os números correspondentes em `x` e `y`. O arranjo resultante `z` contém os números 6, 7, 7, 6.

Você não precisa digitar todos os números em um arranjo se eles forem igualmente espaçados. Em vez disso, você digita o primeiro número e o último número, com o espaçamento no meio, separados por dois pontos. Por exemplo, os números 0, 0,1, 0,2,..., 10 podem ser atribuídos a `u` digitando-se `u = 0:0.1:10`. Nesta aplicação do operador dois pontos, os colchetes não precisam ser utilizados.

Para calcular $w = 5 \text{ sen } u$ para $u = 0, 0{,}1, 0{,}2, \ldots, 10$, a sessão é

```
>>u = 0:0.1:10;
>>w = 5*sin(u);
```

A linha `w = 5*sin(u)` calculou a fórmula $w = 5 \text{ sin } u$ 101 vezes, uma vez para cada valor no arranjo `u`, para produzir um arranjo `w` que possui 101 valores.

Você pode ver todos os valores de `u` digitando `u` depois do prompt; ou, por exemplo, você pode ver o sétimo valor digitando `u(7)`. O número 7 é chamado de um *índice de arranjo*, porque ele aponta para um elemento em particular no arranjo.

ÍNDICE DE ARRANJO

```
>>u(7)
ans =
     0.6000
>>w(7)
ans =
     2.8232
```

Você pode utilizar a função `length` para determinar quantos valores há em um arranjo. Por exemplo, continue a sessão anterior da seguinte maneira:

```
>>m = length(w)
m =
   101
```

Arranjos exibidos na tela como uma única linha de números com mais de uma coluna são chamados de *arranjos linha*. Você pode criar *arranjos coluna*, que possuem mais de uma linha, utilizando ponto e vírgula para separar as linhas.

Raízes de polinômios

Podemos descrever um polinômio no MATLAB com um arranjo cujos elementos são os coeficientes do polinômio, *começando com o coeficiente da maior potência de x*. Por exemplo, o polinômio $4x^3 - 8x^2 + 7x - 5$ seria representado pelo arranjo `[4,-8,7,-5]`. As *raízes* do polinômio $f(x)$ são os valores de x de tal modo que $f(x) = 0$. As raízes de um polinômio podem ser calculadas com a função `roots(a)`, em que `a` é o arranjo de coeficientes do polinômio. O resultado é um arranjo *coluna* que contém as raízes do polinômio. Por exemplo, para encontrar as raízes de $x^3 - 7x^2 + 40x - 34 = 0$, a sessão é

```
>>a = [1,-7,40,-34];
>>roots(a)
ans =
     3.0000 + 5.000i
     3.0000 - 5.000i
     1.0000
```

As raízes são $x = 1$ e $x = 3 \pm 5i$. Os dois comandos poderiam ter sido combinados no comando único `roots([1,-7,40,-34])`.

Teste seus conhecimentos

T1.3-1 Utilize o MATLAB para determinar quantos elementos há no arranjo `cos(0):0.02:log10(100)`. Utilize o MATLAB para determinar o vigésimo quinto elemento. (Resposta: 51 elementos e 1,48)

T1.3-2 Utilize o MATLAB para encontrar as raízes do polinômio $290 - 11x + 6x^2 + x^3$. (Resposta: $x = -10, 2 \pm 5i$)

Funções internas

Vimos algumas funções internas do MATLAB, tais como as funções `sqrt` e `sin`. A Tabela 1.3-1 lista algumas das funções internas do MATLAB comumente utilizadas. O Capítulo 3 traz uma extensa abordagem sobre as funções internas. Os usuários do MATLAB podem criar suas próprias funções para suas necessidades especiais. A criação de funções definidas pelo usuário é abordada no Capítulo 3.

Trabalhando com arquivos

O MATLAB utiliza alguns tipos de arquivos que permitem salvar programas, dados e resultados de sessões. Como veremos na Seção 1.4, arquivos de funções do MATLAB e arquivos de programas são salvos com a extensão `.m`, e portanto são chamados de arquivos M. *Arquivos MAT* possuem a extensão `.mat` e são utilizados para salvar os nomes e os valores das variáveis criadas durante uma sessão do MATLAB.

ARQUIVOS MAT

TABELA 1.3-1 Algumas funções matemáticas comumente utilizadas

Função	Sintaxe do MATLAB†
e^x	`exp(x)`
$\sqrt{x}$	`sqrt(x)`
$\ln x$	`log(x)`
$\log_{10} x$	`log10(x)`
$\cos x$	`cos(x)`
$\text{sen}\ x$	`sin(x)`
$\tan x$	`tan(x)`
$\cos^{-1} x$	`acos(x)`
$\text{sen}^{-1} x$	`asin(x)`
$\tan^{-1} x$	`atan(x)`

†As funções trigonométricas do MATLAB listadas aqui utilizam medidas em radianos. As funções trigonométricas que terminam em `d`, tais como `sind(x)` e `cosd(x)`, consideram o argumento `x` em graus. As funções inversas, como `atand(x)`, retornam valores em graus.

ARQUIVOS ASCII

Como eles são *arquivos ASCII*†, os arquivos M podem ser criados utilizando-se apenas um processador de textos. Os arquivos MAT são arquivos *binários* que geralmente podem ser lidos apenas pelo software que os criou. Eles contêm uma assinatura de máquina que permite que sejam transferidos entre tipos diferentes de máquina, tais como MS Windows e Macintosh.

ARQUIVO DE DADOS

O terceiro tipo de arquivo que utilizaremos é um arquivo de dados, especificamente um *arquivo de dados* ASCII, isto é, criado de acordo com o formato ASCII. Você pode precisar utilizar o MATLAB para analisar dados armazenados em um arquivo gerado por um programa de planilhas, um processador de textos, um sistema de aquisição de dados de um laboratório ou de um arquivo que você compartilha com alguém.

Salvando e recuperando suas variáveis do espaço de trabalho (*workspace*)

Se você quiser continuar uma sessão do MATLAB posteriormente, você pode utilizar os comandos `save` e `load`. Quando você digita `save`, o MATLAB salva as variáveis do espaço de trabalho, isto é, os nomes das variáveis, seus tamanhos e seus valores, em um arquivo binário chamado `matlab.mat`, o qual o MATLAB é capaz de ler. Para recuperar suas variáveis do espaço de trabalho, digite `load`. Você pode então continuar sua sessão como antes. Para salvar as variáveis do espaço de trabalho em outro arquivo chamado `filename.mat`, digite `save filename`. Para carregar as variáveis do espaço de trabalho, digite `load filename`. Se o arquivo MAT `filename` que foi salvo contiver as variáveis `A`, `B`, e `C`, então ao carregá-lo as variáveis serão inseridas de volta no espaço de trabalho e sobrescreverão quaisquer variáveis existentes que tenham os mesmos nomes.

Para salvar apenas algumas das suas variáveis, por exemplo, `var1` e `var2`, no arquivo `filename.mat`, digite `save filename.mat var1 var2`. Você não precisa digitar os nomes das variáveis para recuperá-las; apenas digite `load filename`.

† N. de T.: ASCII é um acrônimo para *American Standard Code for Information Interchange*, que em português significa "Código Padrão Americano para o Intercâmbio de Informação".

Diretórios e caminho É importante saber a localização dos arquivos que você utiliza no MATLAB. Isso frequentemente causa problemas para os iniciantes. Suponha que você utilize o MATLAB no seu computador em casa e salve um arquivo em um disco removível, como será discutido mais adiante nesta seção. Se você levar esse disco para utilizar com o MATLAB em outro computador, como, por exemplo, no laboratório de computadores da sua escola, você deve se certificar de que o MATLAB sabe como encontrar os seus arquivos. Os arquivos são armazenados em *diretórios*, também chamados de *pastas*. Os diretórios podem ter subdiretórios abaixo deles. Por exemplo, suponha que o MATLAB foi instalado no drive c:, no diretório `c:\matlab`. Então o diretório `toolbox` é um subdiretório abaixo do diretório `c:\matlab`, e `symbolic` é um subdiretório abaixo do diretório `toolbox`. O *caminho* informa ao MATLAB como encontrar um arquivo em particular.

CAMINHO

Trabalhando com discos removíveis Na Seção 1.4 você aprenderá como criar e salvar arquivos M. Suponha que você tenha salvado o arquivo `problema1.m` no diretório `\homework` em um disco, o qual você insere no drive f:. O caminho para esse arquivo é `f:\homework`. Como o MATLAB está normalmente instalado, quando você digita `problema1`,

1. O MATLAB primeiro checa para ver se `problema1` é uma variável e, se for, exibe o seu valor.
2. Se não, o MATLAB então checa para ver se `problema1` é um dos seus próprios comandos; em caso afirmativo, ele o executa.
3. Se não, o MATLAB então procura no diretório atual por um arquivo com nome `problem1.m` e executa `problem1`, se ele o encontra.
4. Se não, o MATLAB então busca por `problem1.m` em ordem nos diretórios em seu *caminho de busca* e, caso o encontre, ele o executa.

CAMINHO DE BUSCA

Você pode exibir o caminho de busca do MATLAB digitando `path`. Se `problem1` estiver apenas no disco e se o diretório f: não estiver no caminho de busca, o MATLAB não encontrará o arquivo e gerará uma mensagem de erro, a não ser que você diga a ele onde procurar. Você pode fazer isso digitando `cd f:\homework`, que representa "mude o diretório para f:\homework". Isso mudará o diretório atual para `f:\homework` e forçará o MATLAB a procurar seu arquivo nesse diretório. A sintaxe geral desse comando é `cd dirname`, em que `dirname` é o caminho completo até o diretório.

Uma alternativa a esse procedimento é copiar o seu arquivo para um diretório no disco rígido que está no caminho de busca. Entretanto, há algumas armadilhas com essa abordagem: (1) se você alterar o arquivo durante sua sessão, pode se esquecer de copiar o arquivo revisado de volta para o seu disco; (2) o disco rígido fica desorganizado (esse é um problema em laboratórios de computadores públicos, e talvez você não seja autorizado a salvar seu arquivo em um disco rígido); (3) o arquivo pode ser apagado ou sobrescrito se o MATLAB for reinstalado; e (4) outra pessoa pode ter acesso ao seu trabalho!

Você pode determinar o diretório atual (aquele em que o MATLAB procura por seu arquivo) digitando `pwd`. Para ver uma lista de todos os arquivos no diretório atual, digite `dir`. Para ver os arquivos no diretório `dirname`, digite `dir dirname`.

TABELA 1.3-2 Comandos de sistema, diretórios e arquivos

Command	Descrição
`addpath dirname`	Adiciona o diretório `dirname` ao caminho de busca.
`cd dirname`	Muda o diretório atual para `dirname`.
`dir`	Lista todos os arquivos no diretório atual.
`dir dirname`	Lista todos os arquivos no diretório `dirname`.
`path`	Exibe o caminho de busca do MATLAB.
`pathtool`	Inicializa a ferramenta Set Path.
`pwd`	Exibe o diretório atual.
`rmpath dirname`	Remove o diretório `dirname` do caminho de busca.
`what`	Lista os arquivos específicos do MATLAB encontrados no diretório de trabalho atual. A maioria dos arquivos de dados e outros arquivos que não são específicos do MATLAB não são listados. Utilize `dir` para obter uma lista de todos os arquivos.
`what dirname`	Lista os arquivos específicos do MATLAB no diretório `dirname`.
`which item`	Exibe o nome do caminho de `item` se `item` for uma função ou arquivo. Se `item` for uma variável, o comando o identifica como tal.

O comando `what` exibe uma lista dos arquivos específicos do MATLAB no diretório atual. O comando `what dirname` faz o mesmo para o diretório `dirname`. Digite `which item` para exibir o nome do caminho completo da função `item` ou do arquivo `item` (inclua a extensão do arquivo). Se `item` for uma variável, o MATLAB o identifica como tal.

Você pode adicionar um diretório ao caminho de busca utilizando o comando `addpath`. Para remover um diretório do caminho de busca, utilize o comando `rmpath`. A ferramenta Set Path é uma interface gráfica para trabalhar com arquivos e diretórios. Digite `pathtool` para iniciar o navegador. Para salvar as configurações de caminho, clique em **Save** na ferramenta. Para restaurar o caminho de busca padrão, clique em **Default** no navegador.

Esses comandos estão resumidos na Tabela 1.3-2.

Plotando com o MATLAB

O MATLAB possui muitas funções poderosas e fáceis de serem utilizadas para a criação de plotagens de vários tipos diferentes, tais como plotagens retilíneas, logarítmicas, de superfícies e de contornos. Como um exemplo simples, plotemos a função $y = 5 \operatorname{sen} x$ para $0 \leq x \leq 7$. Utilizaremos um incremento de 0,01 para gerar um número grande de valores de `x` a fim de produzir uma curva suave. A função `plot(x,y)` gera um gráfico com os valores de `x` no eixo horizontal (as abscissas) e com os valores de `y` no eixo vertical (as ordenadas). A sessão é

```
>>x = 0:0.01:7;
>>y = 3*cos(2*x);
>>plot(x,y),xlabel('x'),ylabel('y')
```

JANELA GRÁFICA

A plotagem aparece na tela em uma *janela gráfica*, com o nome **Figure 1**, como pode ser visto na Figura 1.3-1. A função `xlabel` insere o texto entre aspas simples como um rótulo no eixo horizontal. A função `ylabel` faz o mesmo para o eixo vertical.

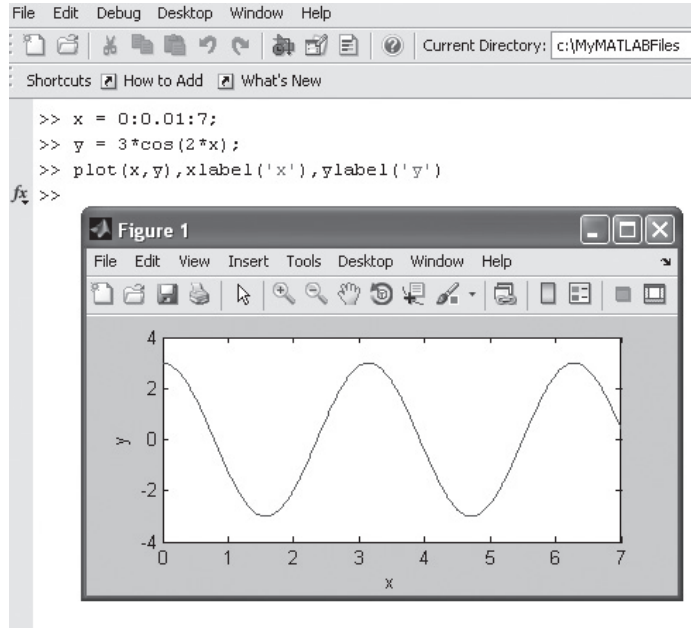

FIGURA 1.3-1 Uma janela gráfica mostrando uma plotagem.

Quando o comando `plot` é executado com sucesso, uma janela gráfica aparece automaticamente. Se for desejada uma cópia da plotagem no disco rígido, ela pode ser obtida clicando-se em **Print** no menu **File** na janela gráfica. A janela pode ser fechada clicando-se em **Close** no menu **File** da janela gráfica; depois disso, o cursor retornará ao prompt na janela de Comandos.

Outras funções de plotagens úteis são `title` e `gtext`, que inserem texto em uma plotagem. Ambas aceitam texto entre parênteses e aspas simples, assim como a função `xlabel`. A função `title` insere o texto na parte superior da plotagem; a função `gtext` insere o texto no ponto da plotagem em que o cursor estiver localizado quando você clicar com o botão esquerdo do mouse.

PLOTAGENS SOBREPOSTAS

Você pode criar plotagens múltiplas, chamadas de *plotagens sobrepostas*, incluindo outro conjunto ou outros conjuntos de valores na função `plot`. Por exemplo, para plotar as funções $y = 2\sqrt{x}$ e $z = 4 \text{ sen } 3x$ para $0 \leq x \leq 5$ com o mesmo comando `plot`, a sessão é

```
>>x = 0:0.01:5;
>>y = 2*sqrt(x);
>>z = 4*sin(3*x);
>>plot(x,y,x,z),xlabel('x'),gtext('y'),gtext('z')
```

Depois que a plotagem aparece na tela, o programa espera para que você posicione o cursor e clique com o botão do mouse, uma vez para cada comando `gtext` utilizado. Utilize a função `gtext` para inserir os rótulos *y* e *z* próximos às curvas apropriadas.

Você também pode distinguir as curvas umas das outras utilizando diferentes tipos de linha para cada curva. Por exemplo, para plotar a curva *z* utilizando

uma linha tracejada, substitua a função `plot(x,y,x,z)` na sessão anterior por `plot(x,y,x,z, '- -')`. Outros tipos de linha podem ser utilizados, os quais são discutidos no Capítulo 5.

Algumas vezes é útil ou necessário obter as coordenadas de um ponto em uma curva plotada. A função `ginput` pode ser utilizada para esse fim. Insira essa função ao final de todas as plotagens e de suas sentenças de formatação, de modo que a plotagem já esteja em seu formato final. O comando `[x,y] = ginput(n)` captura *n* pontos e retorna as coordenadas *x* e *y* nos vetores `x` e `y`, ambos com tamanho *n*. Posicione o cursor e clique com o mouse. As coordenadas retornadas e as coordenadas da plotagem se encontram na mesma escala.

MARCADORES DE DADOS

Nos casos em que você estiver plotando *dados*, ao invés de funções, você deveria utilizar *marcadores de dados* para plotar cada ponto (a não ser que haja muitos pontos). Para marcar cada ponto com o sinal de soma +, a sintaxe necessária para a função `plot` é `plot(x,y,'+')`. Você pode conectar os pontos dos dados com linhas, se desejar. Nesse caso, você deve plotar os dados duas vezes, uma vez com os marcadores de dados e outra sem os marcadores.

Por exemplo, suponha que a variável independente dos dados seja `x = [15:2:23]` e que os valores da variável dependente sejam `y = [20, 50, 60, 90, 70]`. Para plotar os dados com sinais de soma, utilize a seguinte sessão:

```
>>x = 15:2:23;
>>y = [20, 50, 60, 90, 70];
>>plot(x,y,'+',x,y),xlabel('x'),ylabel('y'), grid
```

O comando `grid` insere linhas de grid na plotagem. Outros marcadores de dados estão disponíveis, os quais são discutidos no Capítulo 5.

A Tabela 1.3-3 resume esses comandos de plotagem. Discutiremos outras funções de plotagem e o Editor de Plotagens (*Plot Editor*) no Capítulo 5.

TABELA 1.3-3 Alguns comandos de plotagem do MATLAB

Comando	Descrição
`[x,y] = ginput(n)`	Habilita o mouse a capturar *n* pontos de um plotagem, e retorna as coordenadas *x* e *y* nos vetores `x` e `y`, ambos com tamanho *n*.
`grid`	Insere linhas de grid na plotagem.
`gtext('text')`	Habilita a inserção e o posicionamento de texto com o mouse.
`plot(x,y)`	Gera uma plotagem do arranjo `y` *versus* o arranjo `x` em eixos lineares.
`title('text')`	Insere texto de título na parte superior da plotagem.
`xlabel('text')`	Adiciona um rótulo ao eixo horizontal (as abscissas).
`ylabel('text')`	Adiciona um rótulo ao eixo vertical (as ordenadas).

Teste seus conhecimentos

T1.3-3 Utilize o MATLAB para plotar a função $s = 2\,\text{sen}(3t + 2) + \sqrt{5t + 1}$ ao longo do intervalo $0 \leq t \leq 5$. Insira um título na plotagem e rotule adequadamente os eixos. A variável *s* representa a velocidade em pés por segundo; a variável *t* representa o tempo em segundos.

T1.3-4 Utilize o MATLAB para plotar as funções $y = 4\sqrt{6x + 1}$ e $z = 5e^{0,3x} - 2x$ ao longo do intervalo $0 \leq x \leq 1,5$. Rotule adequadamente a plotagem e cada curva. As variáveis y e z representam forças em newtons; a variável x representa a distância em metros.

Equações algébricas lineares

Você pode utilizar o operador divisão à esquerda (\) no MATLAB para solucionar conjuntos de equações algébricas lineares. Por exemplo, considere o conjunto

$$\begin{aligned} 6x + 12y + 4z &= 70 \\ 7x - 2y + 3z &= 5 \\ 2x + 8y - 9z &= 64 \end{aligned}$$

Para solucionar tais conjuntos no MATLAB, você deve criar dois arranjos; nós os denominaremos `A` e `B`. O número de linhas de `A` é igual ao número de equações e o número de colunas é igual ao número de variáveis. As linhas de `A` devem conter os coeficientes de x, y e z, nesta ordem. Nesse exemplo, a primeira linha de `A` deve ser 6, 12, 4; a segunda linha deve ser 7, −2, 3; e a terceira linha deve ser 2, 8, −9. O arranjo `B` contém as constantes que estão no lado direito da equação; ele tem uma única coluna e tantas linhas quantas forem as equações. Nesse exemplo, a primeira linha de B é 70, a segunda é 5 e a terceira é 64. A solução é obtida digitando-se `A\B`. A sessão é

```
>>A = [6,12,4;7,-2,3;2,8,-9];
>>B = [70;5;64];
>>Solution = A\B
Solution =
     3
     5
     -2
```

A solução é $x = 3$, $y = 5$ e $z = -2$.

Esse método funciona bem quando o conjunto de equações possui uma única solução. Para aprender a lidar com problemas que tenham mais de uma solução (ou talvez nenhuma solução!), veja o Capítulo 8.

Teste seus conhecimentos

T1.3-5 Utilize o MATLAB para resolver o seguinte conjunto de equações:

$$\begin{aligned} 6x - 4y + 8z &= 112 \\ -5x - 3y + 7z &= 75 \\ 14x + 9y - 5z &= -67 \end{aligned}$$

(Resposta: $x = 2$, $y = -5$, $z = 10$)

1.4 Arquivos de script (*script files*) e o Editor/Debugador (*Editor/Debugger*)

Você pode realizar operações no MATLAB de duas maneiras:

1. No modo interativo, em que todos os comandos são inseridos diretamente na janela de Comandos.
2. Rodando um programa no MATLAB armazenado em um arquivo de script (*script file*). Esse tipo de arquivo contém comandos do MATLAB, portanto, rodá-lo é equivalente a digitar todos esses comandos, um por vez, no prompt da janela de Comandos. Você pode rodar o arquivo digitando o seu nome no prompt da janela de Comandos.

Quando o problema a ser solucionado necessita de muitos comandos, de um conjunto repetido de comandos ou envolve arranjos com muitos elementos, o modo interativo é inconveniente. Felizmente, o MATLAB permite a você escrever os seus próprios programas para evitar essa dificuldade. Você escreve e salva programas no MATLAB em arquivos M (M-files), os quais possuem a extensão `.m`; por exemplo, `program1.m`.

ARQUIVO DE SCRIPT

O MATLAB utiliza dois tipos de arquivos M: *arquivos de script* e *arquivos de funções*. Você pode utilizar o Editor/Debugador (*Editor/Debugger*) interno do MATLAB para criar arquivos M. Uma vez que eles contêm comandos, os arquivos de script são algumas vezes chamados de arquivos de *comandos*. Arquivos de funções são discutidos no Capítulo 3.

Criando e utilizando um arquivo de script

COMENTÁRIO

O símbolo `%` designa um *comentário*, que não é executado pelo MATLAB. Comentários são utilizados principalmente em arquivos de script com o propósito de documentar o arquivo. O símbolo de comentário pode ser inserido em qualquer lugar da linha. O MATLAB ignora tudo que estiver escrito à direita do símbolo `%`. Por exemplo, considere a seguinte sessão.

```
>>% Isto é um comentário.
>>x = 2+3 % Isto também é.
x =
   5
```

Note que a parte da linha antes do sinal `%` é executada para calcular `x`.

Aqui está um exemplo simples que ilustra como criar, salvar e rodar um arquivo de script, utilizando o Editor/Debugador interno do MATLAB. Entretanto, você pode utilizar outro editor de textos para criar o arquivo. O arquivo de amostra é apresentado a seguir. Ele calcula o cosseno da raiz quadrada de alguns números e exibe o resultado na tela.

```
% Program Example_1.m
% Este programa calcula o cosseno da
% raiz quadrada e exibe o resultado.
x = sqrt(13:3:25);
y = cos(x)
```

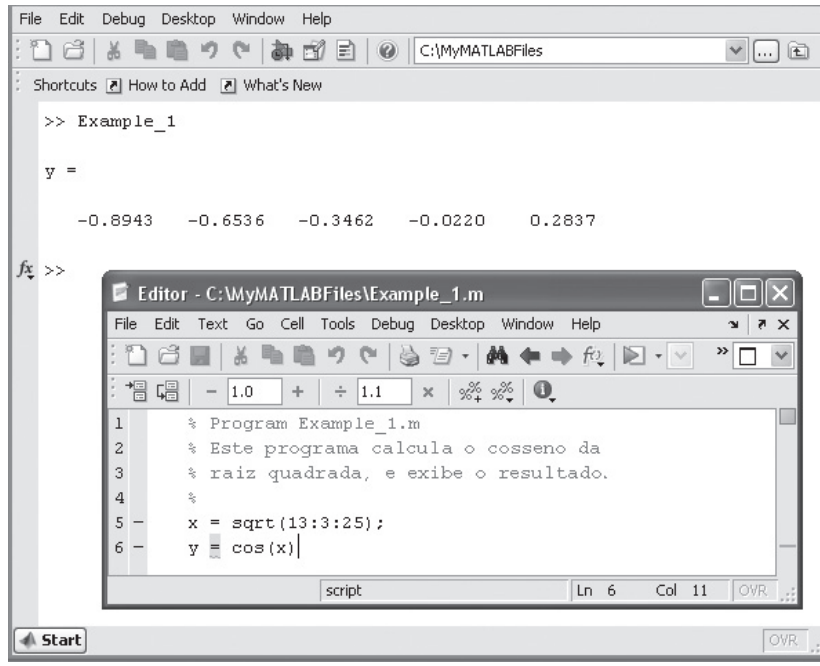

FIGURA 1.4-1 A janela de comandos do MATLAB com o Editor/Debugador aberto.

Para criar esse novo arquivo M na janela de Comandos, selecione **New** no menu **File** e clique em **Blank M-File**. Você verá então uma nova janela de edição. Essa é a janela do Editor/Debugador, mostrada na Figura 1.4-1. Digite no arquivo conforme está na figura. Você pode utilizar o teclado e o menu **Edit** no Editor/Debugador como o faria na maioria dos processadores de texto para criar e editar um arquivo. Quando tiver terminado, selecione **Save** no menu **File** do Editor/Debugador. Na caixa de diálogo que aparece, substitua o nome padrão dado (normalmente `Untitled`) pelo nome `Example_1` e clique em **Save**. O Editor/Debugador automaticamente incluirá a extensão `.m` e salvará o arquivo no diretório atual do MATLAB, que, por enquanto, consideraremos que está no disco rígido.

Uma vez que o arquivo foi salvo, digite na janela de Comandos do MATLAB o nome do arquivo de script `Example_1` para executar o programa. Você verá o resultado exibido na janela de Comandos. A Figura 1.4-1 mostra uma tela contendo a exibição resultante da janela de Comandos e o Editor/Debugador aberto para exibir o arquivo de script.

Utilização eficaz de arquivos de script

Crie arquivos de script para evitar a necessidade de redigitar procedimentos longos e comumente utilizados. Aqui estão alguns outros pontos que você deve ter em mente quando utilizar arquivos de script:

1. O nome de um arquivo de script deve seguir a convenção do MATLAB para nomeação de variáveis.

2. Lembre-se de que, quando você digitar o nome de uma variável no prompt da janela de Comandos, o MATLAB exibirá o valor daquela variável. Assim, não dê a um arquivo de script o mesmo nome de uma variável que ele calcula, porque o MATLAB não será capaz de executar o arquivo de script mais de uma vez, a não ser que você apague a variável.
3. Não dê a um arquivo de script o mesmo nome de um comando ou função do MATLAB. Você pode checar se o comando, função ou nome do arquivo já existe utilizando o comando `exist`. Por exemplo, para ver se uma variável `example1` já existe, digite `exist('example1')`; será retornado 0 se a variável não existir, ou 1 se existir. Para ver se um arquivo M `example1.m` já existe, digite `exist('example.m','file')` *antes* de criar o arquivo; será retornado 0 se o arquivo não existir, ou 2 se ele existir. Finalmente, para ver se uma função interna `example1` já existe, digite `exist('example1','builtin')` *antes* de criar o arquivo; será retornado 0 se a função interna não existir, ou 5 se ela existir.

Note que nem todas as funções fornecidas pelo MATLAB são funções internas. Por exemplo, a função `mean.m` é fornecida mas não é uma função interna. O comando `exist('mean.m','file')` retornará 2, mas o comando `exist('mean','builtin')` retornará 0. Você pode pensar nas funções internas como primitivas que formam a base para outras funções do MATLAB. Você não pode visualizar o arquivo inteiro de uma função interna em um editor de textos, apenas os comentários.

Debugando arquivos de script

DEBUGANDO

Debugar um programa é o processo de encontrar e remover os "bugs", ou erros. Tais erros normalmente estão em uma das seguintes categorias.

1. Erros de sintaxe, como a omissão de parênteses ou vírgulas ou a digitação incorreta do nome de um comando. O MATLAB normalmente detecta os erros mais óbvios e exibe uma mensagem descrevendo o erro e a sua localização.
2. Erros devido a um procedimento matemático incorreto, chamados de *erros em tempo de execução*. Eles não ocorrem necessariamente toda vez que o programa é executado; sua ocorrência em geral depende dos dados de entrada. Um exemplo comum é a divisão por zero.

Para localizar um erro, tente o seguinte:

1. Sempre teste seu programa com uma versão simples do problema, cuja resposta pode ser checada por cálculos à mão.
2. Exiba alguns cálculos intermediários removendo o sinal de ponto e vírgula no final de sentenças.
3. Utilize as funcionalidades próprias para debugar do Editor/Debugador, que serão introduzidas no Capítulo 4. Entretanto, uma vantagem do MATLAB é que ele requer programas relativamente simples para realizar muitos tipos de tarefas. Deste modo, você provavelmente não precisará utilizar o Debugador para os problemas encontrados neste texto.

Estilo de programação

Comentários podem ser inseridos em qualquer lugar no arquivo de script. Entretanto, uma vez que a primeira linha de comentário antes de qualquer sentença executável é a linha procurada pelo comando `lookfor`, que será discutido mais adiante neste capítulo, insira palavras-chave que descrevem o arquivo de script nessa primeira linha (chamada de linha H1). Uma estrutura sugerida para um arquivo de script é a seguinte.

1. *Seção de comentários* Nesta seção, insira sentenças de comentário que contenham
   a. O nome do programa e algumas palavras-chave na primeira linha.
   b. A data de criação e o nome do criador na segunda linha.
   c. As definições dos nomes de variável para cada entrada e saída. Divida esta seção em pelo menos duas subseções, uma para dados de entrada e outra para dados de saída. Uma terceira seção opcional pode incluir definições de variáveis utilizadas nos cálculos. *Lembre-se de incluir as unidades de medida para todas as variáveis de entrada e de saída!*
   d. O nome de toda função definida pelo usuário chamada pelo programa.
2. *Seção de entrada* Nesta seção, insira os dados de entrada e/ou as funções de entrada que possibilitem a inserção de dados. Inclua comentários onde for apropriado para documentação.
3. *Seção de cálculos* Insira os cálculos nesta seção. Inclua comentários onde for apropriado para documentação.
4. *Seção de saída* Nesta seção, insira as funções necessárias para apresentar as saídas na forma desejada. Por exemplo, esta seção pode conter funções para exibir a saída na tela. Inclua comentários onde for apropriado para documentação.

Os programas neste texto frequentemente omitem alguns desses elementos para economizar espaço. Aqui, a discussão do texto associada com os programas fornece a documentação necessária.

Controlando entradas e saídas

O MATLAB fornece alguns comandos úteis para a obtenção de entradas do usuário e para a formatação da saída (os resultados obtidos pela execução dos comandos do MATLAB). A Tabela 1.4-1 resume esses comandos.

A função `disp` (abreviação de "display") pode ser utilizada para exibir o valor de uma variável, mas não o seu nome. A sua sintaxe é `disp(A)`, em que `A` representa o nome de uma variável no MATLAB. A função `disp` também pode exibir texto, como uma mensagem para o usuário. Você deve escrever o texto entre aspas simples. Por exemplo, o comando `disp('A velocidade prevista é:')` faz com que essa mensagem apareça na tela. Este comando pode ser utilizado com a primeira forma da função `disp` em um arquivo de script da seguinte maneira (considerando que o valor de `Velocidade` é 63):

```
disp('A velocidade prevista é:')
disp(velocidade)
```

Quando o arquivo é rodado, essas linhas produzem o seguinte na tela:

TABELA 1.4-1 Comandos de entrada/saída

Comando	Descrição
`disp(A)`	Exibe o conteúdo, mas não o nome, do arranjo A.
`disp('text')`	Exibe o texto entre aspas simples.
`format`	Controla o formato de exibição da saída na tela (ver Tabela 1.1-5).
`x = input('text')`	Exibe o texto entre aspas simples na tela, espera pela entrada do usuário a partir do teclado, e armazena o valor em `x`.
`x = input('text','s')`	Exibe o texto entre aspas simples na tela, espera pela entrada do usuário a partir do teclado, e armazena a entrada como uma string em `x`.
`k=menu('title','option1', 'option2',...)`	Exibe um menu cujo título é a variável string `'title'` e cujas opções são `'option1', 'option2'`, e assim por diante.

```
A velocidade prevista é:
   63
```

A função `input` exibe o texto na tela, espera que o usuário entre com algo a partir do teclado e então armazena a entrada na variável especificada. Por exemplo, o comando `x = input('Por favor, entre com o valor de x:')` faz com que a mensagem apareça na tela. Se você digitar 5 e pressionar **Enter**, à variável `x` será atribuído o valor 5.

VARIÁVEL STRING

Uma variável string é composta de texto (caracteres alfanuméricos). Se você quiser armazenar um texto de entrada como uma variável string, utilize a outra forma do comando de entrada. Por exemplo, o comando `Calendário = input('Entre com o dia da semana:','s')` induz a entrar com o dia da semana. Se você digitar `Quarta-feira`, esse texto será armazenado na variável string `Calendario`.

Utilize a função `menu` para gerar um menu de opções para a entrada do usuário. Sua sintaxe é

```
k = menu('title','option1','option2',...)
```

A função exibe o menu cujo título é a variável string `'title'` e cujas opções são as variáveis string `'option1'`, `'option2'`, e assim por diante. O valor retornado de `k` é 1, 2,..., dependendo se você clicar no botão para `'option1'`, `'option2'`, e assim por diante. Por exemplo, o seguinte script utiliza um menu para selecionar um marcador de dados para um gráfico, presumindo que os arranjos `x` e `y` já existam.

```
k = menu('Escolha um marcador de dados','o','*','x');
type = ['o','*','x'];
plot(x,y,x,y,type(k))
```

Teste seus conhecimentos

T1.4-1 A área de superfície A de uma esfera depende do seu raio r da seguinte maneira: $A = 4\pi r^2$. Escreva um arquivo de script que induza o usuário a entrar com um valor de raio, calcule a área de superfície A e exiba o resultado.

Exemplo de um arquivo de script

A seguir, apresentamos um exemplo simples de um arquivo de script que mostra o estilo de programação apresentado nesta seção. A velocidade v de um objeto em queda livre (velocidade inicial nula, portanto) é dada como uma função do tempo t por $v = gt$, em que g é a aceleração devido à gravidade. Nas unidades do SI, $g = 9{,}81$ m/s^2. Nós queremos calcular e plotar v como uma função de t para $0 \leq t \leq t_{final}$, em que t_{final} é o tempo final definido pelo usuário. O arquivo de script é o seguinte:

```
% Programa Velocidade_de_Queda.m: plota a velocidade de queda
% de um objeto.
% Criado em 01/03/2009, por W. Palm III
%
% Variável de entrada:
% tfinal = tempo final (em segundos)
%
% Variáveis de saída:
% t = arranjo de instantes de tempo em que a velocidade é
% calculada (segundos)
% v = arranjo de velocidades (metros/segundo)
%
% Valor de parâmetro:
g = 9.81; % Aceleração em unidades do SI
%
% Seção de entrada:
tfinal = input('Entre com o tempo final em segundos:');
%
% Seção de cálculo:
dt = tfinal/500;
t = 0:dt:tfinal; % Cria um arranjo com 501 valores de tempo.
v = g*t;
%
% Seção de saída:
plot(t,v) xlabel('Tempo(segundos)'), ylabel('Velocidade (metros/...
segundo)')
```

Após criar esse arquivo, salve-o com o nome `Velocidade_de_Queda.m`. Para rodá-lo, digite `Velocidade_de_Queda` (sem o `.m`) no prompt da janela de Comandos. Você será então solicitado a entrar com um valor para t_{final}. Depois de entrar com o valor e pressionar **Enter**, você verá a plotagem na tela.

1.5 O Sistema de Ajuda do MATLAB

Para explorar as funcionalidades mais avançadas do MATLAB não abordadas neste livro, você precisará saber como utilizar de modo eficaz o Sistema de Ajuda (*Help System*) do MATLAB. O MATLAB apresenta as seguintes opções de obtenção de ajuda para utilizar os produtos MathWorks: