CAPÍTULO 4

Rotinas (M-Files)

Todos os comandos do MATLAB são executados na janela Command Window. Ao longo deste livro, embora cada comando do MATLAB possa ser executado desse modo, não é conveniente usar a janela Command Window para executar muitos comandos em série – especialmente se eles estiverem encadeados logicamente entre si, isto é, constituírem um programa. Muitas vezes, tentar resolver um determinado problema no MATLAB digitando muitos comandos consecutivos pode ser difícil ou até mesmo impossível. Além disso, a janela Command Window não é interativa. Significa que cada vez que a tecla **Enter** é pressionada, somente o último comando digitado é executado, sendo que o restante do trabalho torna-se inalterável. Se forem necessárias modificações ou correções em um determinado comando da série executada e os resultados gerados por ele afetarem comandos em uma certa seqüência encadeada, toda a seqüência de comandos deve ser selecionada e executada novamente na linha do *prompt* da janela Command Window, isto é, um comando da série deve ser chamado e executado por vez.

Uma maneira diferente de executar comandos no MATLAB é primeiro criar um arquivo com uma lista de comandos, salvá-la e, então, rodar o arquivo. Quando o arquivo é chamado, os comandos vão sendo executados na ordem em que estão listados no arquivo. Caso sejam necessárias correções ou modificações nesse arquivo, basta abri-lo, modificar os comandos ou a seqüência lógica de interesse, salvá-lo e testar o novo arquivo. Os arquivos criados com esse propósito são denominados rotinas, M-*files* ou *script files*.

4.1 OBSERVAÇÕES QUANTO AO USO DE ROTINAS

- Rotina é uma seqüência de comandos do MATLAB (também denominada programa).

- Quando uma rotina é testada, o MATLAB executa os comandos na ordem em que eles foram escritos.
- Quando uma rotina possui um comando que gera uma saída (por exemplo, exibição de gráfico, veja Capítulo 5), essa saída é exibida na janela Command Window.
- O uso de rotinas é bastante conveniente, porque elas podem ser reeditadas e executadas muitas vezes.
- É possível digitar e/ou editar rotinas em qualquer editor de texto e, em seguida, colá-las no editor do MATLAB.
- As rotinas do MATLAB são denominadas M-*files*, porque lhes são atribuídas extensões .m ao serem salvas.

4.2 CRIANDO E SALVANDO UMA ROTINA

No MATLAB, rotinas são criadas e editadas na janela Editor/Debugger Window. Essa janela é aberta a partir do Command Window. No menu **File**, selecione **New** e, depois, a opção **M-file**. Um exemplo de janela Editor/Debugger Window é visto na Figura 4-1.

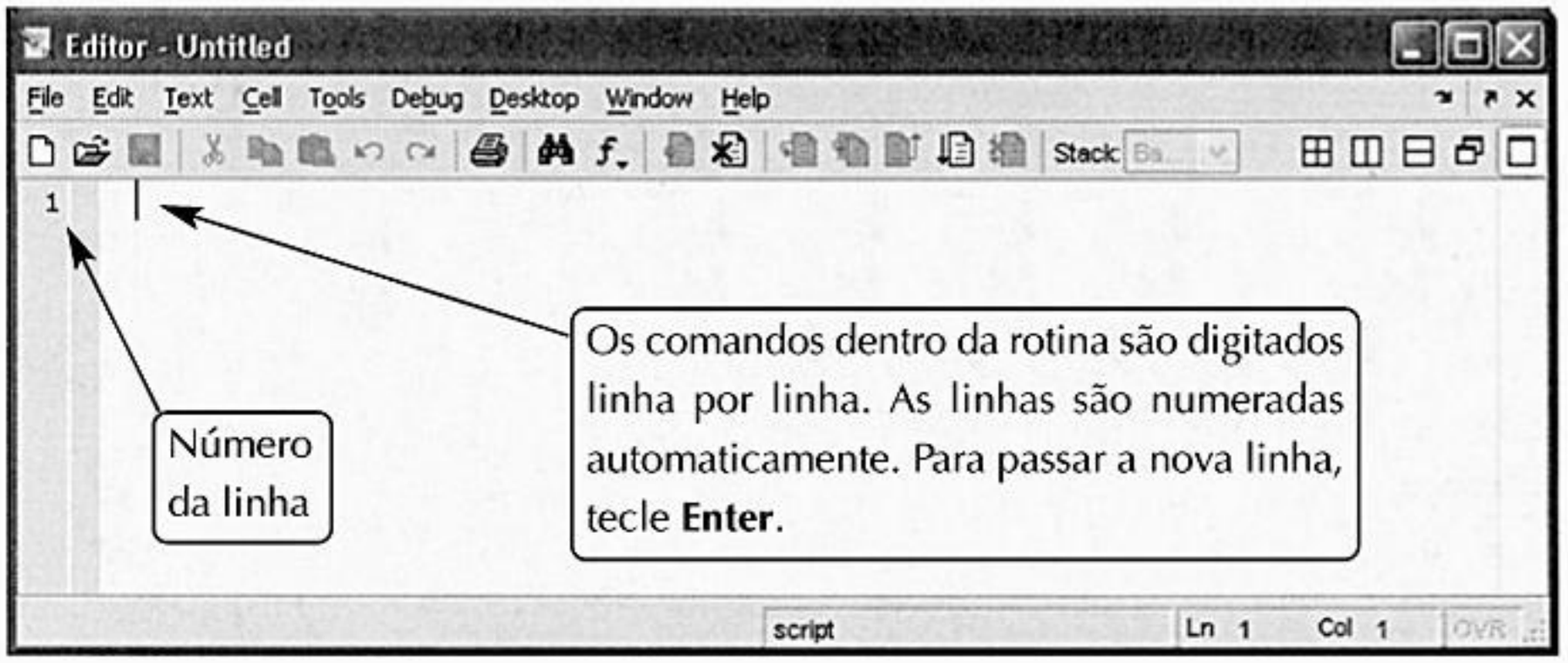


FIGURA 4-1 A janela Editor/Debugger.

Uma vez aberta a janela, os comandos da rotina devem ser digitados linha por linha. O MATLAB enumera automaticamente cada nova linha toda vez que a tecla **Enter** é pressionada. Também é possível digitar os comandos em outro editor de textos e, em seguida, colá-los no Editor/Debugger Window. Uma pequena rotina é mostrada na Figura 4-2. Numa rotina, as primeiras linhas de programa geralmente são dedicadas aos comentários (que não são executados porque diferenciam-se do programa pelo caractere %) e dão uma breve descrição da rotina que se inicia.

Antes de uma rotina ser executada, ela deve ser salva. Para isso, escolha a opção **Save as...** a partir do menu **File** e selecione o caminho e a localização da unidade de armazenamento, preenchendo o campo nome da rotina que está sendo criada. As regras para

se atribuir um nome a uma rotina são as mesmas das variáveis (deve ser iniciada com uma letra, pode incluir dígitos, *underscore* e possuir no máximo 63 caracteres de extensão). Nomes de variáveis declaradas pelo usuário, comandos do MATLAB ou funções nativas não devem ser usados como nome de rotinas.

4.3 RODANDO UMA ROTINA

Uma rotina pode ser executada na janela Command Window por meio da digitação do respectivo nome na linha do *prompt*, seguida da tecla **Enter** ou, então, diretamente da janela Editor Window clicando no ícone **Run** (veja a Figura 4-2). Entretanto, antes disso o usuário deve certificar-se de que o MATLAB consegue encontrar o caminho onde está o arquivo. Conforme o caso, o usuário pode selecionar um caminho diferente no campo Current Directory.

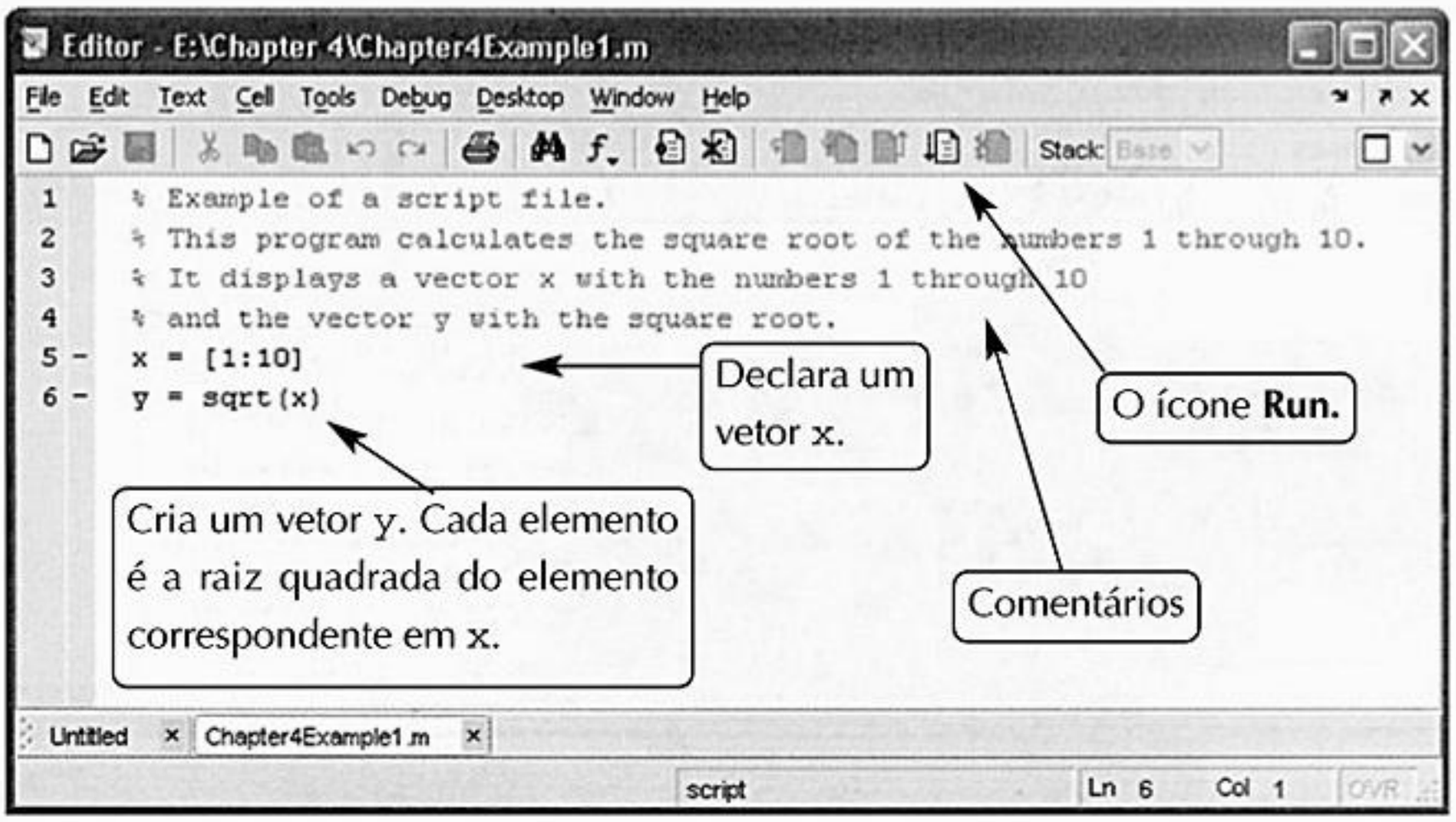


FIGURA 4-2 Um programa digitado na janela Editor/Debbugger.

4.3.1 O campo Current Directory

O diretório ou a pasta de trabalho é mostrado no campo Current Directory na barra de ferramentas da janela Command Window (veja a Figura 4-3). O diretório selecionado pode ser modificado nesse mesmo campo ou digitando-se o comando `cd` na linha do *prompt*, seguido do caminho onde se encontra o diretório desejado.

Uma vez que dois ou mais diretórios diferentes são usados em uma sessão, é possível permutar entre os diretórios no campo **Current Directory**. Além do mais, a janela Current Directory pode ser aberta a partir do menu **Desktop**. Essa janela mostra os detalhes dos arquivos em cada diretório e pode ser modificada escolhendo-se uma nova unidade e pasta onde os arquivos serão salvos.

Um modo alternativo de mudar o diretório ou pasta é usar o comando `cd` na janela Command Window. Para escolher uma nova unidade, digite `cd`, espaço e o nome do diretório seguido de dois pontos (:), pressionando **Enter** em seguida. Por exemplo, para mudar o diretório selecionado para a unidade A (geralmente o disco flexível de $3\frac{1}{2}$), di-

gite `cd A:`. Se uma rotina for salva em uma pasta dentro de uma unidade, o caminho completo dessa pasta deve ser especificado. Por exemplo, `cd ('D:\Capitulo_4')` configura o caminho da pasta Capitulo_4 na unidade D, supondo-se que essa pasta exista na unidade D. A seguir, temos um exemplo onde o diretório selecionado é a unidade A. Então, supondo-se que a rotina da Figura 4-2 tenha sido digitada e salva na unidade A como Capitulo4_Exemplo1.m, é executada na linha do *prompt* por meio do nome e da tecla **Enter**.

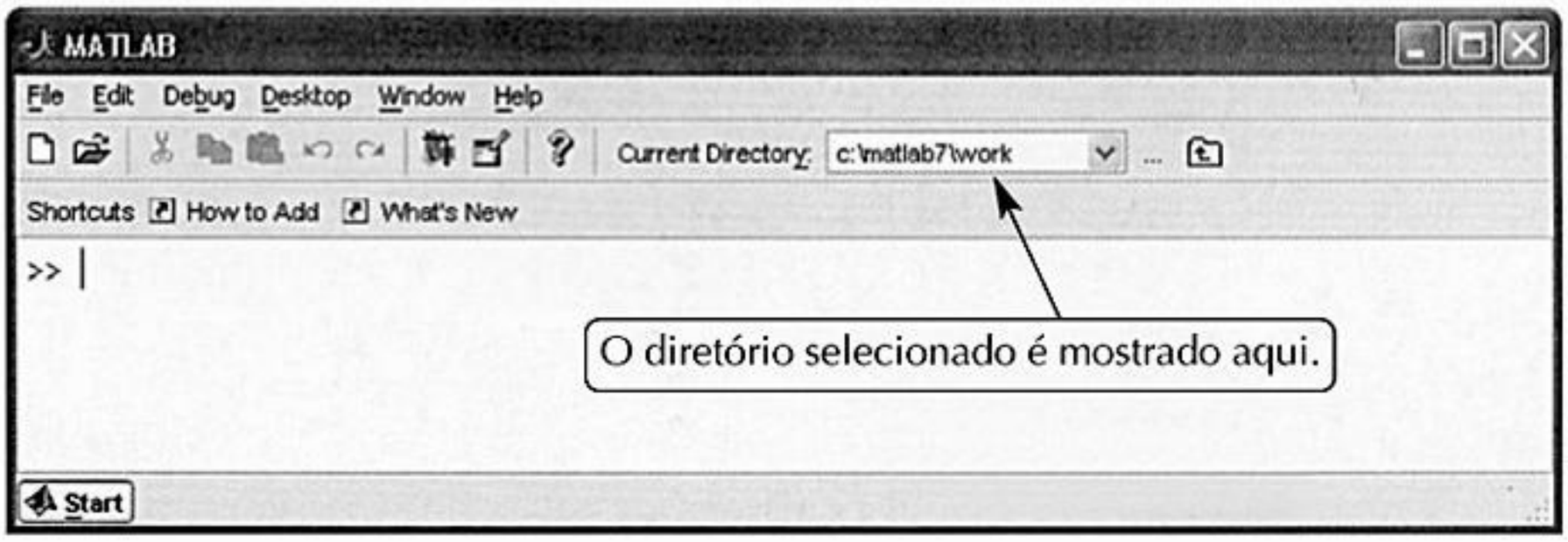


FIGURA 4-3 O campo Current Directory na janela Command Window.

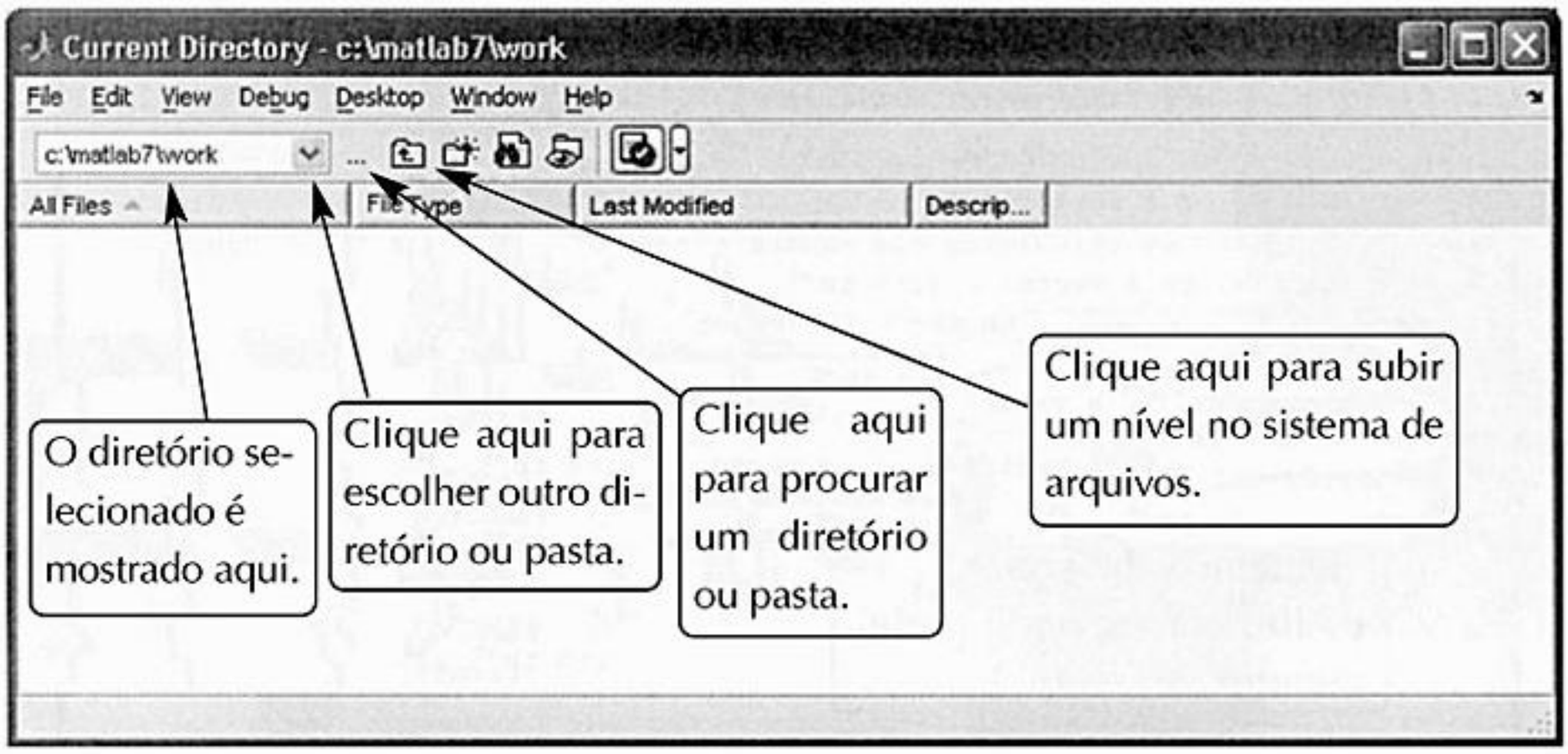
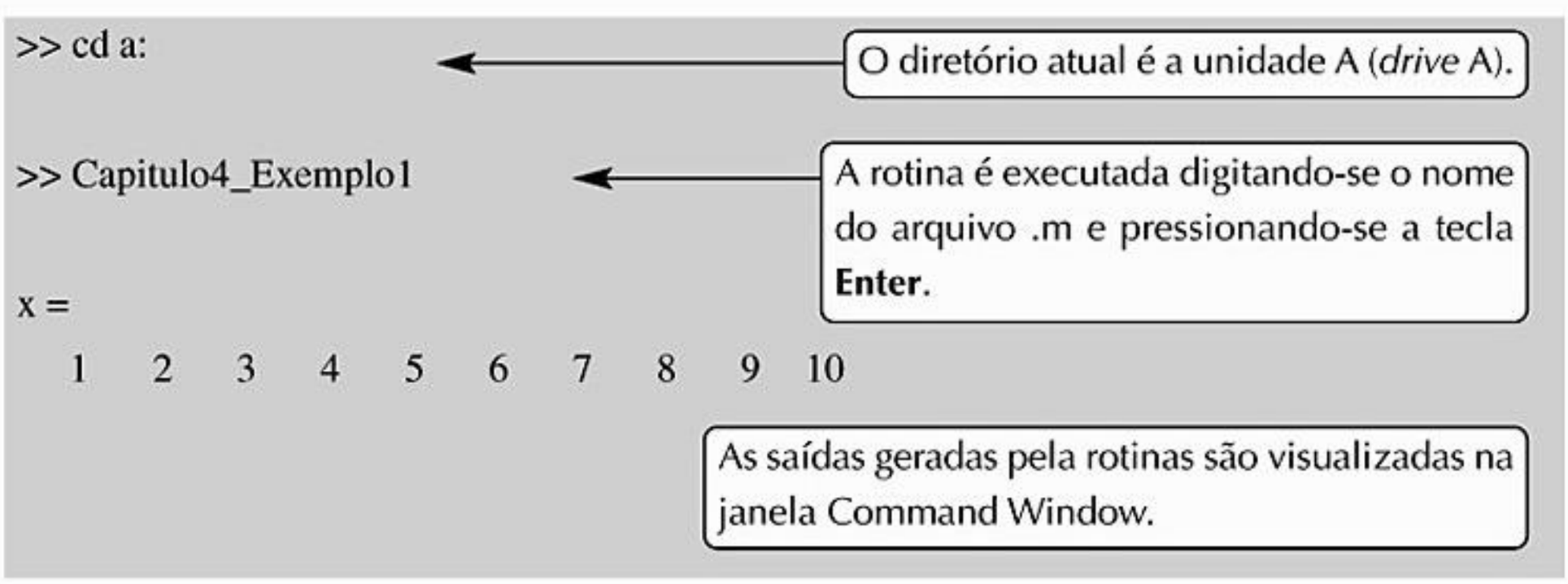


FIGURA 4-4 A janela Current Directory.



```
y =
Columns 1 through 8
   1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284

Columns 9 through 10
   3.0000    3.1623
```

4.3.2 Procurando um diretório ou uma pasta

Quando uma rotina ou uma função é executada, o MATLAB procura pelo arquivo correspondente nos diretórios que estão listados nos caminhos especificados. Os diretórios ou as pastas que estão incluídos nos caminhos procurados são exibidos na janela Set Path, que pode ser aberta selecionando-se **Set Path** no menu **File**. Uma vez aberta, podemos adicionar ou remover novos(as) diretórios (pastas) nessa janela.

4.4 VARIÁVEIS GLOBAIS

Variáveis globais são aquelas que, uma vez criadas em uma parte do MATLAB, são reconhecidas em todas as outras partes do programa. Esse é o caso para variáveis na janela Command Window e as rotinas, visto que ambas agem sobre as variáveis em uma mesma área de trabalho. Quando uma variável é declarada na janela Command Window, pode ser reconhecida e usada em uma rotina. A recíproca também é verdadeira. Em outras palavras, uma variável declarada na janela Command Window ou em uma rotina pode ser usada ou ter o valor modificado a qualquer momento por uma aplicação que faça uso dela.

Há diferentes tipos de arquivos no MATLAB, como os chamados arquivos de funções, que normalmente não compartilham as variáveis com outras partes do programa. Esse aspecto será explicado em detalhes mais adiante, no Capítulo 6 (Seção 6.3).

4.5 ENTRADAS EM ROTINAS

Quando uma rotina é executada, as variáveis usadas nos cálculos dentro do arquivo devem ser inicializadas. A atribuição de um valor a uma variável pode ser feita de três modos diferentes, dependendo de como e onde a variável foi declarada.

1. A variável foi declarada e inicializada dentro da rotina.

Neste caso, a atribuição de um valor à variável é parte integrante da rotina. Se o usuário desejar rodar o arquivo com valores diferentes para a(s) variável(is), o arquivo deve ser reeditado e a(s) variável(is) modificada(s). Então, após o arquivo ser salvo, podemos executá-lo novamente.

A seguir, temos um exemplo. A rotina (salva como Capitulo4_Exemplo2) calcula a média dos placares de pontos em três jogos.

```
% Esta rotina calcula a media dos placares em tres jogos.
% A atribuicao dos valores dos pontos e parte da rotina.
jogo1 = 75;
jogo2 = 93;
jogo3 = 68;
med_pontos = (jogo1 + jogo2 + jogo3)/3
```

As variáveis são inicializadas dentro da rotina.

Rodando o arquivo, a janela Command Window exibe:

```
>> Capitulo4_Exemplo2

med_pontos =
   78.6667
>>
```

Para chamar a rotina, basta digitar o nome do arquivo.

A variável `med_pontos` e o respectivo valor são exibidos na janela Command.

2. A variável foi declarada e inicializada na janela Command Window.

Neste caso, a atribuição de um valor à variável é feita na janela Command Window (lembre-se que as variáveis são entendidas dentro da rotina). Se o usuário quiser rodar a rotina com outros valores para a(s) variável(is), basta atribuí-los na janela Command Window e rodar o arquivo novamente.

No exemplo anterior, em que a rotina calculava a média dos escores de pontos em três jogos, a rotina (salva como Capitulo4_Exemplo3) é:

```
% Esta rotina calcula a media dos placares em tres jogos.
% A atribuicao dos valores dos pontos e parte da rotina.
% jogo1, jogo2 e jogo3 sao inicializadas na janela Command.

med_pontos = (jogo1 + jogo2 + jogo3)/3
```

Rodando o arquivo, a janela Command Window exibe:

```
>> jogo1 = 67;
>> jogo2 = 90;
>> jogo3 = 81;
>> Capitulo4_Exemplo3

med_pontos =
   79.3333
```

As variáveis são inicializadas na janela Command.

A rotina é executada na linha do *prompt*.

A saída `med_pontos` gerada pela rotina para o primeiro conjunto de valores das variáveis jogo1, jogo2 e jogo3 na janela Command.

```
>> jogo1 = 87;
>> jogo2 = 70;
>> jogo3 = 50;

>> Capitulo4_Exemplo3

med_pontos =
   69
>>
```

Novos valores são atribuídos às variáveis jogo1, jogo2 e jogo3.

A rotina é executada novamente.

A saída `med_pontos` gerada pela rotina para o segundo conjunto de valores das variáveis jogo1, jogo2 e jogo3 na janela Command.

3. A variável foi declarada na rotina, mas a especificação do valor é feita na janela Command Window quando a rotina for executada.

Neste caso, uma variável é declarada dentro do corpo da rotina e, rodando o arquivo, o usuário é solicitado a atribuir um valor à variável na janela Command Window. Para tanto, é necessário utilizar o comando `input` para gerenciar esse tipo de variável.

A forma geral do comando `input` é:

```
nome_da_variavel = input ('String com a mensagem a ser
                          exibida na janela Command Window')
```

Executando o comando `input` dentro da rotina, a *string* é exibida na janela Command Window. A *string* deve passar informações sobre o que será atribuído à variável. O usuário deve digitar o valor e pressionar **Enter**. Isso atribui valor à variável. Assim, como acontece com qualquer variável, a variável e o respectivo valor serão exibidos na janela Command Window, se não for utilizado ponto-e-vírgula no final do comando `input`. A seguir, temos uma rotina usando o comando `input` para receber o escore de pontos em cada jogo e calcular a média dos placares.

```
% Esta rotina calcula a media dos placares em tres jogos.
% O placar de cada jogo e atribuido as variaveis por meio do comando input.
jogo1 = input('Entre com o placar do primeiro jogo: ');
jogo2 = input('Entre com o placar do segundo jogo: ');
jogo3 = input('Entre com o placar do terceiro jogo: ');
med_pontos = (jogo1 + jogo2 + jogo3)/3
```

Rodando o arquivo (Capitulo4_Exemplo4), a janela Command Window exibe:

```
>> Capitulo4_Exemplo4
Entre com o placar do primeiro jogo:    67
Entre com o placar do segundo jogo:     91
Entre com o placar do terceiro jogo:    70
```

O computador exibe a mensagem. Em seguida, o valor do placar deve ser digitado pelo usuário e a tecla **Enter** deve ser pressionada.

```
med_pontos =
   76
>>
```

Nesse exemplo, um escalar é atribuído para cada variável. Entretanto, em geral vetores e/ou matrizes também podem receber os valores. Isso é feito digitando-se os elementos do arranjo na ordem em que eles aparecem (colchete à esquerda, digite linha a linha e depois feche o colchete à direita).

Outro uso freqüente do comando `input` ocorre na atribuição de uma *string* a uma variável. É possível fazê-lo de duas formas. Na primeira, usamos o comando da mesma forma recém-discutida e, quando a mensagem aparece no *prompt*, a *string* é digitada entre duas aspas simples (do mesmo modo que uma *string* é atribuída a uma variável sem o comando `input`). A segunda forma requer uma opção no comando `input` que declara os caracteres a serem digitados como uma *string*. A forma geral do comando é:

```
nome_da_variavel = input ('mensagem do prompt', 's')
```

onde a opção `'s'` dentro do comando define os caracteres que serão digitados como uma *string*. Nesse caso, ao aparecer a mensagem no *prompt*, o texto é digitado sem as aspas e é atribuído à variável como uma *string*. Um exemplo do uso do comando `input` com essa opção é mostrado no Problema-exemplo 7-4.

4.6 COMANDOS DE SAÍDA

Conforme foi discutido anteriormente, o MATLAB gera automaticamente uma saída quando alguns comandos são executados. Por exemplo, quando é atribuído algum valor a uma variável ou o nome de uma variável inicializada previamente é digitado na linha do *prompt* e a tecla **Enter** é pressionada, o MATLAB exibe o nome da variável e o seu respectivo valor. Esse tipo de saída é ocultada simplesmente digitando-se um ponto-e-vírgula no final do comando. Ademais, o MATLAB possui diversos comandos que podem ser utilizados para gerar outros tipos de saídas padronizadas. O conteúdo exibido pode ser uma mensagem informativa, dados numéricos e/ou gráficos. Dois comandos típicos na geração de saídas no MATLAB são o `disp` e o `fprintf`. O comando `disp` exibe uma saída padronizada na tela; o comando `fprintf`, além de gerar uma saída na tela, permite salvá-la em um determinado arquivo. Esses comandos podem ser usados na janela Command Window, em uma rotina e, como veremos nos próximos capítulos, na declaração de funções. Quando esses comandos são utilizados em rotinas, as saídas produzidas são mostradas na janela Command Window.

4.6.1 O comando `disp`

O comando `disp` exibe o conteúdo de uma variável, mas ocultando o nome da mesma, e/ou mostra *strings* na forma de textos. A forma geral do comando `disp` é:

```
disp (nome de uma variável) ou disp ('string de texto')
```

- Toda vez que um comando `disp` é executado, a saída que ele produz é mostrada numa nova linha. Por exemplo:

```
>> abc = [5 9 1; 7 2 4];
>> disp(abc)
    5     9     1
    7     2     4

>> disp('O problema nao possui solucao.')

O problema nao possui solucao.
>>
```

A variável `abc` é inicializada com uma matriz 2 × 3.

O comando `disp` é utilizado para exibir a matriz `abc`.

A matriz exibida sem o nome da variável `abc`.

O comando `disp` é utilizado para mostrar uma mensagem.

O próximo exemplo ensina como usar o comando `disp` em uma rotina para calcular a média dos escores de pontos em três jogos.

```
% Esta rotina calcula a media dos placares em tres jogos.
% O placar de cada jogo e atribuido as variaveis por meio do
% comando input.
% O comando disp e utilizado para mostrar a saida.

jogo1 = input('Entre com o placar do primeiro jogo: ');
jogo2 = input('Entre com o placar do segundo jogo: ');
jogo3 = input('Entre com o placar do terceiro jogo: ');
med_pontos = (jogo1 + jogo2 + jogo3)/3;
disp('')
disp('A media dos pontos dos tres jogos e: ')
disp('')
disp(med_pontos)
```

Salta uma linha.

Exibe o texto indicando média.

Salta uma linha.

Exibe o resultado da média.

Quando esse arquivo (Capitulo4_Exemplo5) for executado, a saída padrão na janela Command Window será:

```
>> Capitulo4_Exemplo5
Entre com o placar do primeiro jogo: 89
Entre com o placar do segundo jogo: 60
Entre com o placar do terceiro jogo: 82
```

Quebra de linha.

```
A media dos pontos dos tres jogos e:                    ← A linha texto é exibida.

   77                                                   ← Outra quebra de linha.
                                                        ← O conteúdo da variável med_pontos é exibido.
```

- Apenas uma variável por vez pode ser utilizada em um comando `disp`. Se os elementos de duas variáveis forem necessários simultaneamente na tela, uma nova variável contendo os elementos dessas duas variáveis deverá ser declarada antes de se utilizar o comando `disp`.

Em muitas situações, é interessante apresentar números na forma de tabela. Nesses casos, primeiro declare um arranjo com os números desejados e, em seguida, use o comando `disp` para mostrar o arranjo na tela. Podemos criar ainda cabeçalhos para as colunas com esse comando. Como o comando `disp` não permite que o usuário formate a saída (a largura das colunas e a distâncias entre elas), a posição do cabeçalho deve ser ajustada às colunas por meio de espaços. No exemplo a seguir, uma rotina ensina como exibir, em uma tabela, os dados populacionais do Capítulo 2.

```
yr = [1984  1986  1988  1990  1992  1994  1996];      ← Os dados populacionais são recebidos em dois vetores linha.
pop = [127  130  136  145  158  178  211];

tabelaYP(:,1) = yr';                                   ← yr é colocado na primeira coluna da matriz tabelaYP.

tabelaYP(:,2) = pop';                                  ← pop é colocado na primeira coluna da matriz tabelaYP.

disp('      ANO    POPULACAO')                         ← Exibe o cabeçalho na primeira linha da tabelaYP.

disp('           (MILHOES)')                           ← Exibe a unidade na segunda linha da tabelaYP.


disp('')                                               ← Salta uma linha

disp(tabelaYP)                                         ← Exibe o conteúdo da matriz tabelaYP.
```

Quando essa rotina (salva como PopTable) é executada, a saída visualizada na janela Command Window será:

```
> PopTable
      ANO       POPULACAO                              ← A matriz tabelaYP é exibida.
                (MILHOES)
```

```
1984    127
1986    130
1988    136
1990    145
1992    158
1994    178
1996    211
```

A matriz `tabelaYP` é exibida.

Outro exemplo de tabela formatada pode ser encontrado no Problema-exemplo 4-3. Além disso, tabelas também podem ser criadas e exibidas a partir do comando explicado a seguir, o `fprintf`.

4.6.2 O comando `fprintf`

Já foi mencionado que o comando `fprintf` pode ser utilizado para gerar saídas (texto e dados) na tela ou para salvá-las em um arquivo. Ao contrário do `disp`, nesse comando a saída pode ser formatada pelo usuário. Por exemplo, texto e dados numéricos das variáveis podem ser combinados e exibidos em uma linha. Além disso, o formato dos números também pode ser controlado.

Com tantas opções disponíveis é de se esperar que o comando `fprintf` disponha de mais opções no argumento. Para evitar confusões, apresentaremos o comando paulatinamente. Em primeiro lugar, mostraremos como usá-lo para mostrar mensagens na forma de textos, combinar texto e dados numéricos e, por último, como salvar a saída em um arquivo.

Usando o comando `fprintf` para exibir textos:

Para exibir textos, a forma geral do comando `fprintf` é:

```
fprintf ('string de texto')
```

Por exemplo:

```
fprintf ('O problema, como esta, nao possui solucao. Por favor, verifique o problema.')
```

Se essa linha fizer parte de uma rotina, ao executá-la teremos a seguinte mensagem:

```
O problema, como esta, nao possui solucao. Por favor, verifique o problema.
```

Ao utilizar o comando `fprintf` é possível promover quebras de linha em qualquer instância da *string*. Para tanto, basta inserir `\n` (barra invertida n) antes da quebra de linha. Por exemplo, inserindo `\n` após a primeira sentença do exemplo acima, temos:

```
fprintf ('O problema, como esta, nao possui solucao. \nPor favor, verifique o problema. ')
```

Ao executá-la, teremos a seguinte mensagem:

```
O problema, como esta, nao possui solucao.
Por favor, verifique o problema.
```

O caractere `\n` é denominado caractere de escape. Deve ser utilizado sempre que for necessário controlar o formato das mensagens na tela. Outros caracteres para modificação do modo de exibição dos dados são:

`\b`	Retrocesso
`\t`	Tabulação

Se um programa possuir mais de um comando `fprintf`, as saídas serão geradas continuamente à medida que cada `fprintf` é executado. Lembre-se que o `fprintf` não inicia uma nova linha ao ser executado. Por isso, se houver outros comandos entre dois `fprintf`, é essencial o uso do modificador `\n`. Observe a seguinte rotina:

```
fprintf('O problema, como esta, nao possui solucao. Por favor, verifique os dados de entrada.')
x = 6; d = 19 + 5*x;
fprintf('Tente rodar o programa mais tarde.')
y = d + x;
fprintf('Use diferentes valores para as entradas').
```

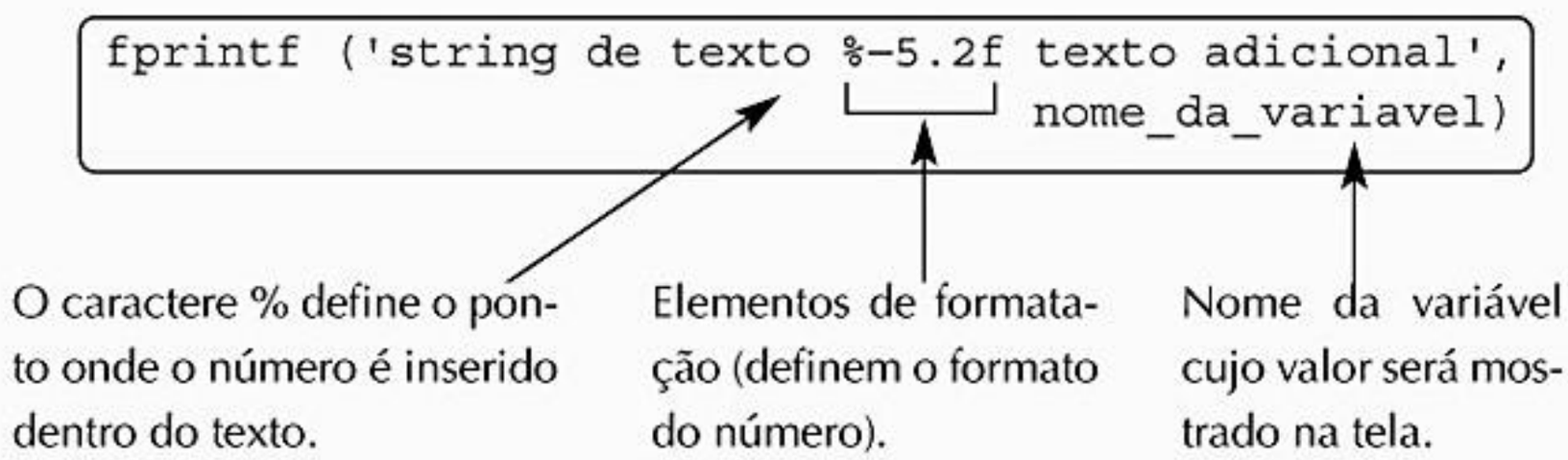
Executando a rotina, a saída na janela Command Window será:

```
O Problema, como esta, nao possui solucao. Por favor, verifique o programa. Tente rodar o programa a seguir. Use diferentes valores para a entrada.
```

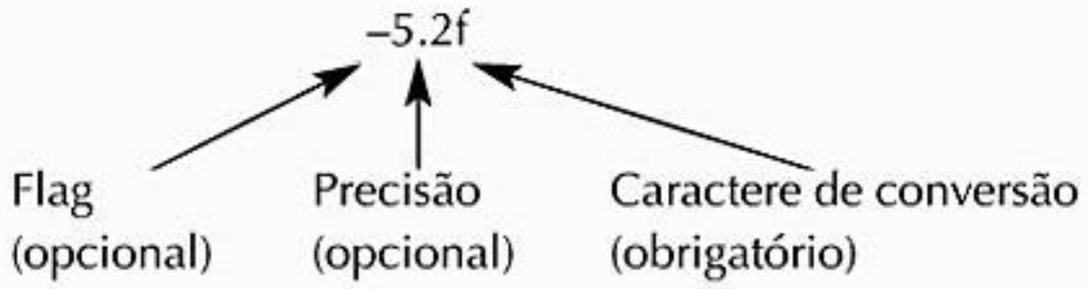
Assim, o exemplo anterior ressalta que, para iniciar uma nova linha com o comando `fprintf`, é necessário introduzir o modificador `\n` onde se deseja realizar a quebra de linha.

Usando o comando `fprintf` para combinar texto e números na tela:

A combinação de texto e número na função `fprintf` é feita no seguinte formato:



Os elementos de formatação são:



Os caracteres possíveis no campo de *flag* são:

Caractere utilizado no *flag*	**Descrição**
– (sinal negativo)	Justifica o número dentro do campo.
+ (sinal positivo)	Imprime um caractere de sinal (+ ou –) à frente do número.
0 (zero)	Adiciona zeros, se o número for menor que o especificado no campo de precisão.

O campo de precisão (5.2 no exemplo anterior) é opcional. O primeiro número (5, no exemplo) é o campo de largura. Especifica o número de dígitos que os números terão ao serem exibidos. Se o número a ser impresso for menor do que a quantidade selecionada, zeros serão introduzidos à esquerda desse número. O segundo número define a precisão em termos de casas decimais (2, no exemplo anterior). Especifica o número de dígitos a serem mostrados à direita da vírgula (ponto, no caso do MATLAB).

O último elemento de formatação é o caractere de conversão (obrigatório). Especifica a representação segundo a qual o número será mostrado na tela. Veja alguns dos possíveis caracteres de conversão:

e Número em notação científica com o "e" minúsculo (p. ex.: 1.709098e+001).
E Número em notação científica com o "E" maiúsculo (p. ex.: 1.709098E+001).
f Ponto flutuante decimal (p. ex.: 17.090980).
g "f " ou "e" (o mais curto).
G "f " ou "E" (o mais curto).
i Inteiro

Mais informações sobre os caracteres de conversão podem ser obtidas no Help do MATLAB. Tome a seguinte rotina como exemplo de combinação texto + dados no `fprintf`:

```
% Esta rotina calcula a media dos placares em tres jogos.
% O placar de cada jogo e atribuido as variaveis por meio do comando input.
% O comando fprintf e utilizado para mostrar a saida.
jogo(1) = input('Entre com os pontos do escore do primeiro jogo: ');
jogo(2) = input('Entre com os pontos do escore do segundo jogo: ');
jogo(3) = input('Entre com os pontos do escore do terceiro jogo: ');
```

```
med_pontos = mean(jogo);
fprintf('A media dos escores dos tres jogos foi %f pontos.', med_pontos)
```

Texto

% indica a posição do número.

Texto adicional.

Nome da variável cujo conteúdo será exibido.

Perceba que, além de usar a função `fprintf`, esse exemplo ilustra o uso de vetor, diferentemente do exemplo apresentado no início do capítulo, pois armazena os escores dos placares nos três primeiros elementos de um vetor, denominado `Jogo`, e a média dos escores era calculada usando-se a função `mean`. Rodando este último exemplo, a saída gerada na janela do Command Window é:

```
>> Capitulo4_Exemplo7
Entre com os pontos do escore do primeiro jogo  75
Entre com os pontos do escore do segundo jogo  60
Entre com os pontos do escore do terceiro jogo  81
A media dos escores dos tres jogos foi 72.00000.
>>
```

A saída padronizada gerada pelo comando `fprintf` combina texto e número.

O comando `fprintf` não fica limitado a um único elemento de formatação. Basta digitar % seguido de qualquer caractere de formatação no lugar onde se deseja introduzir os números. Em seguida, após a *string* de comando (seguida de vírgula), os nomes das variáveis são digitados na ordem em que devem ser inseridos no texto. Em geral, o formato do comando é:

```
fprintf ('...texto...%g...%g...%f...',variável1,
                 variável2,variável3)
```

O exemplo a seguir ilustra esse último formato de `fprintf`:

```
% Este programa calcula o alcance de lancamento de um projetil,
% dados a velocidade inicial e o angulo de lancamento.
% O comando fprintf e utilizado para exibir texto e numeros.

v = 1584; % Velocidade inicial (em km/h)
teta = 30; % angulo (em graus)
vms = v*1000/3600;
t = vms*sin(30*pi/180)/9.81;
```

Calcula a velocidade em m/s.

Determina o tempo até o máximo.

```
d = vms*cos(30*pi/180)*2*t/1000;                    Calcula o alcance do projétil.

fprintf('O projetil foi lancado a %3.2f graus, com velocidade inicial %4.2f km/h, e
atingira o solo a uma distancia de %g km.', teta, v, d)
```

Salve a rotina com o nome de Capitulo4_Exemplo8 e execute-a digitando esse nome na linha do *prompt* na janela Command Window. O resultado será:

```
>> Capitulo4_Exemplo8
O projetil foi lancado a 30 graus, com velocidade inicial 1584.00 km/h, e  atingira o solo a
uma distancia de 17.091 km.
>>
```

Observações adicionais quanto ao uso do comando `fprintf`***:***

- Caso seja necessário uma aspa simples no texto de saída, digite duas aspas simples dentro do corpo da *string*, no local onde deverá ser exibido no texto.
- O comando `fprintf` é "vetorizado". Significa que, se um arranjo do tipo vetor ou matriz estiver incluído no comando, haverá repetição desse comando até que o último elemento do vetor seja exibido na tela. Se a variável for do tipo matriz, os dados serão utilizados coluna por coluna.

Por exemplo, a rotina a seguir cria uma matriz `T` (2 × 5) cuja primeira linha são os números de 1 a 5 e, a segunda, as correspondentes raízes:

```
x = 1:5;                                            Declara um vetor x.

y = sqrt(x);                                        Declara um vetor y.

T = [x; y]                          Cria a matriz T (2 × 5); primeira linha é x e a segunda é y.

fprintf('Se o numero e %i, sua raiz quadrada e %f\n',T)

                                    O comando fprintf exibe dois números de T em cada linha.
```

Executando o arquivo Capitulo4_Exemplo9 na linha do *prompt*:

```
>> Capitulo4_Exemplo9
   T =
   1.0000    2.0000    3.0000    4.0000    5.0000
   1.0000    1.4142    1.7321    2.0000    2.2361
Se o numero e 1, sua raiz quadrada e 1.000000
Se o numero e 2, sua raiz quadrada e 1.414214
```

A matriz `T` (2 × 5).

```
Se o numero e 3, sua raiz quadrada e 1.732051
Se o numero e 4, sua raiz quadrada e 2.000000
Se o numero e 5, sua raiz quadrada e 2.236068
```

O comando `fprintf` é repetido 5 vezes, usando os números da matriz `T` (coluna por coluna).

Usando o comando* `fprintf` *para salvar o resultado de saída num arquivo:

O comando `fprintf` pode ser utilizado para gravar os resultados de saída em um arquivo. Os dados salvos podem ser posteriormente usados no MATLAB ou em outras aplicações.

São necessárias três etapas para escrever um resultado gerado pelo `fprintf`:

a) Abrir um arquivo usando o comando `fopen`.
b) Escrever a saída no arquivo aberto usando o comando `fprintf`.
c) Fechar o arquivo usando o comando `fclose`.

Passo *a*:

Antes de os dados serem gravados em um arquivo, esse arquivo deve ser aberto. O comando `fopen` tem esse propósito. Com ele, é possível abrir um arquivo novo ou algum arquivo já existente. A forma geral do comando `fopen` é:

```
fid = fopen ('nome_do_arquivo', 'permissao')
```

onde `fid` é uma variável denominada identificador do arquivo. Um número é atribuído à variável `fid` ao executar `fopen`. Além disso, o nome do arquivo deve ser escrito, incluindo a extensão, entre aspas simples. O código de permissão é um campo (escrito como *string*) que norteia como o arquivo é aberto. Algumas permissões típicas são:

`'r'` Abrir arquivo para leitura (*default*).
`'w'` Abrir arquivo para escrita. Se o arquivo existir, seu conteúdo será excluído. Caso contrário, um novo arquivo é criado.
`'a'` Idêntico à permissão `'w'`, exceto que, se o arquivo existir, o dado será anexado ao conteúdo existente nesse arquivo (no final).

Se uma permissão não for especificada, o arquivo será aberto na opção `'r'`. Outras permissões são descritas no menu Help.

Passo *b*:

Uma vez que o arquivo encontra-se aberto, o comando `fprintf` pode ser usado para escrever nele. O comando deve ser utilizado como na criação de rotinas, exceto que a variável `fid` deve ser inserida dentro do comando. A forma geral do `fprintf` para esses propósitos é:

```
fprintf(fid, 'texto % -5.2f texto adicional', nome_da_variavel)
```

`fid` é adicionada ao comando `fprintf`.

Passo *c*:

Ao completar a escrita dos dados no arquivo, ele deve ser fechado por meio do comando `fclose`, que tem a seguinte forma geral:

```
fclose (fid)
```

Notas adicionais quanto ao uso do comando `fprintf` ***para salvar uma saída num arquivo:***

- O arquivo criado é salvo automaticamente no diretório selecionado em Current Directory.
- É possível usar o comando `fprintf` para escrever múltiplos arquivos diferentes. Para isso, os arquivos devem ser abertos, atribuídos a diferentes `fids` (p. ex.: `fid1`, `fid2`, `fid3`, etc.) e, então, escrever o `fid` de um arquivo específico no comando `fprintf` para gravar o conteúdo nesse arquivo.

A seguir é mostrado um exemplo de rotina que utiliza o comando `fprintf` para salvar resultados de saída em dois arquivos. O programa no arquivo gera duas tabelas de conversão. Uma tabela converte velocidades em milhas por hora (mi/h) em velocidades em quilômetros por hora (km/h). A outra tabela converte libras (lb) em Newtons (N). Cada uma das tabelas é salva em um arquivo-texto (extensão .txt) diferente.

```
% Rotina com fprintf que escreve os resultados de saida em arquivos.
% Duas tabelas de conversoes sao criadas e salvas em dois arquivos diferentes.
% Uma tabela converte mi/h em km/h e a outra converte lb em N.
clear all

Vmph = 10:10:100;                          Gerando um vetor de velocidades em mph.

Vkmh = Vmph.* 1.609;                       Convertendo mph em km/h.

TBL1 = [Vmph; Vkmh];                       Criando uma tabela (matriz) com duas linhas.

Flb = 200:200:2000;                        Gerando um vetor de forças em lb.

FN = Flb.* 4.448;                          Convertendo lb em N.

TBL2 = [Flb; FN];                          Criando uma tabela (matriz) com duas linhas.

fid1 = fopen ('Vmph_Vkmh.txt', 'w');       Abrindo um arquivo txt chamado Vmph_Vkmh.

fid2 = fopen ('Flb_FN.txt', 'w');          Abrindo um arquivo txt chamado Flb_FN.

fprintf (fid1, 'Tabela de Conversao de Velocidades\n\n');

                                           Escrevendo um título e um espaço no arquivo fid1.
```

```
fprintf (fid1, '      mi/h       km/h    \n');

fprintf (fid1, '      %8.2f      %8.2f\n', TBL1);

fprintf (fid2, 'Tabela de Conversao de Forcas\n\n');
fprintf (fid2, '      Libras     Newtons\n');
fprintf (fid2, '      %8.2f      %8.2f\n', TBL2);
fclose (fid1);
fclose (fid2);
```

Escrevendo o cabeçalho em duas colunas do arquivo fid1.

Escrevendo os dados da variável `TBL1` no arquivo fid1.

Escrevendo a tabela de conversão de força (dados na variável `TBL2`) para o arquivo fid2.

Fechando os arquivos fid1 e fid2.

A execução dessa rotina gera e salva dois novos arquivos-texto, nomeados Vmph_Vkmh.txt e Flb_FN.txt, no diretório selecionado em Current Directory. Esses arquivos podem ser abertos em qualquer editor de textos. As Figuras 4-5 e 4-6 mostram como os dois arquivos aparecem quando são abertos no Microsoft Word.

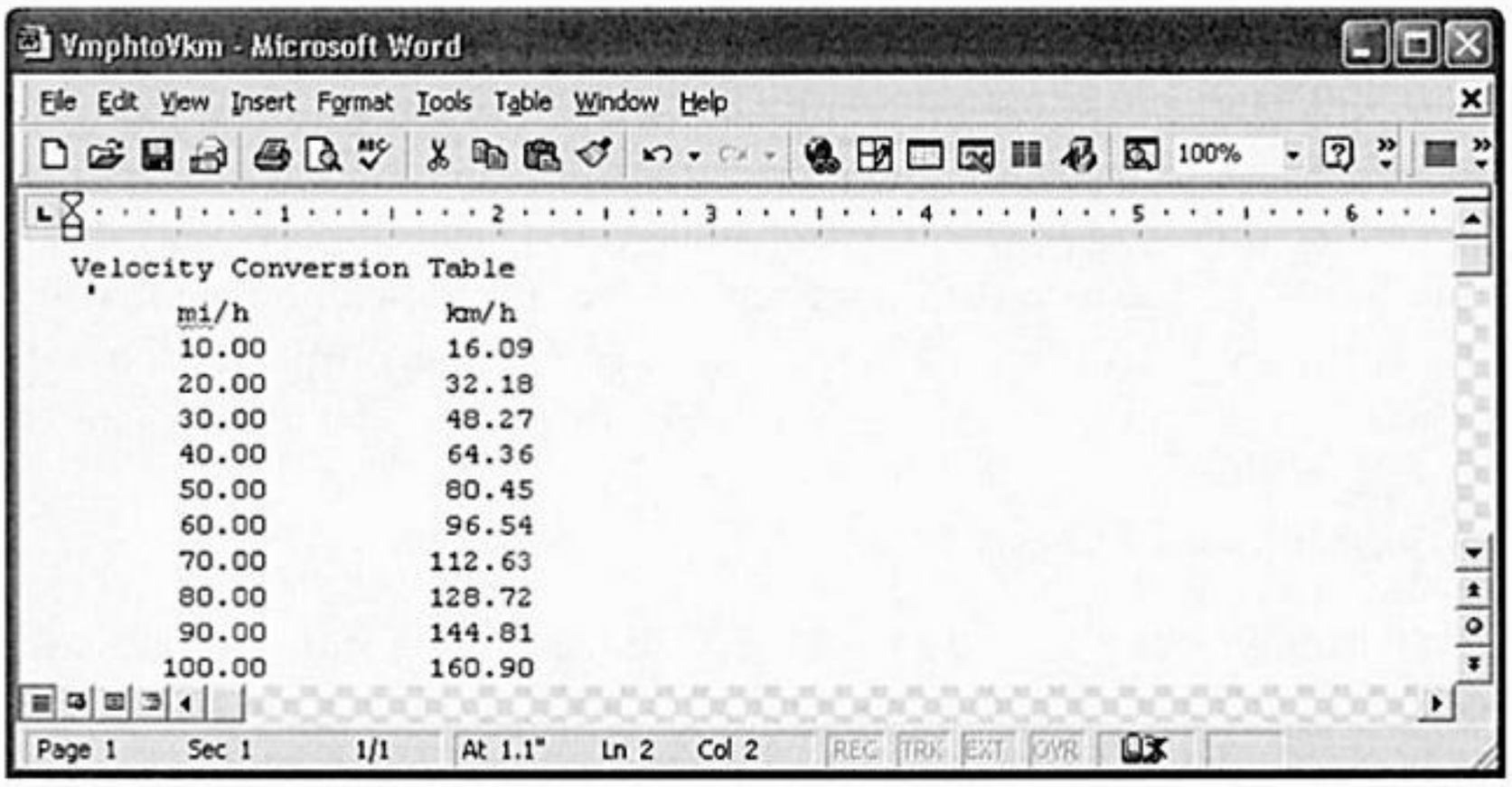


FIGURA 4-5 O arquivo Vmph_Vkmh.txt aberto no Word.

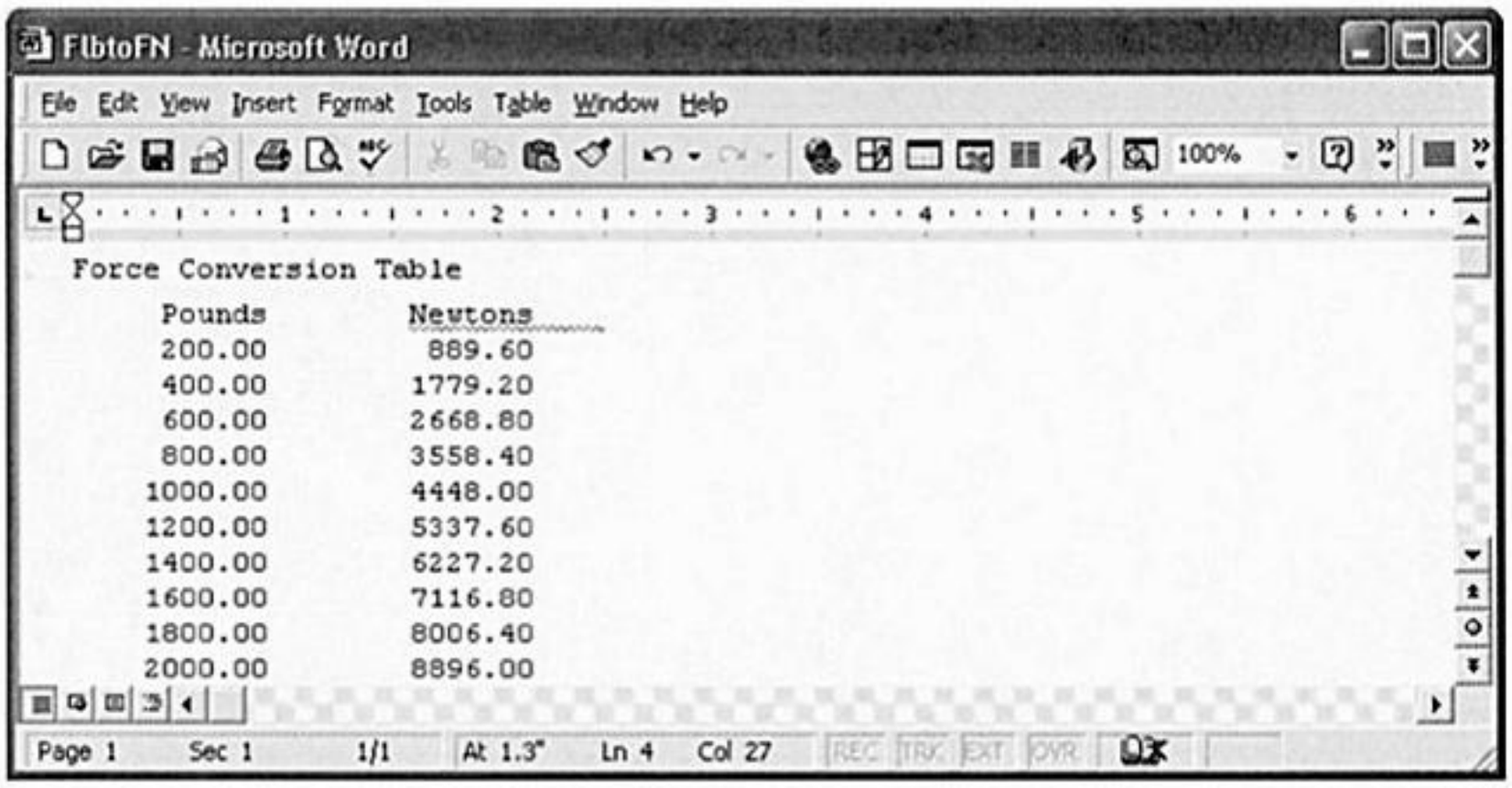


FIGURA 4-6 O arquivo Flb_FN.txt aberto no Word.

4.7 IMPORTANDO E EXPORTANDO DADOS

O MATLAB é usado freqüentemente para analisar dados gravados em experimentos ou gerados por outros programas de computador. Para realizar a análise dos dados, é necessário importá-los para o ambiente MATLAB. Outras vezes, dados gerados no MATLAB precisam ser utilizados por outras aplicações. Há muitos tipos de dados que o MATLAB pode tratar: numéricos, textos, áudio, gráficos e imagens. Esta seção descreve apenas como importar e exportar dados numéricos, o que provavelmente é a aplicação mais comum utilizada pelos novos usuários do MATLAB. Para outros tipos de transferência de dados, consulte o Help do programa e selecione a opção File I/O.

A importação de dados pode ser feita por comandos ou utilizando-se o Assistente de Importação (Import Wizard). Os comandos são bastante úteis quando o formato dos dados a serem importados é conhecido previamente. A diversidade de comandos do MATLAB destinados à importação de dados numéricos é impressionante de tão completa. Os comandos de importação de dados também podem fazer parte de uma rotina, de modo que os dados sejam importados apenas quando a rotina for executada. O Assistente de Importação mostra-se útil quando o formato dos dados ou o comando aplicável ao tipo de dados que se deseja importar é desconhecido. O Assistente de Importação se encarrega de determinar o formato dos dados e importá-los automaticamente.

4.7.1 Comandos para importação e exportação de dados

Esta seção descreve em detalhes como transferir dados para dentro e para fora de uma planilha do Excel. O Microsoft Excel é utilizada largamente para organizar/armazenar dados; o Excel é compatível com muito dispositivos de gravação e aplicações de computador. Além disso, as pessoas freqüentemente preferem importar e exportar vários formatos de dados para dentro e para fora do Excel. O MATLAB também possui comandos para transferir dados diretamente para e de outros formatos, como o csv e ASCII, e para planilhas padrão Lotus 123. Os detalhes desses e de muitos outros comandos podem ser encontrados no Help do MATLAB, na seção File I/O.

Importando e exportando dados para e do Excel:

A importação de dados do Excel é feita por meio do comando `xlsread`. Quando o comando é executado, os dados da planilha são organizados em um arranjo para uma variável. A forma mais simples do comando `xlsread` é:

```
Nome_variavel = xlsread ('Nome_arquivo')
```

- `'Nome_arquivo'` (digitado como uma *string*) é o nome do arquivo do Excel. O diretório do Excel deve concordar com o diretório especificado no campo Current Directory ou listado no caminho de procura.
- Se o arquivo do Excel tiver mais de uma folha, os dados serão importados apenas da primeira folha.

Quando um arquivo do Excel tiver mais de uma folha, o comando `xlsread` é capaz de especificar a folha de origem dos dados. A forma do comando passa a ser então:

```
Nome_variavel = xlsread ('Nome_arquivo', 'Nome_folha')
```

- O nome da folha deve ser digitado como uma *string*, isto é, entre aspas simples.

Outra opção é importar somente uma parte dos dados contidos em uma planilha. Para tanto, digite o argumento adicional no comando `xlsread`:

```
Nome_variavel = xlsread ('Nome_arquivo', 'Nome_folha',
                                          'Intervalo')
```

- O `'Intervalo'` (digitado como *string*) é uma região retangular da planilha de onde se deseja fazer aquisição dos dados, endereçada na notação do Excel. Por exemplo, `'C2:E5'` é uma região 4 por 3 de linhas 2, 3 e 4 e colunas *C*, *D* e *E*.

A exportação de dados do MATLAB para uma planilha do Excel é feita por meio do comando `xlswrite`. A forma mais simples desse comando é:

```
xlswrite ('Nome_arquivo', Nome_variavel)
```

- `'Nome_arquivo'` (digitado como *string*) é o nome do arquivo do Excel para o qual os dados devem ser exportados. O arquivo deve ser especificado no Current Directory. Se o arquivo não existir, um novo arquivo do Excel com o nome especificado deve ser criado.
- `Nome_variavel` é o nome da variável do MATLAB de onde os dados serão exportados.
- Os argumentos `'Nome_folha'` e `'Intervalo'` podem ser utilizados no argumento do comando `xlswrite` nas exportações de dados para uma folha específica ou para um intervalo específico de uma folha, respectivamente.

Como exemplo, os dados mostrados na planilha do Excel da Figura 4-7 serão importados para o MATLAB por meio do comando `xlsread`.

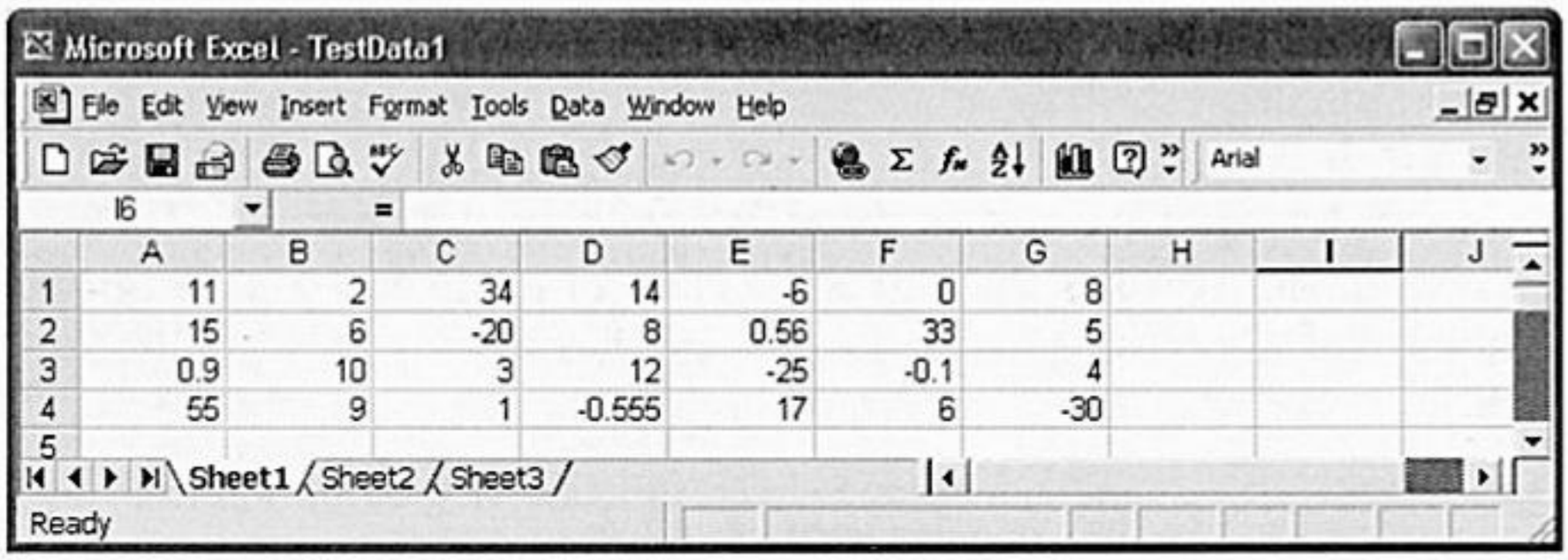


FIGURA 4-7 Planilha do Excel alimentada com dados.

A planilha foi salva em um arquivo denominado TestData1 em um disquete na unidade A. Especificando o Current Directory para o drive A, os dados são importados para o MATLAB e atribuídos à variável DADOS:

```
>> DADOS=xlsread('TestData1')
DADOS =
    11.0000    2.0000   34.0000   14.0000   -6.0000         0    8.0000
    15.0000    6.0000  -20.0000    8.0000    0.5600   33.0000    5.0000
     0.9000   10.0000    3.0000   12.0000  -25.0000   -0.1000    4.0000
    55.0000    9.0000    1.0000   -0.5550   17.0000    6.0000  -30.0000
```

4.7.2 Usando o assistente de importação

Utilizar o Assistente de Importação (Import Wizard) é provavelmente o modo mais simples de se importar dados para o MATLAB, já que o usuário não precisa especificar o formato dos dados. O Assistente de Importação é ativado selecionando-se **Import Data** no menu **File** da janela Command Window (também podemos inicializá-lo digitando o comando `uiimport` na linha do *prompt*). O Assistente de Importação inicia exibindo a caixa de seleção de arquivos que o Assistente consegue reconhecer. O usuário seleciona então o arquivo de onde os dados devem ser importados e clica em **Open**. O Assistente de Importação abre o arquivo e exibe uma pequena porção dos dados truncados na caixa denominada *preview* do assistente, de modo que o usuário possa verificar a autenticidade dos dados importados. O Assistente de Importação tenta processar os dados e, se a importação for bem-sucedida, são exibidas as variáveis criadas especificamente para receber os dados importados. Em seguida, o usuário clica em **next** e o assistente mostra o Column Separator (separador de colunas) utilizado. Se a variável receber os dados corretamente, o usuário poderá continuar a utilizar o Assistente clicando em **next**; caso contrário, poderá escolher um separador de colunas diferente.

Na próxima janela, o assistente mostra o nome e o tamanho da variável criada no MATLAB para receber os dados (quando todos os dados são numéricos, a variável do MATLAB recebe o mesmo nome do arquivo de onde os dados foram importados). Quando o assistente é finalizado (clicando-se em **finish**), os dados são importados para o MATLAB.

A seguir vamos ilustrar o processo de importação de dados numéricos no formato ASCII via Assistente de Importação. Vamos importar um arquivo salvo no formato txt. Os dados salvos com o nome de TestData são mostrados na Figura 4-8.

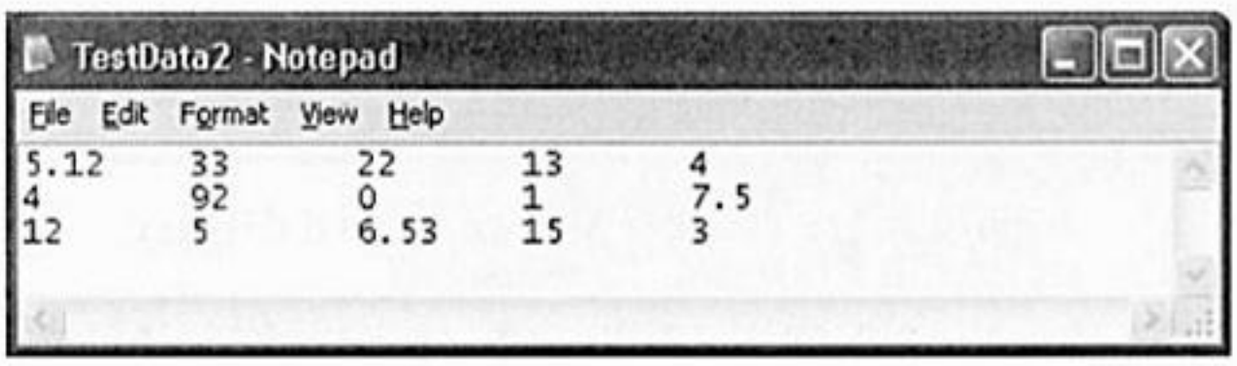


FIGURA 4-8 Dados numéricos no formato ASCII.

As telas exibidas durante o processo de utilização do Assistente de Importação para importar o arquivo TestData2 são mostradas nas Figuras 4-9 a 4-11. A terceira figura mostra que o nome da variável do MATLAB que irá receber os dados é TestData2 e o tamanho dela é 3 por 5. Na janela Command Window, os dados importados são mostrados digitando-se o nome da variável sem ponto-e-vírgula no final.

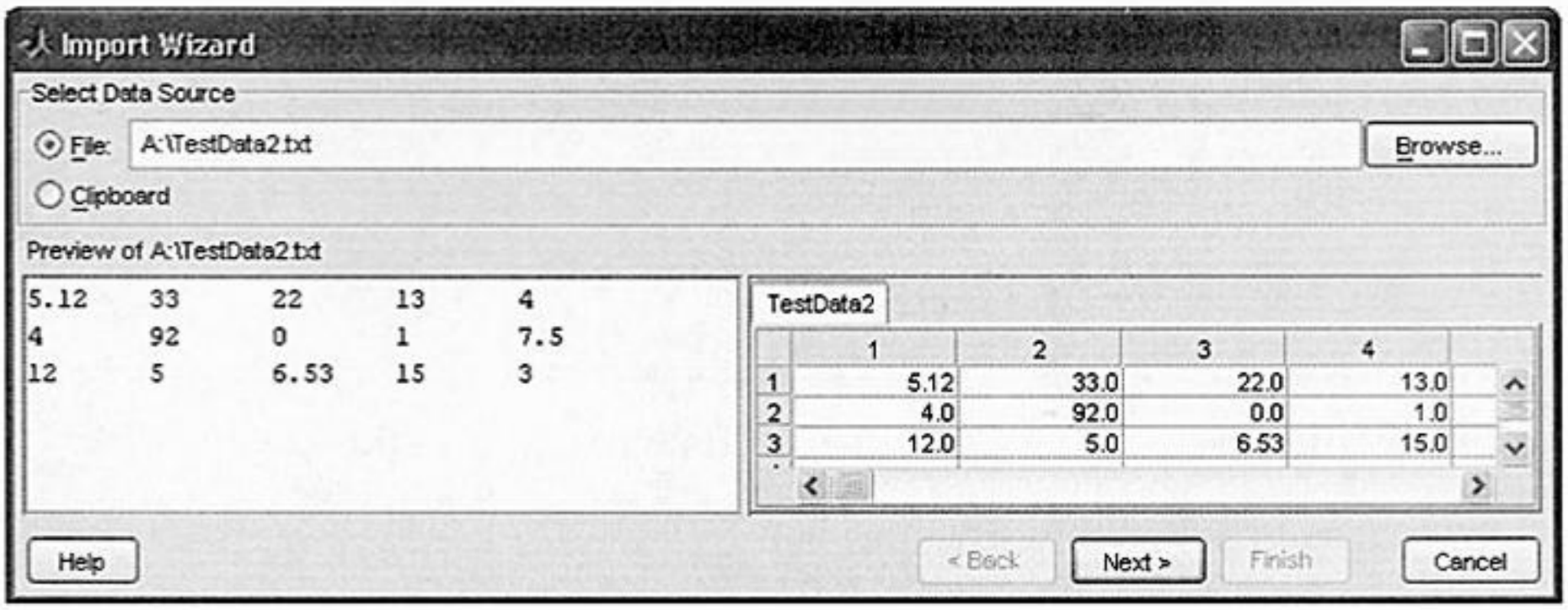


FIGURA 4-9 Assistente de importação, primeira janela.

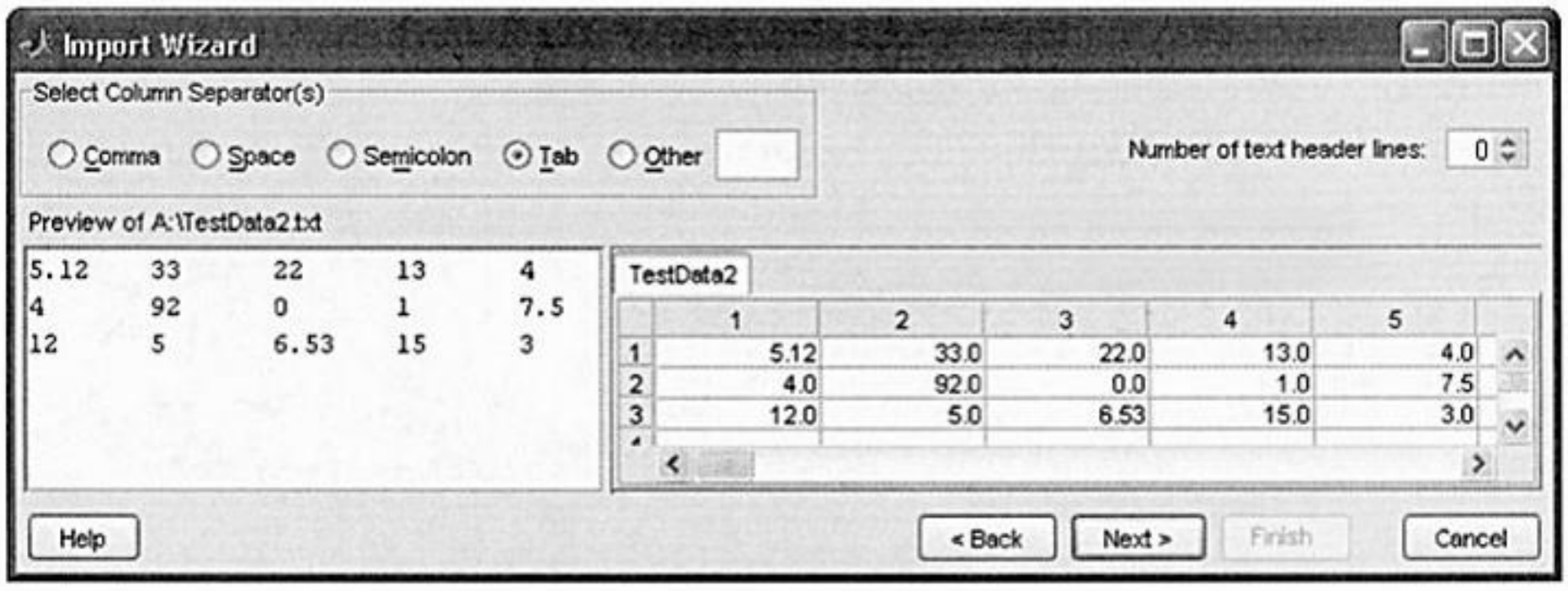


FIGURA 4-10 Assistente de importação, segunda janela.

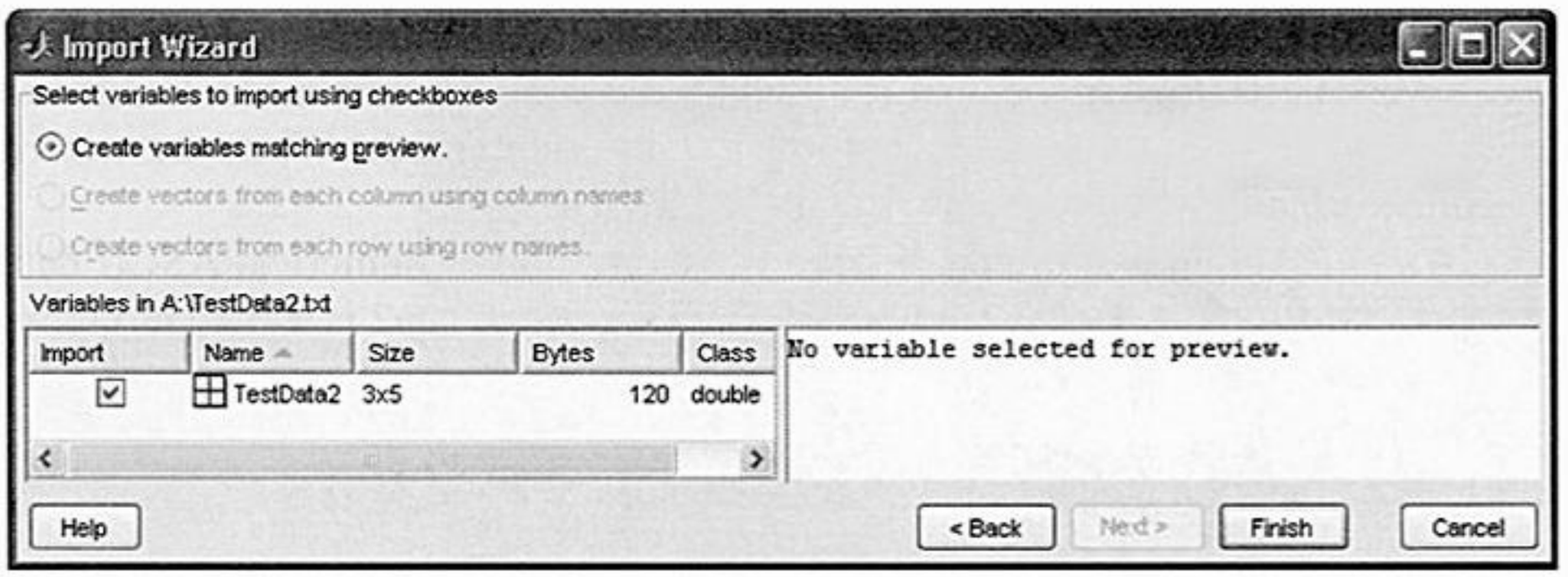


FIGURA 4-11 Assistente de importação, terceira janela.

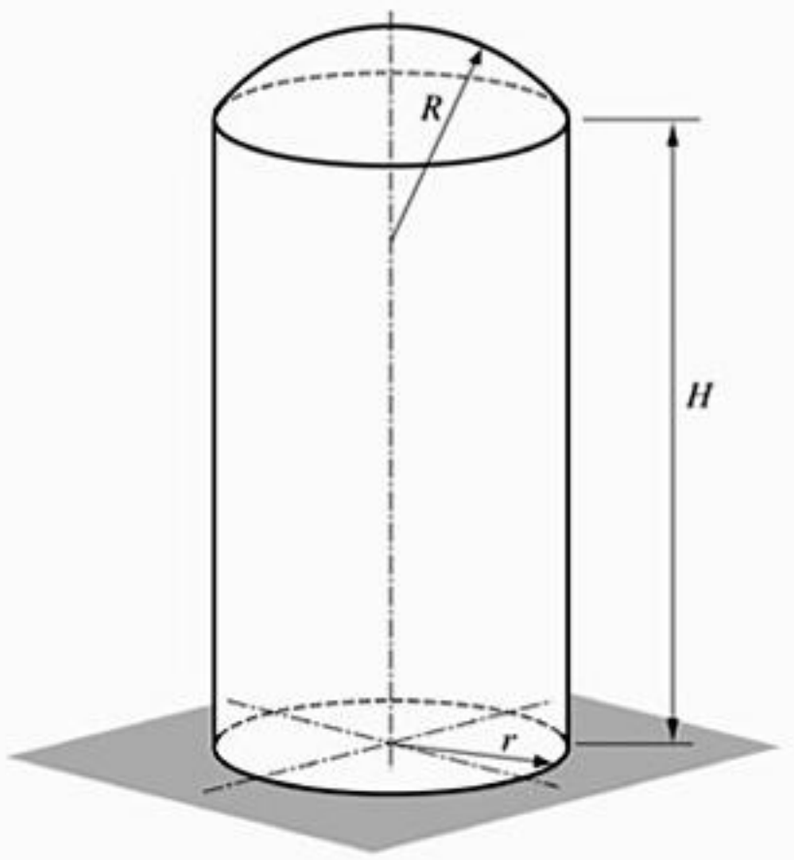
```
>> TestData2
TestData2 =
    5.1200   33.0000   22.0000   13.0000    4.0000
    4.0000   92.0000         0    1.0000    7.5000
   12.0000    5.0000    6.5300   15.0000    3.0000
```

4.8 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 4-1: Altura e área de superfície de um silo

Um silo cilíndrico de raio r possui uma cobertura na forma de uma calota esférica de raio R. A altura da porção cilíndrica é H. Escreva uma rotina que determine a altura H dados os valores de r, R e o volume V. Em seguida, o programa também deve calcular a área da superfície do silo.

Finalmente, use o programa para calcular a altura e a área da superfície de um silo de r = 30m, R = 45m e volume de 120.000m^3. Atribua valores a r, R e V na linha do *prompt* em Command Window.



Solução

O volume total do silo é o obtido por adição dos volumes da parte cilíndrica e da calota esférica. O volume do cilindro é dado por:

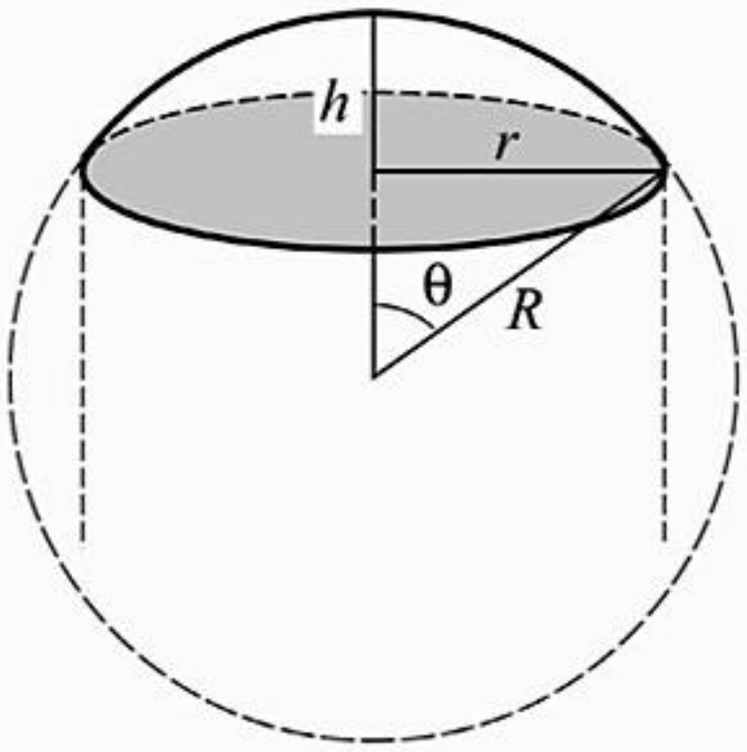
$$V_{cilin} = \pi r^2 H$$

e o volume da calota esférica é:

$$V_{cal} = \frac{1}{3}\pi h^2 (3R - h)$$

onde $h = R - R\cos\theta = R(1 - \cos\theta)$ e θ é calculado de $\sin\theta = \frac{r}{R}$. Utilizando as equações acima, a altura H do cilindro pode ser expressa como:

$$H = \frac{V - V_{cal}}{\pi r^2}$$



A área de superfície do silo é obtida pela adição das áreas de superfície da parte cilíndrica (A_{cilin}) e esférica (A_{cal}):

$$S = S_{cilin} + S_{cal} = 2\pi rH + 2\pi Rh$$

Uma rotina que resolve o problema acima é apresentada a seguir:

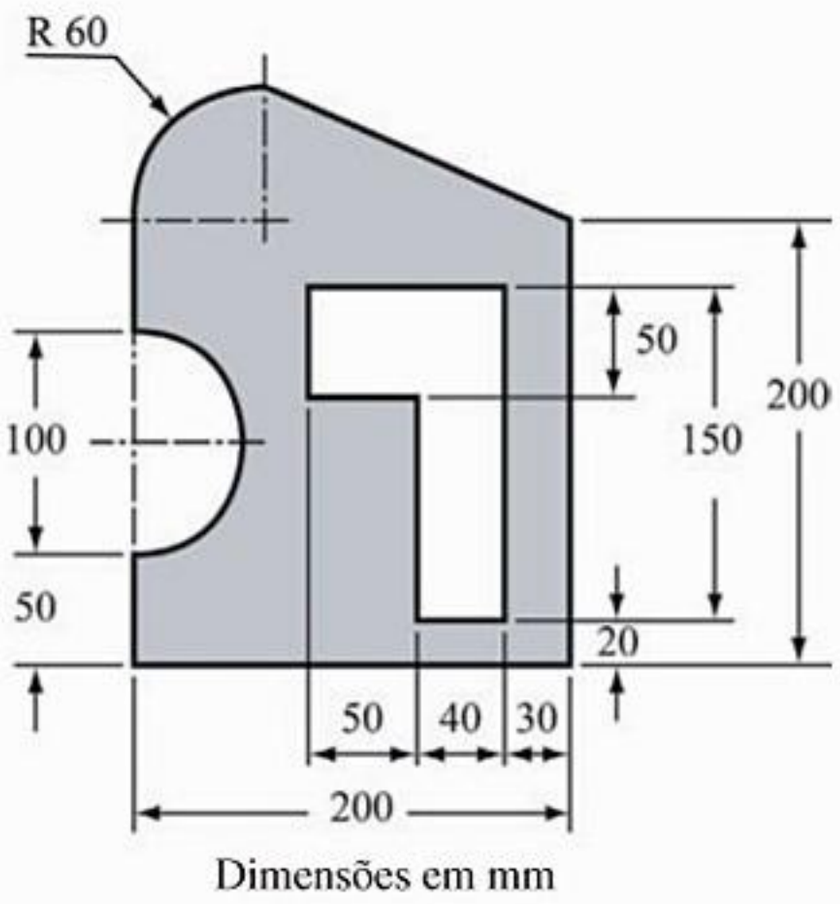
```
teta = asin(r/R);                                  Calculando θ.

h = R*(1–cos(teta));                               Calculando h.

  Vcal = pi*h^2*(3*R – h)/3;                       Calculando o volume da calota esférica.

                                                   Calculando H.
H = (V–Vcal)/(pi*r^2);

                                                   Calculando a área de superfície S.
S = 2*pi*(r*H + R*h);
fprintf ('Altura H e: %.2f m.', H)
fprintf ('\nA area de superficie do silo e: %.2f m2.', S)
```

Na janela Command Window, executando o arquivo Capitulo4_Exemplo10 temos:

```
>> r =30; R= 45; V = 200000;                       Atribuindo valores a r, R e V.
>> Capitulo4_Exemplo10
Altura H e: 36.43 m.                               Executando o arquivo Capitulo4_Exemplo10.
A area de superficie do silo e: 10107.44 m2.
```

Problema-exemplo 4-2: Centro geométrico (centróide) de um objeto composto

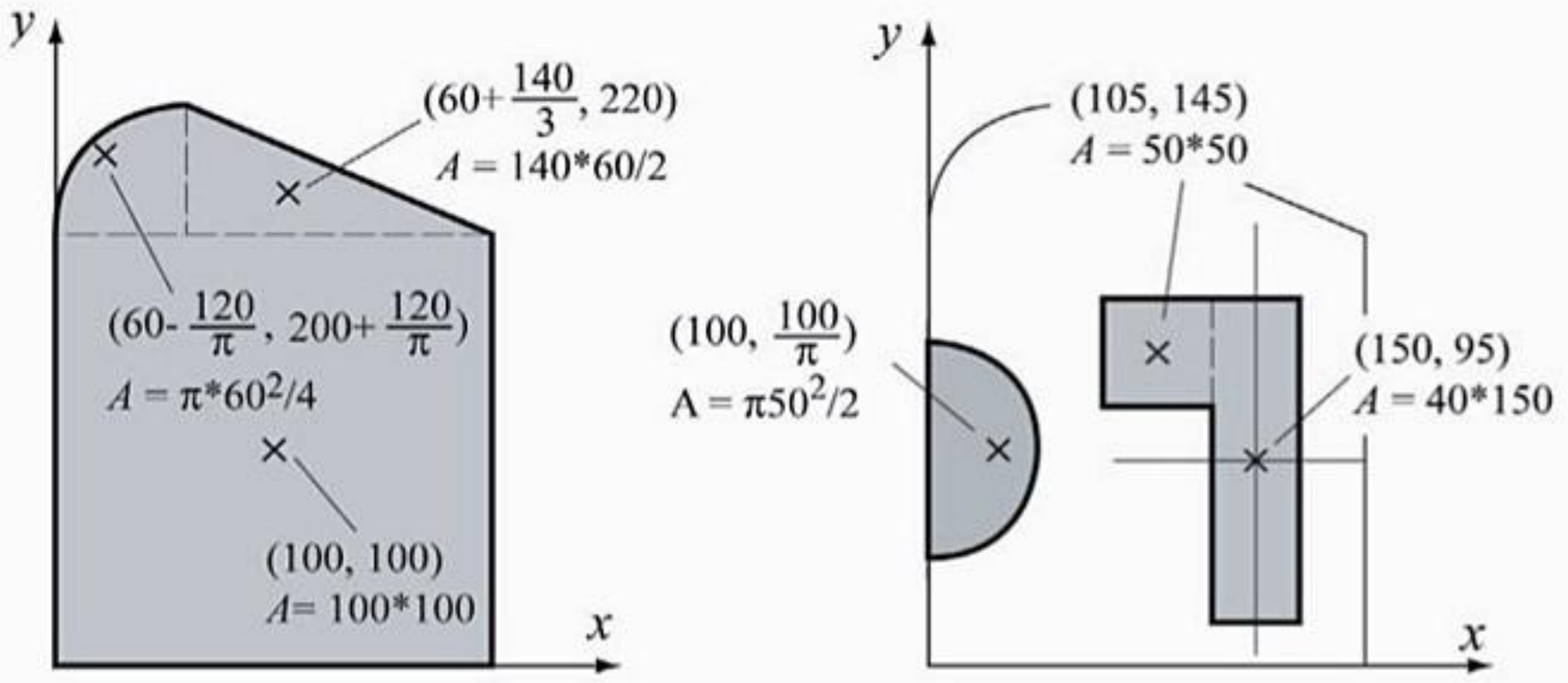
Escreva uma rotina que calcule as coordenadas do centro geométrico de uma figura (o objeto composto pode ser decomposto em seções cujos centróides são conhecidos). Divida o objeto em seções e descubra as coordenadas dos centros geométricos de cada seção. Ao executar a rotina, o programa deve solicitar três números, que devem ser armazenados em uma linha de uma matriz. O número de linhas da matriz deve concordar com o número de seções que o objeto possuir. Além disso, uma seção que representa um buraco no objeto deve possuir área negativa. Como saída, o programa deve exibir as coordenadas do centróide do objeto composto. Use o programa para calcular o centróide da área mostrada na figura.



Dimensões em mm

Solução

A área é dividida em seis seções, como mostra a figura a seguir. A área total é a soma das áreas das seções à esquerda menos a soma das três áreas à direita. A localização e as coor-



Unidades em mm, área em mm^2

denadas do centróide de cada seção estão assinaladas na figura, assim como a área de cada seção.

As coordenadas do centróide $\overline{X}$ e $\overline{Y}$ do objeto composto são dados por: $\overline{X} = \frac{\sum A\overline{x}}{\sum A}$ e $\overline{Y} = \frac{\sum A\overline{y}}{\sum A}$, onde $\overline{x}$, $\overline{y}$ e A são as coordenadas do centróide e a área de cada seção, respectivamente.

Um programa escrito em uma rotina para calcular as coordenadas do centróide do objeto composto é mostrado a seguir:

```
% Este programa calcula as coordenadas do centroide de um objeto composto.
Clear C xs ys As
C = input ('Entre com uma matriz onde cada linha possui tres elementos.\nCada linha deve receber as coordenadas do centroide (x e y) e a area de cada seção A. \n');
xs = C(:,1)';
ys = C(:,2)';
As = C(:,3)';
A = sum(As);
x = sum(As.*xs)/A;
y = sum (As.*ys)/A;
fprintf('As coordenadas do centroide sao: (%.f, %.f)', x, y)
```

Criando um vetor linha para a coordenada *x* de cada seção (primeira coluna de C).

Criando um vetor linha para a coordenada *y* de cada seção (segunda coluna de C).

Criando um vetor linha para área de cada seção (terceira coluna de C).

Calculando a área total.

Calculando as coordenadas do centróide do objeto composto.

A rotina foi salva com o nome Capitulo4_Exemplo11. A seguir, temos o resultado da execução da rotina na janela Command Window:

```
>> Capitulo4_Exemplo11
Entre com uma matriz onde cada linha possui tres elementos.
Cada linha deve receber as coordenadas do centroide (x e y) e a area de cada secao.
[100 100 200*200
60-120/pi 200+120/pi pi*60^2/4
60+140/3 220 140*60/2
100 100/pi -pi*50^2/2
150 95 -40*150
105 145 -50*50]
As coordenadas do centroide sao: (87.691, 140.823)
>>
```

Digitando as entradas na matriz C. Cada coluna possui três elementos: *x*, *y* e *A* de uma seção.

Problema-exemplo 4-3: Divisor de tensão

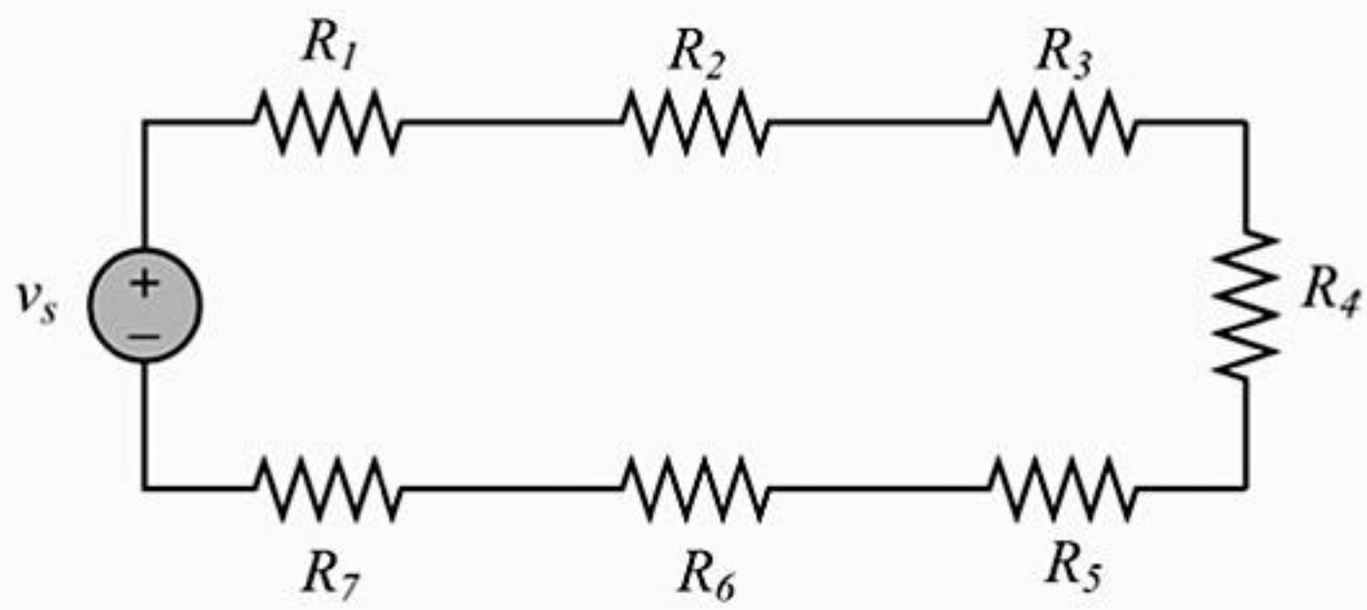
Quando resistores são conectados para formar um circuito série, a queda de tensão em cada um deles pode ser determinada pela regra do divisor de tensão:

$$v_n = \frac{R_n}{R_{eq}} v_s$$

onde v_n e R_n são a queda de tensão sobre o resistor n e a resistência desse resistor, respectivamente, $R_{eq} = \Sigma R_n$ é a resistência equivalente ou total e v_s é a tensão da fonte. A potência dissipada em cada resistor é dada por:

$$P_n = \frac{R_n}{R_{eq}^2} v_s^2$$

A figura abaixo mostra, por exemplo, um circuito com sete resistores conectados em série.



Escreva um programa em uma rotina que calcula a queda de tensão e a potência dissipada em cada resistor de um circuito série com 7 resistores. Executando a rotina, o usuário deve ser solicitado a digitar em um vetor o valor da fonte de tensão e a resistência de cada resistor. O programa exibirá uma tabela com as resistências listadas na primeira coluna, a queda de tensão na segunda coluna e a potência dissipada na terceira coluna. Após a tabela, o programa deve exibir a corrente no circuito e a potência total dissipada. Teste a rotina com os seguintes valores de v_s e Rs:

$v_s = 24\text{V}$, $R_1 = 20\Omega$, $R_2 = 14\Omega$, $R_3 = 12\Omega$, $R_4 = 18\Omega$, $R_5 = 8\Omega$, $R_6 = 15\Omega$ e $R_7 = 10\Omega$

Solução

Uma rotina escrita para solucionar esse programa é mostrada a seguir.

```
% Este programa calcula a queda de tensao em cada resistor num circuito serie.
vs = input('Entre com o valor da fonte de tensao');
Rn = input('Entre com os valores dos resistores como elementos de um vetor linha.\n');
Req = sum(Rn);                                  Calcula a resistência equivalente.

vn = Rn*vs/Req;                                 Aplica a regra do divisor de tensão.

Pn = Rn*vs^2/Req^2;                             Determina a potência em cada resistor.

i = vs/Req;                                     Calcula a corrente no circuito.

Ptotal = vs*i;                                  Calcula a potência total no circuito.

Tabela = [Rn',vn',Pn'];
                                                Cria uma variável denominada tabela
disp('')                                        com os vetores Rn, vn e Pn como colunas.

disp('    Resistencia      Tensao      Potencia')   Exibe as unidades de medidas das colunas.

disp('      (Ohms)         (Volts)     (Watts)')    Exibe o rótulo das colunas.

disp('')                                        Imprime uma linha em branco.

disp(Tabela)                                    Exibe os dados da tabela.

disp('')

fprintf('A corrente no circuito e %.3f Amps', i)

fprintf('\nA potencia total dissipada no circuito e %.3f Watts', Ptotal)
```

Executando a rotina (Capitulo4_Exemplo12) o resultado será:

```
>> Capitulo4_Exemplo12

Entre com o valor da fonte de tensao 24
Entre com os valores dos resistores como elementos de um vetor linha.
[20  14  12  18  8  15  10]

   Resistencia        Tensao        Potencia
   (Ohms)             (Volts)       (Watts)
     20.0000          4.9485         1.2244
     14.0000          3.4639         0.8571
     12.0000          2.9691         0.7346
     18.0000          4.4536         1.1019
      8.0000          1.9794         0.4897
     15.0000          3.7113         0.9183
     10.0000          2.4742         0.6122

A corrente no circuito e 0.247 Amps.
A potencia total dissipada no circuito e 5.938 Watts.
```

Nome da rotina.

Inicializa a variável fonte de tensão.

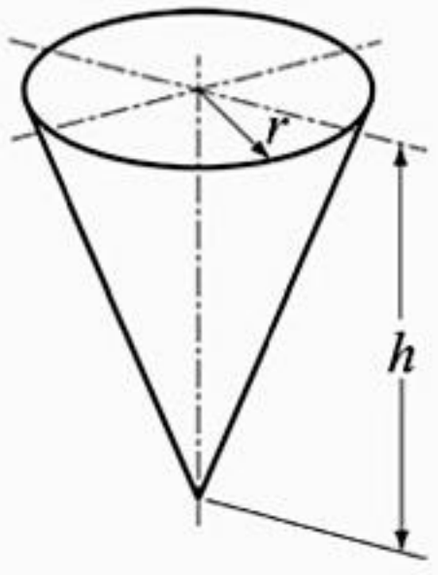
Inicializa o vetor `Rn` com os valores dos resistores.

4.9 PROBLEMAS

Nota: *assim como os problemas a seguir, os problemas ao final do Capítulo 3 podem ser resolvidos por meio de rotinas.*

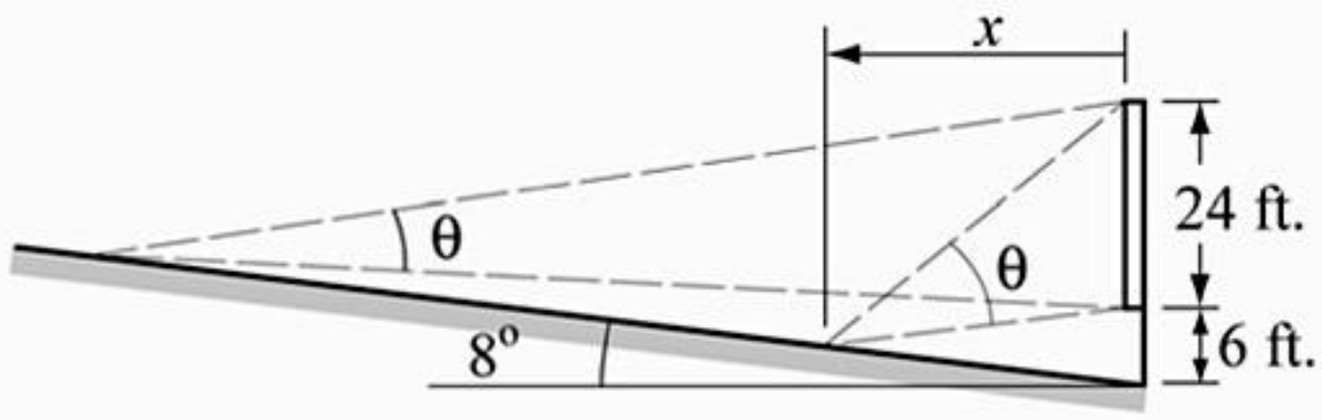
Resolva os seguintes problemas escrevendo em primeiro lugar a rotina, executando-a depois na linha do *prompt* na janela Command Window.

1. Um cone de papel foi construído de modo a possuir um volume de 250cm^3. Determine o raio r da base e a área da superfície (S) do cone, considerando as seguintes alturas h do cone: 5, 6, 7, 8 e 9 cm.
O volume V e a área da superfície do cone são dados por:



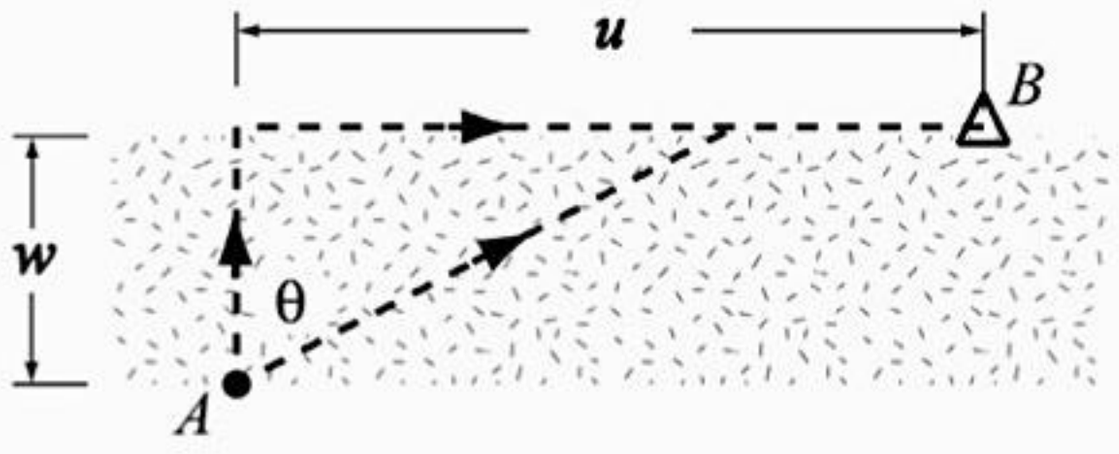
$$V = \frac{1}{3}\pi r^2 h \qquad S = \pi r\sqrt{r^2 + h^2}$$

2. Numa sala de cinema, o ângulo θ de observação da platéia depende da distância x de cada espectador em relação à tela. A figura mostra as dimensões de uma sala de cinema. Determine o ângulo θ em graus para espectadores distantes 9m; 13.5m; 18m; 22.5m e 27m da tela.



3. A população de um determinado país é 50 milhões e a expectativa é de que ela duplique em 20 anos. Determine a população em 5, 10 e 15 anos, partindo da população inicial, declarando um vetor t com três elementos e realizando cálculos escalares, isto é, elemento por elemento. O crescimento populacional segue a equação $P = P_0 2^{t/d}$, onde P é a população no tempo t, P_0 é a população em $t = 0$ e d é o tempo necessário para que a população duplique.

4. Um banhista precisa cruzar uma certa faixa de areia de praia, partindo do ponto A em direção ao acampamento no ponto B. Ele pode fazer isso atravessando perpendicularmente a faixa de areia e andando o restante ao longo de uma trilha ou cruzando-a em um ângulo θ, até um certo ponto, e andando o restante ao longo da trilha. Considerando-se que o banhista caminha a 3.5km/h na areia e a 5km/h na trilha, determine o tempo para que ele chegue ao acampamento para θ = 0, 10, 20, 30, 40, 50 e 60 graus.



As distâncias w e u são 4,5km e 14km, respectivamente. Escreva um programa em uma rotina para resolver o problema. Declare todas as variáveis dentro da rotina. Mostre os resultados em duas colunas de uma tabela, onde a primeira coluna representa o ângulo θ e a segunda coluna corresponde ao tempo t.

5. Escreva uma rotina que determina o saldo de uma conta de poupança no final de cada ano. Considere que a conta não será movimentada durante 10 anos, que o investimento inicial é $1,000.00 e que a taxa de juros anual é 6.5%. Apresente a informação numa tabela.

Supondo que o investimento inicial seja A e a taxa de juros r, o saldo da conta B após n anos é dado:

$$B = A\left(1+\frac{r}{100}\right)^n$$

6. A velocidade v e a distância d, em função do tempo, de um carro que acelera uniformemente a partir do repouso são dadas, respectivamente, por:

$$v(t) = at \quad \text{e} \quad d(t) = \frac{1}{2}at^2$$

sendo a a aceleração do carro. Se a aceleração do carro é $a = 1.55\text{m/s}^2$, determine d e v do carro para cada segundo, após os primeiros 10s. Apresente os resultados numa tabela de três colunas, onde a primeira é o tempo em segundos (s), a segunda é a distância em metros (m) e a terceira é a velocidade em quilômetro por hora (km/h).

7. Quando resistores são conectados em paralelo num circuito elétrico, a corrente que flui através de cada resistor é:

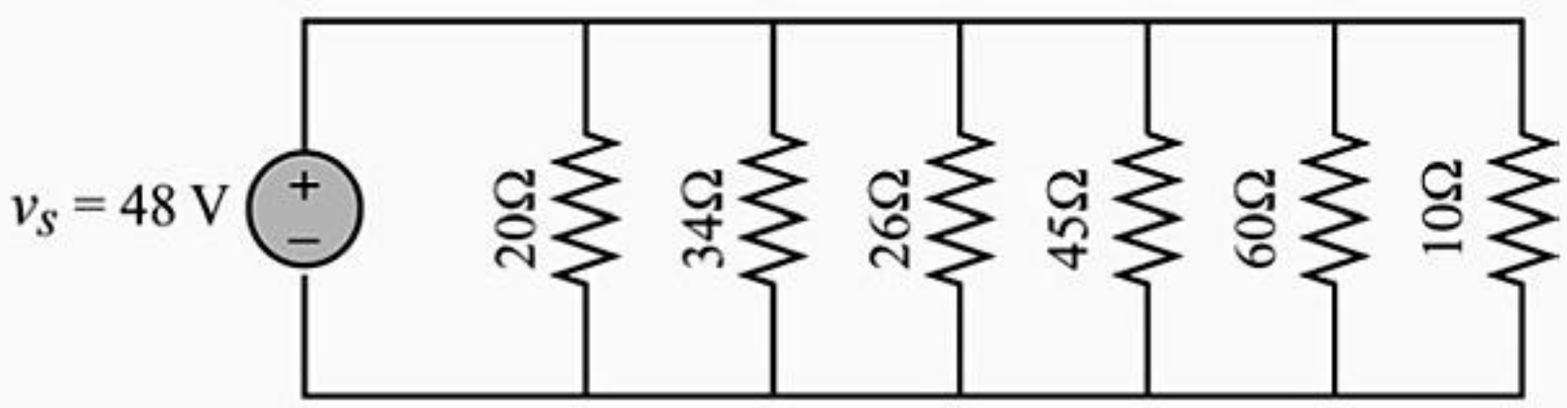
$$i_n = \frac{v_s}{R}$$

onde i_n e R_n são a corrente e a resistência do resistor n, respectivamente, e v_s é a tensão da fonte. A resistência equivalente (R_{eq}) pode ser determinada a partir da equação:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$$

A fonte de corrente é dada por $i_s = v_s / R_{eq}$ e a potência dissipada em cada resistor é dada por $P_n = v_s i_n$.

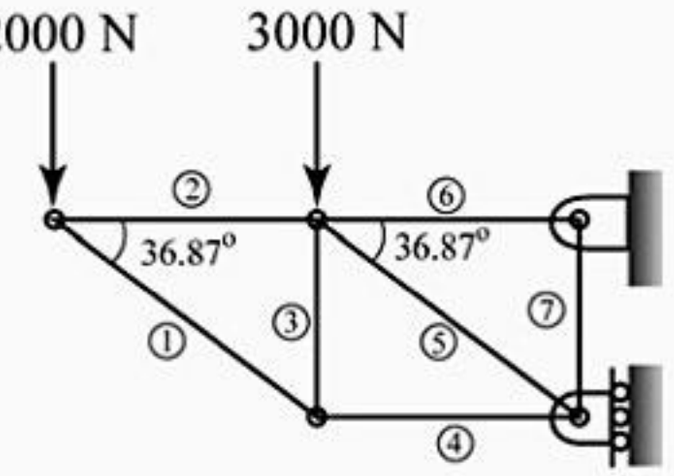
Escreva um programa numa rotina que calcule a corrente e a potência em cada resistor do circuito paralelo. Ao executar a rotina, o usuário será solicitado primeiro a entrar em um vetor com o valor da fonte de tensão e, em seguida, com os valores de resistência dos resistores. O programa deve apresentar uma tabela com as resistências listadas na primeira coluna, a corrente na segunda colu-



na e a potência dissipada na terceira coluna. Após a tabela, o programa deve mostrar a corrente e a potência total da fonte.

Teste a rotina no circuito apresentado a seguir:

8. O gráfico da função $f(x) = ax^3 + bx^2 + cx + d$ passa pelos pontos (–2; –3,4), (–0.5; 5.525), (1; 16.7) e (2.5; 70.625). Determine as constantes a, b, c e d. (Sugestão: escreva um sistema de quatro equações e quatro incógnitas e use o MATLAB para resolvê-lo.)



9. Treliça é o nome de uma estrutura feita de várias vigas unidas pelas extremidades (em junções ou nós, considerados pontuais nos cálculos teóricos). No caso da treliça mostrada na figura, as forças que agem nas sete vigas são determinadas resolvendo-se o sistema de equações a seguir:

$$F_1\sin(36.87^\circ) = -2000$$
$$F_1\cos(36.87^\circ) + F_2 = 0$$
$$F_3 + F_1\sin(36.87^\circ) = 0$$
$$F_4 - F_1\cos(36.87^\circ) = 0$$
$$-F_3 - F_5\sin(36.87^\circ) = 3000$$
$$F_6 + F_5\cos(36.87^\circ) - F_2 = 0$$
$$F_5\sin(36.87^\circ) + F_7 = 0$$

Escreva as equações na forma matricial e use o MATLAB para determinar as forças nas vigas. Uma força positiva indica força de tensão e uma força negativa indica força de compressão. Apresente os resultados numa tabela.