```
case 'J'
   Esai = EJ;
case 'ft-lb'
   Esai = EJ*0.738;
case 'cal'
   Esai = EJ*0.239;
case 'eV'
   Esai = EJ*6.24e18;
otherwise
   erro = 1;
end
if erro
   disp ('ERRO!!! A unidade atual ou a requerida foi digitada incorretamente.')
else
   fprintf('E = %g %s',Esai,Esai_unid)
end
```

Cada uma das quatro sentenças `case` possui um valor (*string*) correspondendo a uma unidade nova e um comando que converte a energia `EJ` na unidade nova (atribui o valor à `Esai`.).

Atribui 1 à variável `erro` caso não seja encontrado o valor. A unidade nova foi digitada incorretamente.

Estrutura `If-else-end`.

Se a variável `erro` for verdadeira (não-nula), será exibida uma mensagem de erro.

Se `erro` for falsa, a saída exibirá a energia convertida.

A título de exemplo, a rotina (salva como Capitulo7_Exemplo4) é testada na janela Command Window para resolver a letra *b* do problema.

```
>> Capitulo7_Exemplo4
Entre com o valor da energia(trabalho) a ser convertido: 2800
Entre com a unidade atual (J, ft-lb,cal ou eV): cal
Selecione a nova unidade de energia (J,ft-lb,cal ou eV): J
E = 11715.5 J
```

7.4 LAÇOS (*LOOPS*)

O laço é outro método de alterar o fluxo de um programa. Em um laço, a execução de um ou um grupo de comandos é repetida diversas vezes, sucessivamente. Cada seqüência de execução (repetição) do laço é denominada passo. A cada passo, ao menos uma variável é modificada dentro do laço. O MATLAB traz dois tipos de laço: `for-end` e `while-end`. No laço `for-end` (Seção 7.4.1), o número de repetições é conhecido desde o início do laço. No laço `while-end` (Seção 7.4.2), o número de repetições não é conhecido de antemão e o laço é repetido até que uma condição específica de parada seja satisfeita. Ambos os tipos de laço podem ser finalizados a qualquer momento, por meio do comando `break` (veja a Seção 7.6).

7.4.1 Laços `for-end`

Neste tipo de laço, a execução de um ou um grupo de comandos é repetido um número de vezes predeterminado. A forma do laço é mostrada na Figura 7-5.

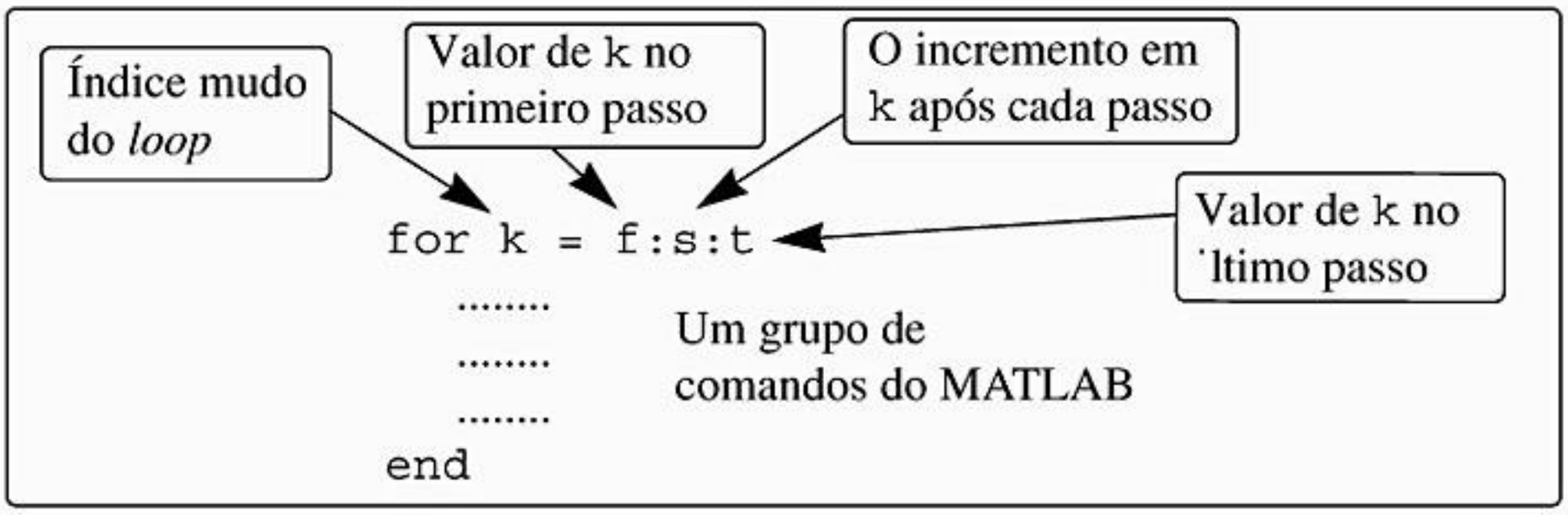


FIGURA 7-5 A estrutura do *loop* `for-end`.

- O nome da variável índice é arbitrário (tipicamente, são usadas as letras `i`, `j`, `k`, `m` e `n`. As letras `i` e `j` não devem ser utilizadas se o MATLAB estiver sendo programado para lidar com números complexos).
- Inicialmente, `k = f` e o computador executa os comandos entre `for` e `end`. Em seguida, o programa retorna ao comando `for`, incrementa o valor de `k = f + s`, compara com o valor de `t` e, caso sejam diferentes, segue para um novo passo (repetição) dos comandos entre `for` e `end`. O processo é repetido até que a condição `k = t` seja satisfeita. Nesse caso, o programa não retorna ao `for`, mas salta o grupo de comandos do laço e continua abaixo de `end`. Por exemplo, se `k = 1:2:9`, serão executados cinco laços, e os valores de `k` em cada laço são 1, 3, 5, 7 e 9.
- O incremento `s` pode ser um número negativo (i.e., `k = 25:-5:10` produz quatro laços com `k = 25, 20, 15, 10`).
- Se o incremento `s` for omitido, o MATLAB assumirá o valor-padrão ou *default* 1 (por exemplo, `k = 3:7` gera cinco laços `k = 3, 4, 5, 6, 7`).
- Se `f = t`, o laço é executado uma vez.
- Se `f > t` e `s` > 0 (ou `f < t` e `s` < 0), o laço não é executado.
- Se os valores de `k`, `s` e `t` forem tais que `k` não possa ser igual a `t`, então, se `s` > 0, o último passo acontecerá para o último valor de `k` menor que a condição de parada `t` (por exemplo, `k = 8:10:50` produz cinco repetições do laço, `k = 8, 18, 28, 38, 48`). Se `s` < 0, a última repetição acontecerá para o último valor de `k` maior que a condição de parada `t`.
- Em um comando `for`, é possível atribuir valores específicos a `k`, como num vetor. Por exemplo: `k = [7 9 –1 3 3 5]`.
- O valor de `k` não deve ser redefinido dentro do laço.
- Cada comando `for` dentro de um programa **deve** terminar em um comando `end`.

- O valor da variável índice do laço (`k`) não é mostrado automaticamente. É possível exibir o valor de `k` a cada passo (às vezes, útil no trabalho de *debugging* – depuração – do programa), digitando `k` como um dos comandos do laço.
- Terminado o laço, a variável índice (`k`) sai com o último valor atribuído a ela.

Um exemplo simples de um laço `for-end` (escrito como uma rotina) é:

```
for k=1:3:10
   x=k^2
end
```

Executando a rotina, o laço é repetido quatro vezes. O valor de `k` nas quatro repetições é `k` = 1, 4, 7 e 10, e os respectivos valores atribuídos à variável `x` são: `x` = 1, 16, 49 e 100. Como não foi digitado um ponto-e-vírgula após a segunda linha, os valores de `x` são exibidos, um a um, na janela Command Window. Executando a rotina, os resultados de `x` na janela Command Window são:

```
>>x =
   1
x =
   16
x =
   49
x =
   100
```

Problema-exemplo 7-5: Soma de séries

a) Construa um laço `for-end` em uma rotina para determinar a soma dos n primeiros termos da série: $\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$. Execute a rotina para $n = 4$ e $n = 20$.

b) A função sin(x) pode ser escrita em termos da série de Taylor como:

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}$$

Escreva uma função que calcule sin(x) usando a série de Taylor. Sugestão para o nome da função e argumentos: `y = Tsin(x,n)`. Os argumentos de entrada são o ângulo `x`, em graus, e o número de termos da série `n`. Teste a função para calcular sin(150°), usando 3 e 7 termos.

Solução

a) Uma rotina elaborada para calcular a soma dos n primeiros termos da série é mostrada a seguir:

```
n = input('Entre com o numero de termos da serie:');
S = 0;
for k = 1:n
S = S+(-1)^k*k/2^k;
end
fprintf('A soma da serie e: %f ',S)
```

Estabelecendo o resultado inicial da soma em zero.

Laço `for-end`

A cada repetição é determinado um elemento da série e o resultado é adicionado à soma do laço anterior.

O somatório é realizado dentro do laço. A cada passo um termo da série é calculado, sendo automaticamente adicionado à soma dos termos precedentes na série. Resta-nos executar o arquivo (salvo como Capitulo7_Exemplo5) duas vezes na janela Command Window:

```
>> Capitulo7_Exemplo5
Entre com o numero de termos da serie:4
A soma da serie e: –0.125000
>> Capitulo7_Exemplo5
Entre com o numero de termos da serie:20
A soma da serie e: –0.222216
```

b) Uma declaração de função que encontra sen(x) adicionando n termos da fórmula de Taylor é mostrada a seguir:

```
function y =Tsin(x,n)
% Tsin(x) determina o sin(x) com base na formula de Taylor.
% Argumentos de entrada:
% x (angulo em graus) e n (numero de termos da serie).
xr = x*pi/180;
y = 0;
for k = 0:n-1
   y = y+(-1)^k*xr^(2*k+1)/factorial(2*k+1);
end
```

Convertendo o ângulo de graus para radianos.

Laço `for-end`

O primeiro elemento da série corresponde a $k = 0$ e o último passo acontece para $k = n - 1$, de modo a adicionar n termos da série. Utilizando a função duas vezes na janela Command Window para calcular o sin(150°), com 3 e 7 termos na série, respectivamente, temos:

```
>> Tsin(150,3)                    Calculando o sin(150°) com 3 termos na série de Taylor.
ans =
0.6523
>> Tsin(150,7)                    Calculando o sin(150°) com 3 termos na série de Taylor.
ans =
0.5000                                                  O valor exato é 0.5.
```

Uma observação sobre o laço `for-end` ***e as operações elemento por elemento:***

Em algumas situações, resultados semelhantes podem ser produzidos usando-se o laço `for-end` ou operações envolvendo elemento a elemento de um arranjo. O Problema-exemplo 7-5 ilustra como funciona o laço `for-end`. Esse mesmo problema poderia ser resolvido usando-se operações elemento a elemento (veja os Problemas 7 e 8 na Seção 3.9). As operações elemento a elemento com arranjos são um dos diferenciais do MATLAB, se comparado a outras ferramentas de programação (por exemplo, a linguagem C). Em geral, operações elemento a elemento são mais rápidas que os laços e são recomendadas quando um problema é passível de solução pelos dois métodos.

Problema-exemplo 7-6: Modificando elementos de um vetor

Um vetor é dado por: V = [5, 17, –3, 8, 0, –1, 12, 15, 20, –6, 6, 4, –2, 16]. Escreva uma rotina que duplique os elementos positivos do vetor, divisíveis por 3 e/ou 5, e eleve à terceira potência os elementos maiores que –5.

Solução

O problema será resolvido por meio de um laço `for-end`, chamando-se internamente uma sentença condicional `if-elseif-end`. O número de repetições será igual ao número de elementos do vetor. Um elemento do vetor será verificado pela sentença condicional cada vez que o laço for executado (cada passo). O elemento verificado será modificado se obedecer a uma das condições estabelecidas no enunciado do problema. A rotina a seguir realiza as operações desejadas:

```
V = [5, 17, -3, 8, 0, -7, 12, 15, 20, -6, 6, 4, -2, 16];
n = length(V);                  Estabelecendo n igual ao número de elementos de V.
for k=1:n
   if V(k)>0&(rem(V(k),3) = = 0|rem(V(k),5)= =0)
      V(k)=2*V(k);
   elseif V(k)<0&V(k)>-5
      V(k)=V(k)^3;
   end
end
V
```

Laço `for-end`.

Sentença condicional `if-elseif-end`.

Executando a rotina na janela Command Window:

```
>> Capitulo7_Exemplo6
V =
  10   17  –27    8    0   –7   24   30   40   –6   12    4   –8   16
```

7.4.2 Laços `while-end`

O comando `while-end` é útil quando se deseja realizar um laço mas o número de repetições não é conhecido de antemão. Sendo assim, nesse laço, o formato do comando não contém um campo especificando o número de repetições (antes de iniciá-lo). Em vez disso, o processo continua sendo executado até que uma certa condição seja satisfeita. A estrutura do comando `while-end` é mostrada na Figura 7-6.

```
while expressão condicional
    .....
    .....          um grupo de
    .....          comandos do MATLAB
end
```

FIGURA 7-6 A estrutura do laço `while-end`.

A primeira linha traz a sentença `while`, incluindo a sentença condicional. Ao encontrar essa linha, o programa verifica a expressão condicional. Sendo falsa (0), o MATLAB salta para a sentença `end` e continua na primeira linha abaixo. Sendo verdadeira (1), o MATLAB executa o grupo de comandos compreendidos entre `while` e `end`. Depois disso, o MATLAB salta para a sentença `while` e verifica a expressão condicional. O laço é realizado enquanto a expressão condicional for verdadeira, abandonando o laço apenas quando o resultado do teste for falso.

Para um laço `while-end` ***ser executado corretamente:***

- A expressão condicional no comando `while` deve incluir ao menos uma variável.
- As variáveis na expressão condicional devem ter sido inicializadas com os valores corretos quando o MATLAB encontrar, na primeira vez, o comando `while`.
- Ao menos uma variável na expressão condicional deve ser modificada dentro do laço, ou seja, entre os comandos `while` e `end`. De outro modo, o programa entrará em um laço infinito.

Um exemplo simples de laço `while-end` é mostrado no programa a seguir. Nesse programa, a variável `x`, inicializada com o valor 1, é dobrada a cada passo, enquanto o valor for menor ou igual a 15.

```
x =1                                          Declarando x =1.
while x<=15
                                   O próximo comando é executado somente se x <=15.
   x=2*x
end
```

O resultado na janela Command Window é:

```
x =                                           Valor inicial de x.
   1
x =
   2
x =                                  O valor de x é dobrado a cada repetição.
   4
x =
   8
x =                          Quando x = 16, a expressão condicional no comando
   16                        while torna-se falso e o laço termina.
```

Observação:

Ao escrever um programa que utilize um laço `while-end`, o programador deve certificar-se de que a variável ou variáveis presente(s) na expressão condicional, e que deve ser modificada dentro do laço, receberá um valor que tornará falsa a expressão condicional do comando `while`. De outro modo, conforme dito anteriormente, o laço continuará indefinidamente (laço infinito). No exemplo acima, se a expressão condicional for modificada para `x >= 0.5`, o laço nunca terminará. Essas situações podem ser evitadas contando-se o número de repetições antes de parar o laço. Isso pode ser feito adicionando-se o número máximo admissível de repetições ou utilizando um comando `break` (Seção 7.6).

Mesmo os melhores programadores cometem erros e um *loop* infinito pode ocorrer por descuido do programador. Caso isso aconteça, o usuário pode parar a execução do programa pressionando, simultaneamente, as teclas **Ctrl+C** ou **Ctrl+Break**.

Problema-exemplo 7-7: Representação em série de potência de uma função (série de Taylor)

A função $f(x) = e^x$ pode ser representada em série de Taylor por: $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

Escreva um rotina que determine e^x usando a representação em série de Taylor. O programa deve calcular e^x adicionando-se os termos da série e parar quando o valor absoluto da última parcela adicionada tornar-se menor que 0.0001. Use um laço `while-end`, mas limite o número de repetições a 30. Se na trigésima repetição o valor absoluto do termo

ainda for maior que 0.0001, o programa deve parar e exibir uma mensagem do tipo: "São necessários mais de 30 termos para a representação em série de Taylor com a precisão requerida.".

Teste o programa para calcular e^2, e^{-4} e e^{21}.

Solução

Alguns dos primeiros termos dessa série são:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

A seguir está indicado um programa que utiliza a série para calcular a função. O programa solicita ao usuário que entre com o valor de `x`. O primeiro termo da série é declarado como `an = 1` e será adicionado ao restante da soma `S`. Em seguida, do segundo termo em diante, o programa chama um laço `while` para calcular o *n*-ésimo termo da série e adicioná-lo à soma (`S`). O programa também conta o número de termos já utilizados. A expressão condicional no comando `while` é verdadeira enquanto o valor absoluto do *n*-ésimo termo for maior ou igual 0.0001 e o número de repetições `n` for menor ou igual a 30. Significa que, se o trigésimo termo não for menor que 0.0001, o programa irá parar e exibirá a mensagem.

```
x = input('Entre com o valor de x: ');
n = 1; an = 1;S=an;
while abs(an)>= 0.0001&n<= 30
   an = x^n/factorial(n);
   S = S+an;
   n = n+1;
end
if n>= 30
   disp('Sao necessarios mais de 30 termos
    para a representacao em serie de Taylor com a...
    precisao requerida.')
else
fprintf('exp(%f)=%f',x,S)
fprintf('\nO numero de termos utilizado foi: %i',n)
fprintf('\nO n-esimo termo e:%f',n)
end
```

Início do laço `while`.

Calculando o *n*-ésimo termo.

Adicionando o *n*-ésimo termo à soma.

Contando o número de repetições.

Fim do laço `while`.

Laço `if-else-end`.

O programa se utiliza de uma estrutura `if-else-end` para exibir os resultados. Se o *loop* for terminado porque o trigésimo termo não ficou menor que 0.0001 (em módulo), o programa exibirá uma mensagem indicando o ocorrido. Além disso, são exibidos o valor da função, o número de termos utilizados e o valor do *n*-ésimo termo. Perceba que o

número de repetições depende do valor da variável x. Testando a rotina (salva como Capitulo7_Exemplo7) na linha do *prompt* da janela Command Window para e^{2}, e^{-4} e e^{21}:

```
>> Capitulo7_Exemplo7
Entre com o valor de x: 2                          Calculando exp(2).
exp(2.000000) = 7.389046
O numero de termos utilizado foi: 12               Foram utilizados 12 termos.
O n-esimo termo e:0.000051                         O valor do n-ésimo termo.
>> Capitulo7_Exemplo7
Entre com o valor de x: –4                         Calculando exp(–4).
exp(–4.000000) = 0.018307
O numero de termos utilizado foi: 18               Foram utilizados 18 termos.
O n-esimo termo e:-0.000048
>> Capitulo7_Exemplo7
Entre com o valor de x: 21                         Tentando calcular exp(21).
Sao necessarios mais de 30 termos para a representacao em serie de Taylor com a precisao requerida.
```

```
else
   fprintf('exp(%f)=%f',x,S)
   fprintf('\nO numero de termos utilizado foi: %i',n)
   fprintf('\nO n-esimo termo e:%f',an)
end
```

7.5 LAÇOS ANINHADOS E NINHOS DE SENTENÇAS CONDICIONAIS

Laços e sentenças condicionais podem ser aninhadas entre si. Isso significa que um *loop* ou uma sentença condicional pode ser inicializada (e terminada) dentro de outro *loop* ou sentença condicional. Não há limites para o número de laços e sentenças condicionais que podem ser aninhadas. Entretanto, deve ser salientado que toda estrutura `if`, `case`, `for` e `while` deve terminar em um comando `end` correspondente. A Figura 7-7 ilustra a estrutura de um ninho de `for-end` dentro de outro laço `for-end`.

Nos laços mostrados na figura, se, por exemplo, `n` = 3 e `m` = 4, então o primeiro `k` = 1 e o laço aninhado são executados quatro vezes, para `h` = 1, 2, 3 e 4. Em seguida, `k` = 2 e o laço aninhado são executados novamente outras quatro vezes. Por último, `k` = 3 e outra seqüência de quatro laços internos são executados. Toda vez que laços aninhados são digitados, o MATLAB automaticamente promove a indentação do laço mais interno, relativamente ao(s) laço(s) externo(s). O Problema-exemplo 7-8 demonstra o uso de ninhos de laços e sentenças condicionais.

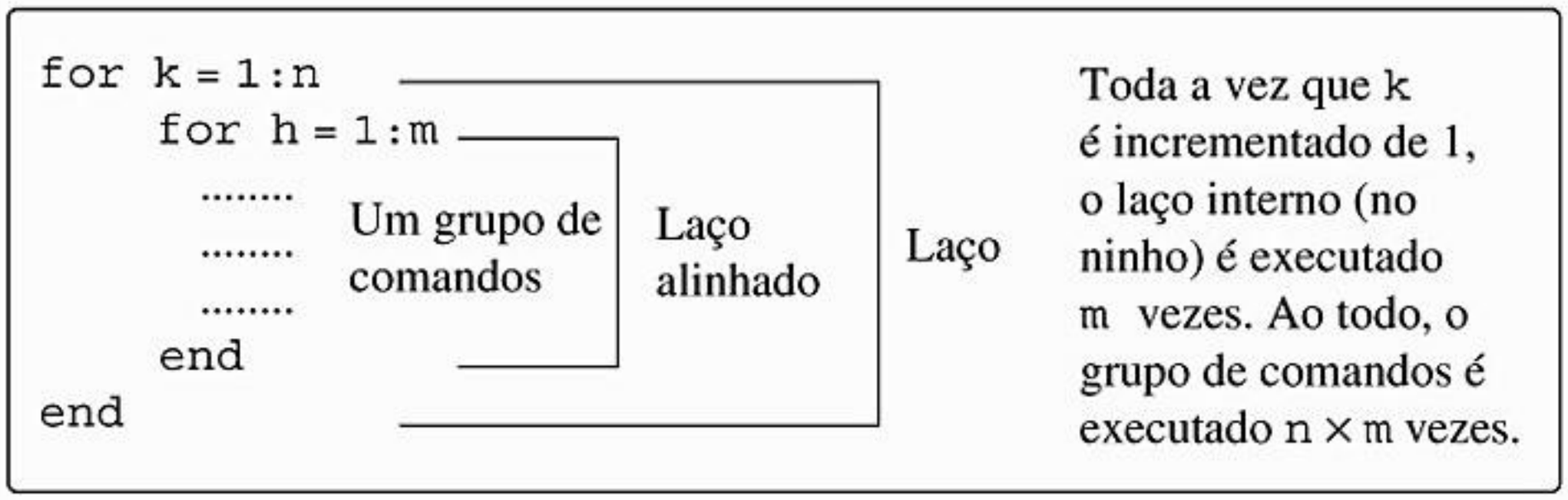


FIGURA 7-7 Estrutura de ninhos de *loops*.

Problema-exemplo 7-8: Criando uma matriz com um loop

Escreva uma rotina que crie uma matriz $m \times n$ cujos elementos tenham os seguintes valores: os valores dos elementos da primeira linha seguem a numeração das colunas. Os valores dos elementos da primeira coluna seguem a numeração das linhas. Os demais elementos são iguais à soma dos elementos imediatamente acima e à esquerda do elemento em questão. Quando executado, o programa deve solicitar ao usuário entrar com os valores de m e n.

Solução

O programa a seguir possui dois laços (um ninho) e uma sentença condicional aninhada `if-elseif-else-end`. Os elementos na matriz são inicializados um a um, linha por linha. As variáveis índice `k` e `h` endereçam as linhas e as colunas, respectivamente, do primeiro e segundo *loop*.

```
m = input('Entre com o numero de linhas da matriz: ');
n = input('Entre com o numero de colunas da matriz: ');
A = [];                                          % Declara a matriz A e não a inicializa.
for k = 1:n                                      % Início do primeiro loop for-end.
   for h = 1:m                                   % Início do segundo loop for-end.
      if k = = 1                                 % Início da sentença condicional.
         A(k,h) = h;                             % Atribui valores aos elementos da primeira linha.
      elseif h = = 1
         A(k,h) = k;                             % Atribui valores aos elementos da segunda linha.
      else
         A(k,h) = A(k,h-1) + A(k-1,h);           % Atribui valores aos outros elementos.
      end                                        % end da sentença if.
   end                                           % end do loop for-end interno.
end                                              % end do loop for-end externo.
A
```

Executando a rotina (salva como Capitulo7_Exemplo8) na linha do *prompt* da janela Command Window é gerada um matriz 4×5.

```
>> Capitulo7_Exemplo8
Entre com o numero de linhas: 4
Entre com o numero de colunas: 5
A =
     1   2   3   4   5
     2   4   7   11  16
     3   7   14  25  41
     4   11  25  50  91
```

7.6 OS COMANDOS `break` E `continue`

O comando `break`***:***

- Quando inserido em um laço, `for` ou `while`, provoca a saída imediata do laço. Assim, quando o MATLAB encontra um comando `break` dentro do laço, o programa salta para o comando `end` desse laço e segue para o próximo comando imediatamente abaixo, saindo incondicionalmente do laço.
- Inserido dentro de um *loop* aninhado, finaliza somente o *loop* que contém o `break`.
- Aparecendo na estrutura de uma rotina ou função, provoca o término imediato da execução das mesmas, a partir do ponto onde está situado.
- Geralmente, aparece dentro de sentenças condicionais. Caso uma certa condição de teste seja satisfeita, é um modo rápido e seguro de provocar a parada na execução do processo de *looping*. Por exemplo, quando o número de laços exceder um valor predeterminado ou acontecer um erro em algum procedimento numérico. Nesses casos, o comando `break` provoca uma saída imediata, para que dados incorretos (inconsistentes com o esperado) sejam propagados pela função.

O comando `continue`:

- É utilizado dentro de um laço, `for` ou `while`, de maneira a parar o passo atual e iniciar o próximo passo no processo de *looping*.
- Geralmente, o comando `continue` entra como parte de uma sentença condicional. Quando o MATLAB encontra o comando `continue`, deixa de executar o restante do laço atual, saltando para o comando `end` no final do laço, para então iniciar um novo passo.

7.7 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 7-9: Saques em uma conta de poupança

Um aposentado depositou \$300,000 em uma caderneta de poupança que paga uma taxa de juros anual de 5%. Ele planeja sacar o dinheiro da conta uma vez a cada ano, come-

çando com um saque de $25,000 no primeiro ano e incrementando a quantia anual de acordo com a inflação. Por exemplo, se a inflação anual for 3%, ele sacará $25,750 após o segundo ano. Sabendo disso, determine a quantidade de saques que ele poderá fazer na conta (em número de anos), considerando uma inflação anual constante de 2%. Construa um gráfico que mostre os saques anuais em função do saldo da conta ao longo dos anos.

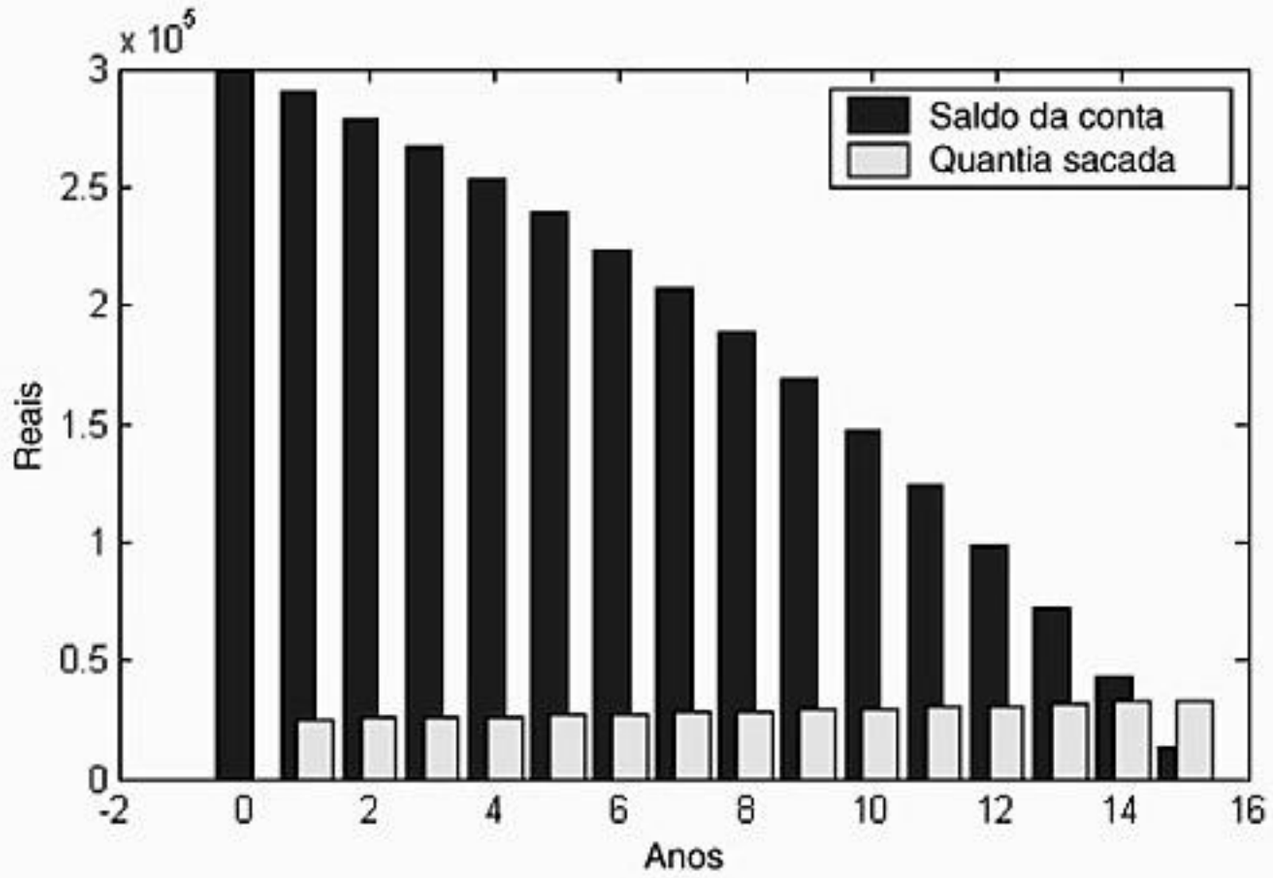
Solução

O problema será resolvido usando um *loop*; no caso, `while`, porque o número de repetições não é conhecido de antemão. A cada passo, a quantia a ser sacada e o novo saldo da conta devem ser calculados. O processo de *looping* continua até que o saldo da conta seja menor ou igual à quantidade a ser sacada. O programa a seguir resolve o problema usando uma rotina. Nesse programa, as variáveis `ano`, `saque` e `saldo` são vetores que representam o número de anos, a quantia sacada anualmente e o saldo anual da conta, respectivamente.
A rotina (salva como Capitulo7_Exemplo9) executada na janela Command Window:

```
juros = 0.05; inf = 0.02;

clear saque saldo ano

ano(1) = 0;                                  % Primeiro elemento é o ano 0.

saque(1) = 0;                                % Quantia sacada no ano 0.

saldo(1) = 0;                                % Saldo da conta no ano 0.

prosaque = 25000;                            % Quantia a ser sacada no ano 1.

saldocor = 300000*(1+juros);                 % Saldo da conta no final do ano 1.

n = 2;

    while saldocor >= prosaque               % while verifica se o saldo corrigido é
                                             % maior que o próximo saque.
        ano(n) = n-1;

        saque(n) = prosaque;                 % Quantia a ser sacada no ano n – 1.

        saldo(n) = saldocor-saque(n);        % Saldo da conta no ano n – 1, pós-saque.

        saldocor = saldo(n)*(1+juros);       % Saldo corrigido no final de um ano.

        prosaque = saque(n)*(1+inf);         % Quantia a ser sacada no final do
                                             % próximo ano.
        n = n+1;

    end

fprintf('Os saques serao efetuados durante %.f anos.',ano(n-1))

bar(ano,[saldo' saque'],2.0)
```

```
>> Capitulo7_Exemplo9
Os saques serao efetuados durante 15 anos.
A ultima quantia sacada sera de R$32986.97.
```

O programa também gera a figura a seguir (os rótulos dos eixos e a legenda foram adicionados manualmente na janela Figure, menu Insert + X Label, Insert + Y Label e Insert + Legend):



Problema-exemplo 7-10: Gerando números para aposta na loteria

Em uma certa loteria, o apostador deve selecionar alguns números dentre uma lista de números predeterminados. Escreva uma função que gere uma lista de n números inteiros distribuídos uniformemente entre os números a e b. Todos os números selecionados na lista devem ser diferentes entre si. Teste a função para gerar:

a) uma lista de 6 números escolhidos entre os números 1 e 49.

b) uma lista de 8 números escolhidos entre os números 60 e 75.

c) uma lista de 9 números escolhidos entre os números −15 e 15.

Solução

A função personalizada mostrada a seguir utiliza o uso da função `rand`, nativa do MATLAB (veja a Seção 3.7). Para assegurar que todos os números sejam diferentes entre si, os números são escolhidos individualmente. Cada número escolhido pela função `rand` é comparado com os demais. Caso haja uma correspondência, o número é descartado e um novo número é escolhido.
O *loop* `while` verifica o número escolhido, comparando-o com os demais no vetor `x`. Se for encontrada alguma correspondência, continua a escolher números até que uma nova entrada, diferente das demais, seja detectada.

```
function x =loto(a,b,n)
% loto seleciona n numeros (diferentes) a partir de um dominio a,b.
% x e um vetor de n numeros.
for p = 1:n
   numero = round((b-a)*rand+a);
   if p = =1
   x(p) = numero;
else

   r = 0;
   while r = = 0
      r = 1;
      for k = 1:p-1
         if x(k) = = numero
            numero = round((b-a)*rand+a);
            r = 0;
            break
         end
      end
   end
end
   x(p) = numero;
end
```

Gera um número inteiro entre *a* e *b*.

Se o número-escolhido for o primeiro, atribui a *x*(1).

Senão, o número recém escolhido é comparado com os números anteriores.

Inicializa `r` com 0.

Leia a explicação abaixo.

Atribui 1 a `r`.

O *loop* `for` compara o novo número com os demais em `x(p)`.

Caso haja uma correspondência, um novo número é escolhido e a `r` é atribuído o valor 0.

O *loop* `for` interno é imediatamente terminado. O programa retorna ao *loop* `while`. Como `r = 0`, o *loop* `for`, interno a `while`, é repetido com um novo número.

O número que não corresponder a nenhum dos elementos escolhidos é atribuído a `x(p)`.

A função (salva como `loto`) é testada a seguir para os três casos sugeridos no enunciado do problema:

```
>> loto(1,49,6)
ans =
    44  4   18  40  1   8
>> loto(60,75,8)
ans =
    63  69  64  60  71  67  74  66
>> loto(-15,15,9)
ans =
    10  1   –9  5   –14    -4  0   6   –2
```

*Problema-exemplo 7-11: Modelo para a equação de movimento de um foguete**

A trajetória de um pequeno foguete pode ser modelada como segue: durante os primeiros 0.15s, o foguete é propulsionado com uma força de 16N. Assim, o foguete segue em vôo vertical enquanto a força da gravidade age, puxando-o para baixo. Atingindo o ápice da trajetória, o foguete começa a cair, sob a ação da força da gravidade. Quando a velocidade do foguete atinge 20m/s, um sistema de pára-quedas é aberto para reduzir a velocidade de queda do foguete (assumindo que o pára-quedas abra instantaneamente) e o foguete continua a queda a uma velocidade constante de 20m/s até atingir o solo. Escreva um programa que calcule e plote a velocidade e a altura do foguete em função do tempo de vôo.

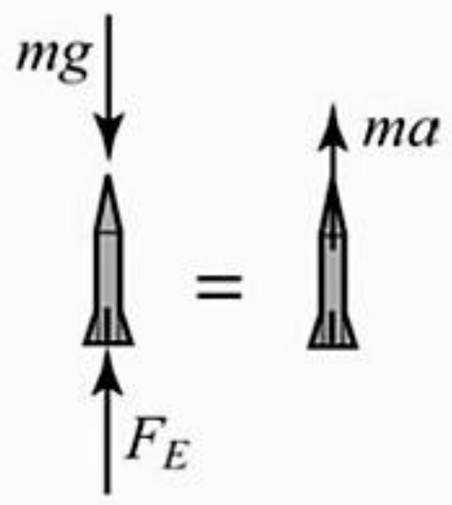
Solução

Consideremos que o foguete é uma partícula que se move ao longo de uma linha reta na vertical. Para o movimento retilíneo com aceleração constante, a velocidade e a posição de uma partícula são dadas, respectivamente, por:

$$v(t) = v_0 + at \quad \text{e} \quad s(t) = s_0 + v_0 t + \frac{1}{2}at$$

onde v_o e s_o são a velocidade e a posição iniciais, respectivamente. No programa, o vôo do foguete será dividido em três partes. Cada uma delas é calculada por um laço `while`. Em cada passo, o tempo é incrementado de um infinitésimo.

Parte 1: os primeiros 0.15s, quando o propulsor está ligado. Durante esse período, o foguete move-se verticalmente para cima. A aceleração pode ser determinada desenhando-se as forças em um diagrama de corpo livre e a partir do diagrama da aceleração resultante (mostrado à direita). Da segunda Lei de Newton, a soma das forças na direção vertical é igual a massa vezes aceleração do foguete (equação de equilíbrio):



$$+\uparrow \sum F = F_E - mg = ma$$

Resolvendo a equação para a aceleração:

$$a = \frac{F_E - mg}{m}$$

A velocidade e a altura em função do tempo são, respectivamente:

$$v(t) = 0 + at \quad \text{e} \quad h(t) = 0 + 0 + \frac{1}{2}at^2$$

onde a velocidade e a posição iniciais são nulas. Na rotina, essa parte do problema inicia quando $t = 0$, e o *looping* continua enquanto $t < 0.15$s. A velocidade, a altura e o tempo são denotados por v_1, h_1 e t_1, respectivamente.

---

* N. de T.: O modelo real para a equação de movimento de um foguete deve levar em consideração, entre outros aspectos, o fato de que o sistema foguete + gases é um sistema de massa variável.

Parte 2: o movimento do foguete desde o instante em que o motor pára até que o pára-quedas seja aberto. Nessa parte, o foguete move-se com uma desaceleração constante $-g$. A velocidade e a altura do foguete em função do tempo são dadas, respectivamente, por:

$$v(t) = v_1 - g(t - t_1) \quad \text{e} \quad h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2$$

O processo de *looping* continua até que a velocidade final do foguete atinja –20m/s (negativa, porque o foguete move-se para baixo). A altura e o tempo são denotados por h_2 e t_2.

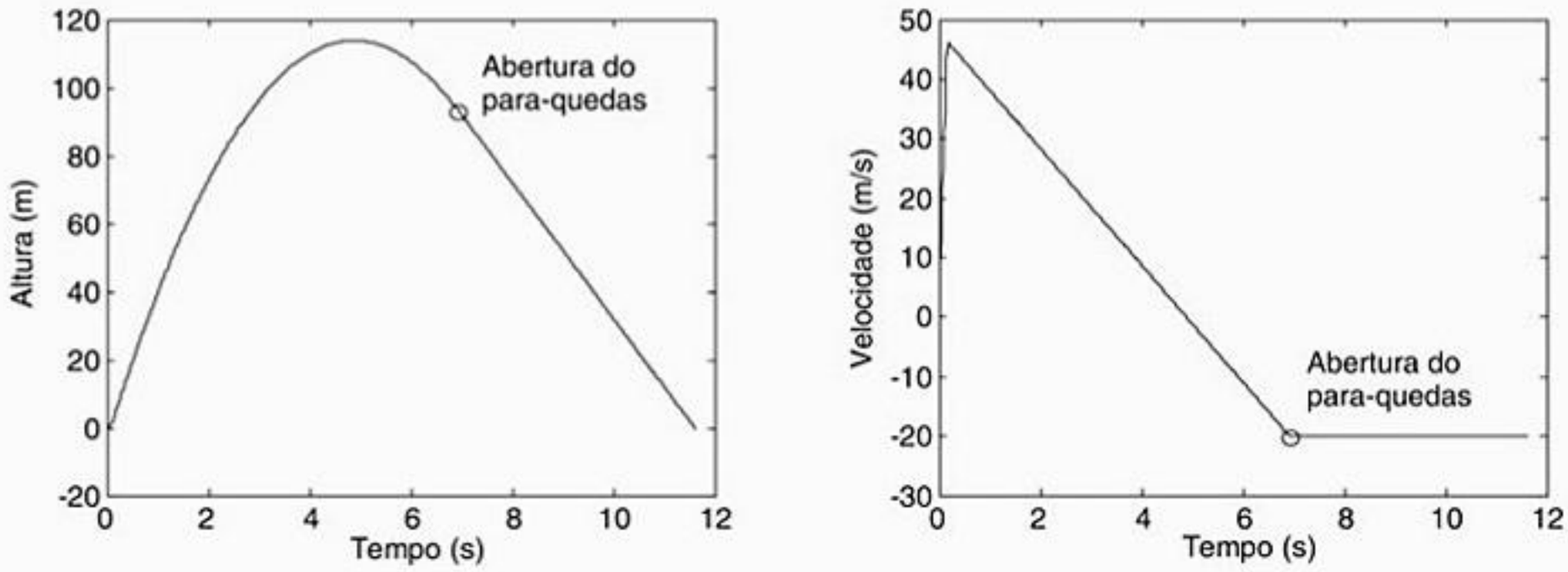
Parte 3: o movimento a partir da abertura do pára-quedas até que o foguete atinja o solo. O foguete move-se a uma velocidade constante (aceleração zero). A altura em função do tempo é dada por: $h(t) = h_2 - v_{\text{pára-quedas}}(t - t_2)$, onde $v_{\text{pára-quedas}}$ é a velocidade constante após a abertura do pára-quedas. O processo de *looping* continua enquanto a altura é maior que zero.

Uma rotina que desenvolve esses cálculos é mostrada a seguir:

```
m = 0.05; g=9.81; tmotor = 0.15; Forca=16; vpara_quedas=-20; Dt=0.01;
clear t v h
n=1;
t(n) = 0; v(n) = 0; h(n) = 0;
% Parte 1
a1 = (Forca – m*g)/m;
while t(n)<tmotor & n < 50000                          Primeiro loop while.
   n = n + 1;
   t(n) =  t(n-1) + Dt;
   v(n) = a1*t(n);
   h(n) = 0.5*a1*t(n)^2;
end
v1 = v(n);h1 = h(n); t1 = t(n);
% Parte 2
while v(n)> = vpara_quedas & n < 50000                 Segundo loop while.
   n = n+1;
   t(n) = t(n-1)+Dt;
   v(n) = v1-g*(t(n)-t1);
   h(n) = h1+v1*(t(n)-t1)-0.5*g*(t(n)-t1)^2;
end
v2 = v(n); h2 = h(n);t2 = t(n);
% Parte 3
while h(n)>0 & n < 50000                               Terceiro loop while.
   n = n+1;
   t(n) = t(n-1)+Dt;
```

```
v(n) = vpara_quedas;
h(n) = h2 + vpara_quedas*(t(n)-t2);
end
subplot(1,2,1)
plot(t,h,t2,h2,'o')
subplot(1,2,2)
plot(t,v,t2,v2,'o')
```

A precisão dos resultados depende do valor do incremento `Dt`. Um incremento de 0.01 apresenta bons resultados. A expressão condicional no comando `while` inclui uma condição para `n` (se `n` for maior que 50,000, o laço é terminado). Isso é uma precaução para evitar um *loop* infinito, caso haja um erro em alguma sentença dentro do *loop*. Os gráficos gerados pela rotina (salva como Capitulo7_Exemplo11) são mostrados a seguir (os títulos e as escalas adotadas para os eixos foram adicionados manualmente):

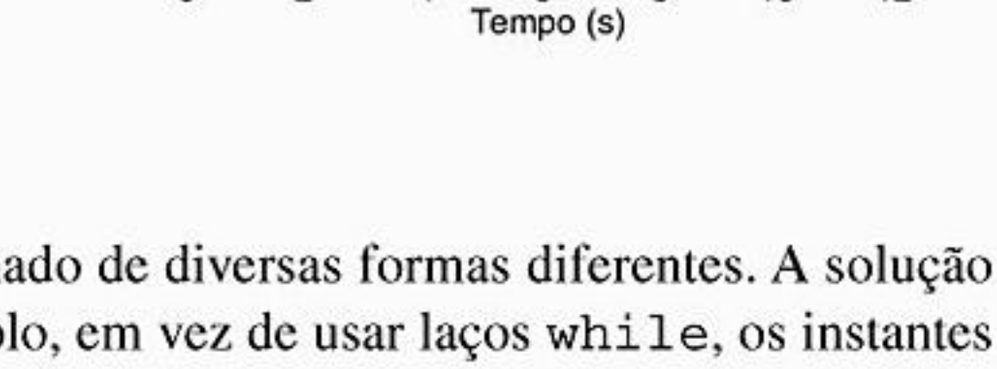


Observação:

O problema pode ser resolvido e programado de diversas formas diferentes. A solução dada aqui é bastante particular. Por exemplo, em vez de usar laços `while`, os instantes de tempo para o pára-quedas ser aberto e o foguete atingir o solo podem ser calculados primeiro e laços `for-end` podem ser utilizados no lugar dos laços `while`. Se os tempos são determinados em primeiro lugar é possível usar cálculos elemento a elemento do vetor, em vez de laços.

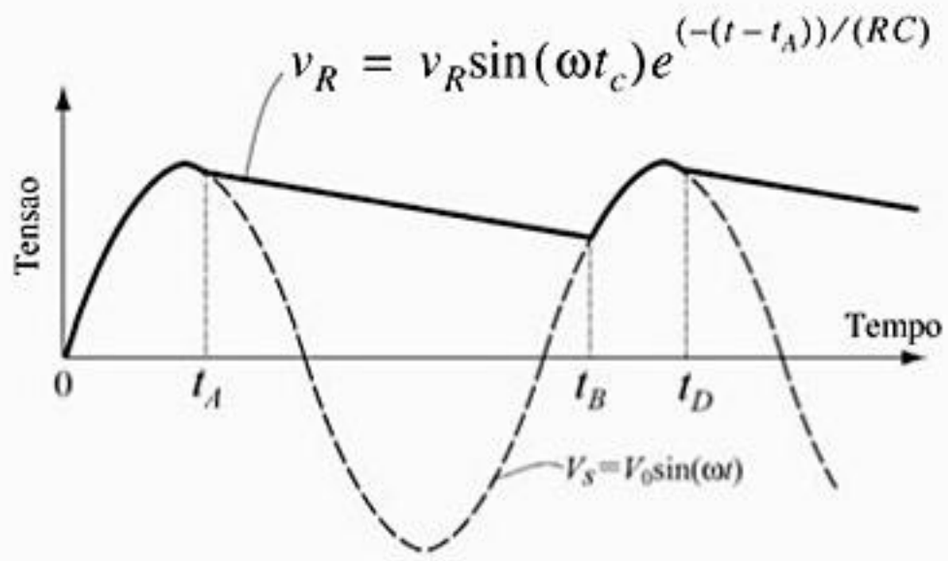
Problema-exemplo 7-12: Conversor CA/CC (circuito retificador)

O circuito retificador de meia-onda a diodo é um circuito eletrônico que, em síntese, converte uma tensão CA em uma tensão CC. Um circuito retificador de meia-onda é constituído por uma fonte CA, um diodo, um capacitor e uma carga (no caso, um resistor), mostrado na figura a seguir.



A tensão da fonte varia de acordo com $v_s = v_o \sin(\omega t)$, onde $\omega = 2\pi f$ e f é a freqüência da fonte CA.

A operação do circuito pode ser ilustrada nas formas de onda, onde a linha tracejada corresponde à forma de onda da fonte CA e a linha cheia mostra a queda de tensão através do resistor. No primeiro ciclo, o diodo está ligado (conduzindo corrente) de $t = 0$ a $t = t_A$. Durante o intervalo de tempo que o diodo está desligado (polarizado reversamente), a potência consumida pela carga é toda fornecida pelo capacitor em descarga (de $t = t_A$ a $t = t_B$). Em $t = t_B$, o diodo é ligado novamente (polarizado diretamente) e passa a conduzir corrente, alimentando a carga e repondo as perdas do capacitor até $t = t_D$. O ciclo será repetido enquanto a fonte de tensão estiver ligada. Nessa análise simplificada, consideraremos que o diodo é ideal e que o capacitor não possui nenhuma carga inicial (em $t = 0$). Quando o diodo está ligado, a queda de tensão e a corrente através do resistor são dadas por:



$$v_R = v_0 \sin(\omega t) \quad \text{e} \quad i_R = v_0 \sin(\omega t) / R$$

A corrente no capacitor é:

$$i_C = \omega C v_0 \cos(\omega t)$$

Quando o diodo está desligado, a queda de tensão através da carga é dada por:

$$v_R = v_0 \sin(\omega t_A) e^{(-(t-t_A))/(RC)}$$

Os instantes em que o diodo passa ao estado desligado (t_A, t_D, etc.) são calculados a partir da imposição de que $i_R = -i_C$.O diodo chaveia para o estado ligado quando a fonte se torna maior ou igual à tensão no resistor (o instante t_B na figura).

Escreva uma rotina no MATLAB que construa as formas de onda no resistor v_R e na fonte v_s, em função do tempo para $0 \leq t \leq 70$ms. A resistência da carga é 1.8KΩ, a fonte de tensão possui uma amplitude $v_0 = 12$V e $f = 60$Hz. Estude o efeito da capacitância do capacitor sobre a tensão da carga testando o programa para dois valores de capacitância, $C = 45\mu$F e $C = 10\mu$F.

Solução

A seguir, temos uma rotina que resolve o problema. O programa foi dividido em duas partes: a primeira calcula a tensão v_R quando diodo está no estado ligado e a segunda calcula a tensão v_R quando o diodo está desligado. O comando `switch` é utilizado para comutar entre as duas partes do programa. Os cálculos começam com o diodo no estado ligado (variável `estado = 'on'`) e, quando $i_R - i_C \leq 0$, o valor da variável `estado = 'off'`, comutando para o grupo de comandos que determinam v_R para esse estado. Esses cálculos se repetem até que $v_s \geq v_R$, quando o programa retorna às equações válidas para o diodo no estado ligado.

```
V0 = 12; R = 1800; f = 60;
Tf = 70e-3; w = 2*pi*f;
```

```
clear t VR Vs
t = 0:0.05e-3:Tf;
n = length(t);
estado = 'on';
for i = 1:n
    Vs(i) = V0*sin(w*t(i));
switch estado
    case 'on'
    VR(i) = Vs(i);
    iR = Vs(i)/R;
    iC = w*C*V0*cos(w*t(i));
    sumI = iR+iC;
    if sumI <= 0
        estado ='off';
        tA = t(i);
    end
    case 'off'
        VR(i) = V0*sin(w*tA)*exp(-(t(i)-tA)/(R*C));
        if Vs(i) >= VR(i)
            estado = 'on';
        end
    end
end
plot (t, Vs,':',t,VR,'k','linewidth',1)
xlabel('Tempo(s)'); ylabel('Tensao (V)')
```

Atribui `'on'` à variável `estado`.

Determina a tensão da fonte num tempo *t*(i).

Diodo ligado.

Verifica se $i_R - i_C \leq 0$.

Caso seja, atribui `'off'` à variável `estado`.

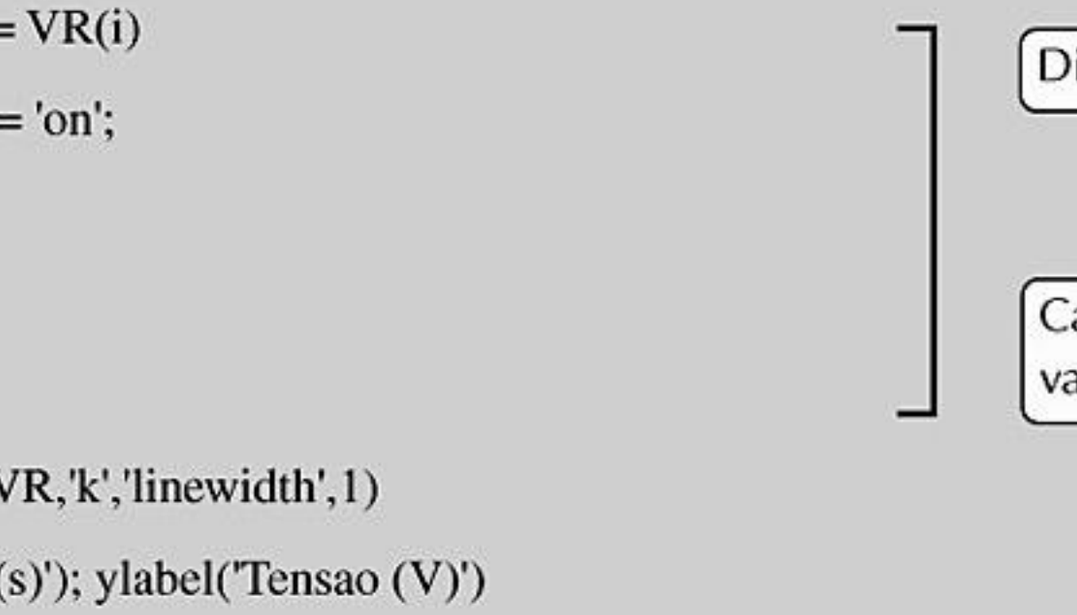
Atribui o valor de t(i) a t_A.

Diodo desligado.

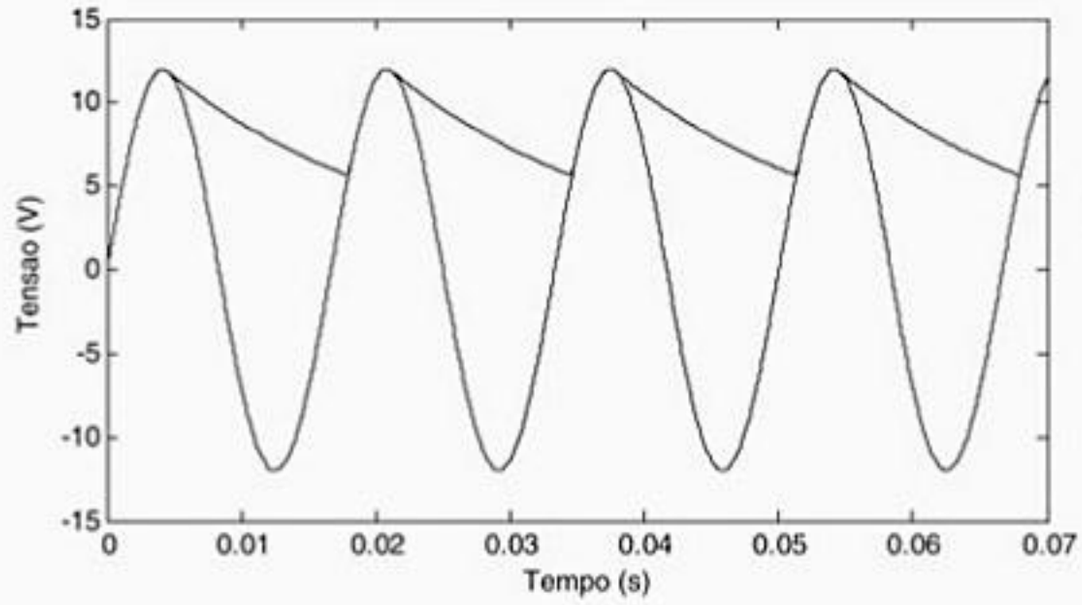
Verifica se $v_s \geq v_R$.

Caso seja, atribui `'on'` à variável `estado`.

Testando a rotina (salva como Capitulo7_Exemplo12) na janela Command Window:



$C = 45\ \mu F$



$C = 10\ \mu F$

7.8 PROBLEMAS

1. Verifique as seguintes expressões sem usar o MATLAB. Em seguida, compare suas respostas utilizando o MATLAB.

a) $5 <= 8 - 3$

b) $y = 7 < 3 - 1 + 6 > 2$

c) $y = (7 < 3) - 1 + (6 > 2)$

d) $y = 2 \times 4 + 5 == 7 + \frac{20}{4}$

2. Considere $a = 10$ e $b = 6$. Verifique as seguintes expressões sem a ajuda do MATLAB. Em seguida, use o MATLAB e compare com os seus resultados.

a) $y = a >= b$

b) $y = a - b <= \frac{b}{2}$

c) $y = a - (b <= \frac{b}{2})$

3. Considere v = [4 –2 –1 5 0 1 –3 8 2] e w = [0 2 1 –1 0 –2 4 3 2]. Verifique as seguintes expressões sem ajuda do MATLAB. Em seguida, use o MATLAB e compare com os seus resultados.

a) $v >= w$

b) $w \sim= v$

4. Utilize os vetores v e w do problema anterior, mais operadores relacionais, para criar um vetor y formado pelos elementos de w maiores que os elementos de v.

5. Verifique as seguintes expressões sem a ajuda do MATLAB. Confira suas respostas utilizando o MATLAB.

a) 5&–2

b) 8–2|6+5&~2

c) ~(4&0)+8*~(4|0)

6. As temperaturas máximas diárias (em °F) de Nova York (NY) e Anchorage (Alasca), durante o mês de janeiro de 2001, são mostradas nos vetores abaixo (dados da U.S. National Oceanic and Atmospheric Administration).
TNY = [31 26 30 33 33 39 41 41 34 33 45 42 36 39 37 45 43 36 41 37 32 32 35 42 38 33 40 37 36 51 50]
TANC = [37 24 28 25 21 28 46 37 36 20 24 31 34 40 43 36 34 41 42 35 38 36 35 33 42 42 37 26 20 25 31]
Escreva uma rotina para responder às seguintes questões:

a) Calcule a temperatura média mensal em cada cidade.

b) Em quantos dias do mês de janeiro as temperaturas estiveram abaixo da média em cada cidade?

c) Em quantos e em quais dias do mês a temperatura em Anchorage esteve maior que a temperatura em Nova York?

d) Em quantos e em quais dias do mês as temperaturas foram as mesmas em ambas as cidades?

e) Em quantos e em quais dias do mês as temperaturas em ambas as cidades estiveram acima do ponto de congelamento da água (acima de 32°F)?

7. Use o MATLAB de duas formas diferentes, descritas abaixo, para plotar a função:

$$f(x) = \begin{Bmatrix} 4^{ex+2} & \text{para} & -6 \le x \le -2 \\ x^2 & \text{para} & -2 \le x \le -2.5 \\ (x+6.5)^{1/3} & \text{para} & -2.5 \le x \le -6 \end{Bmatrix}$$

a) Escrevendo uma rotina composta de sentenças condicionais e laços.

b) Declarando uma função para $f(x)$ e,então, usando-a em uma rotina para construir o gráfico.

8. Escreva uma rotina que determine as raízes reais da equação quadrática $ax^2 + bx + c = 0$. Salve a função como `raizquad`. Quando o arquivo for executado, o usuário deverá entrar com os valores das constantes a, b e c. Para calcular as raízes da equação, o programa deve primeiro calcular o discriminante D dado por:

$$D = b^2 - 4ac$$

Se $D > 0$, o programa deve exibir a mensagem: "A equação possui duas raízes reais.", e as raízes devem ser mostradas na próxima linha.
Se o $D = 0$, o programa deve exibir a mensagem: "A equação possui duas raízes iguais.", e a raiz deve ser mostrada na próxima linha.
Se $D < 0$, o programa deve exibir a mensagem: "A equação não possui raízes reais."

Teste a rotina na janela Command Window três vezes para obter a solução das seguintes equações:

a) $2x^2 + 8x - 3 = 0$

b) $15x^2 + 10x + 5 = 0$

c) $18x^2 + 12x + 2 = 0$

9. Use laços para criar uma matriz 4×7, onde o valor de cada elemento é a soma dos índices da matriz (número da linha e da coluna do elemento). Por exemplo, o valor elemento A(2,5) é 7.

10. Use laços e sentenças condicionais para criar uma matriz 5×8, onde o valor de cada elemento é igual à raiz quadrada da soma dos índices do elemento, a não ser que o elemento esteja em uma linha ou coluna onde os índices são números pares. Nesses casos, o valor de um elemento é igual à soma dos quadrados dos índices. (Os índices de um elemento em uma matriz são o número da linha e da coluna do elemento.)

11. Escreva um programa (usando um laço) que determine a soma dos m primeiros termos da série:

$$\sum_{n=0}^{m} (-1)^n \frac{1}{2n+1} \quad (n = 0, 1, 2, \ldots, m)$$

Rode o programa com $m = 10$ e $m = 500$. Compare o resultado encontrado com $\pi/4$. Essa série é denominada série de Leibniz e converge a $\pi/4$.

12. Um vetor é dado por: x =[15 –6 0 8 –2 5 4 –10 0.5 3]. Utilizando sentenças condicionais e laços, escreva uma rotina para determinar a soma dos elementos positivos do vetor.

13. Escreva uma rotina para encontrar o menor número inteiro ímpar divisível por 3, cuja terceira potência é maior que 4000.Utilize apenas um laço no programa. O laço deve iniciar em 1 e parar quando o número for encontrado. Em seguida, o programa deve imprimir uma mensagem: "O número requerido é:" e imprime o número.

14. Escreva uma função que ordene os elementos de um vetor do maior para o menor. Utilize `y =ordenadecres(x)` para o nome da função e dos argumentos. A função deve receber um vetor `x` de qualquer tamanho e retornar um vetor `y`, contendo os elementos de `x` organizados em ordem decrescente. Não use a função `sort` do MATLAB. Teste sua função em um vetor com 14 elementos distribuídos aleatoriamente entre –30 e 30. Use a função `rand` do MATLAB para gerar o vetor inicial.

15. Escreva uma função que ordene os elementos de uma matriz. Utilize `B = sortmatriz(A)` para o nome da função e dos argumentos, onde `A` é uma

matriz de qualquer dimensão e B é uma matriz com mesma dimensão de A, cujos elementos estão organizados em ordem crescente, linha após linha, onde o elemento (1,1) é o menor e o elemento (m,n) é o maior.
Teste sua função em uma matriz 4×7 cujos elementos sejam números inteiros distribuídos aleatoriamente entre –30 e 30. Utilize a função rand do MATLAB para gerar a matriz inicial.

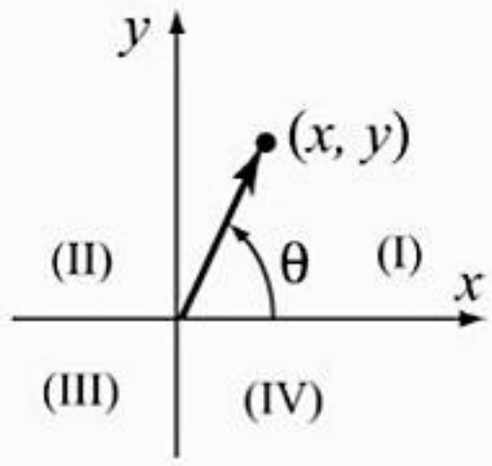
16. Escreva uma rotina que calcule o custo de postagem de um pacote, de acordo com a tabela de preços:

Tipo de serviço	**Peso 0 – 2 kg**	**Peso 2 – 10 kg**	**Peso 10 – 50 kg**
Terrestre	$1.50	$1.50 + $0.50 por quilo (ou fração) que exceder o limite de 2 kg.	$5.50 + $0.30 por quilo (ou fração) que exceder o limite de 10 kg.
Aéreo	$3.00	$3.00 + $0.90 por quilo (ou fração) que exceder o limite de 2 kg.	$10.20 + $0.60 por quilo (ou fração) que exceder o limite de 10 kg.
Expresso	$18.00	$18.00 + $6.00 por quilo (ou fração) que exceder o limite de 2 kg.	O serviço expresso não está disponível para pacotes acima de 10 kg.

O programa solicita ao usuário que entre com o peso e o tipo de serviço. Então, o programa mostra o custo de postagem. Se um peso maior que 50 kg for digitado para o serviço terrestre ou aéreo é exibida a mensagem "O serviço terrestre ou aéreo não está disponível para pacotes cujos pesos excedam 50 kg". Se um pacote pesando mais do que 10 kg for digitado para o serviço expresso, será exibida a mensagem "O serviço expresso não está disponível para pacotes acima de 10 kg". Rode o programa e entre com pacotes pesando 0.5; 6.3; 20; e 50.4 kg para serviços terrestres e aéreos, e 2; 8.1; 13 kg para o serviço expresso.

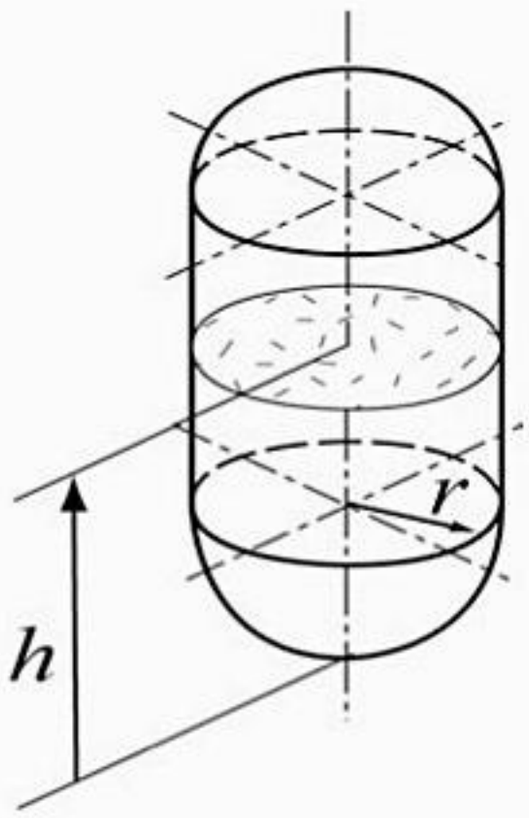
17. Um vetor x é dado por: x = [1:50]. Escreva uma rotina que remova todos os elementos do vetor divisíveis por 3, 4 ou 5. No final, apresente o novo vetor.

18. Escreva uma função que determine as coordenadas polares de um ponto escrito em coordenadas cartesianas. Utilize `[teta raio] = Cartesiano_Polar (x,y)` para o nome da função e dos argumentos. Os argumentos de entradas são as coordenadas x e y do ponto em coordenadas cartesianas e os argumentos de saída são o ângulo θ e a distância radial do ponto à origem. O ângulo θ é dado em graus e é medido relativamente ao eixo x, tal que ele cresça positivamente na direção dos quadrantes



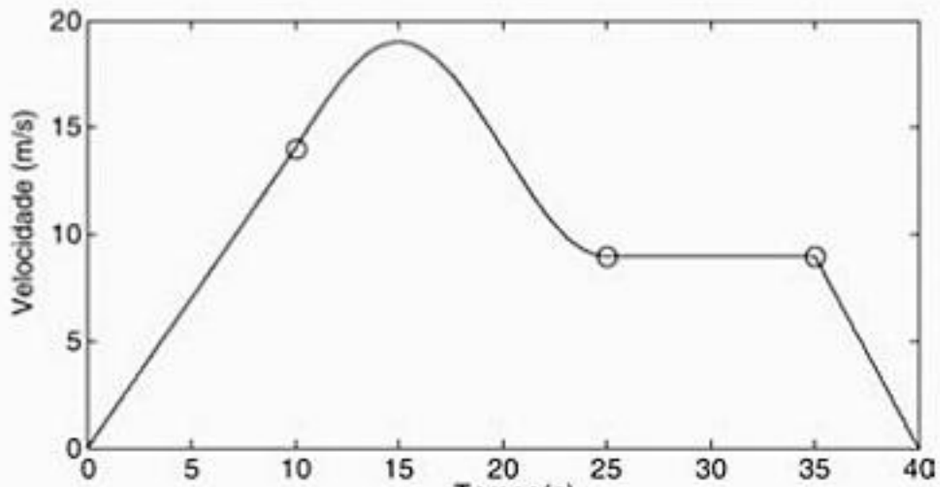
I,II,III, IV e cresça negativamente na direção dos quadrantes IV, III, II, I. Utilize a função para determinar as coordenadas polares dos pontos (15,3), (–7,12), (–17, –9) e (10; –6.5).

19. Um tanque de combustível cilíndrico possui as extremidades arredondadas, como dois hemisférios (veja a figura). O raio do cilindro e das calotas é $r = 40$cm e o comprimento da parte cilíndrica é 1.2m.



Escreva uma função (denominada `V=Voltanque (h)`) que retorne o volume de combustível no tanque, dada a altura h da coluna de combustível. Utilize a função para construir um gráfico, em função da altura h, para $0 \le h \le 2$ m.

20. A velocidade de uma partícula, em função do tempo, que se move retilineamente é dada no gráfico e nas equações a seguir.



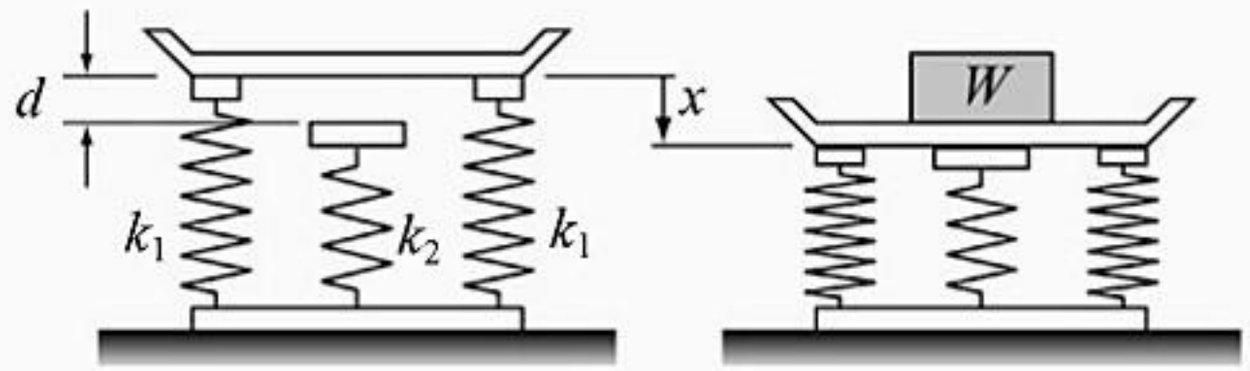
$$v(x) = \begin{cases} 1.4t & \text{para} \quad 0 \le t \le 10s \\ 14 + 5\sin\left(\dfrac{\pi}{10}(t-10)\right) & \text{para} \quad 10 \le t \le 25s \\ 9 & \text{para} \quad 25 \le t \le 35s \\ 9 - \dfrac{9}{5}(t-35) & \text{para} \quad 35 \le t \le 40s \end{cases}$$

Escreva duas funções: uma que calcule a velocidade da partícula em um tempo t (como o nome da função e argumentos utilize `v = velocidade (t)`), a outra que determine a aceleração da partícula em um tempo t (como nome da função e dos argumentos utilize `a = aceleracao (t)`). Em seguida, escreva uma rotina e utilize-a em um programa que gere os gráficos da velocidade e da aceleração em função do tempo (os dois gráficos na mesma janela). No programa, crie primeiramente o vetor t, $0 \le t \le 40$s, então use as funções `velocidade` e `aceleracao` para criar os vetores de velocidade e aceleração a serem plotados.

21. Uma balança é construída com base em uma bandeja suportada por molas (veja a figura a seguir). Quando um objeto é colocado sobre a bandeja, ela se move para baixo sob a ação da força peso do objeto. O peso do objeto pode ser determinado pelo deslocamento vertical da bandeja. Inicialmente, apenas as duas

molas de fora suportam o peso da bandeja. Se os objetos tiverem um peso suficiente, a bandeja toca na terceira bandeja do meio.

$k_1 = 800$ N/m, $k_2 = 1700$ N/m e $d = 20$ mm.



Escreva uma função que calcule o peso W do objeto para um dado deslocamento x da bandeja. Utilize `W = balanca (x)` como nome da função e dos argumentos.

a) Utilizando a função na janela Command Window, determine o peso de dois objetos para deslocamentos da bandeja de 1.5 e 3.1 cm.

b) Escreva uma rotina que gere os gráficos do peso em função dos deslocamentos para $0 \leq x \leq 4$ cm.