

MATLAB

MATLAB - MATrix LABoratory

Versão: R12. (versão 6.0)

ESTRUTURA DO MATLAB


Arquivos de comando tem extensão .m (M-files) e os arquivos de dados binários *default* tem extensão .mat (MAT-files).

Diretórios:

| | |
|----------|--|
| BIN | contém os programas binários do MATLAB; |
| DEMOS | demonstração em HTML; |
| EXTERN | suporte às linguagens FORTRAN, C e C++ |
| HELP | contém arquivos de auxílio e documentação do MATLAB; |
| JAVA | suporte à linguagem Java |
| NOTEBOOK | suporte à geração de documentação no formato MS-Word |
| TEMP | diretório temporário; |
| TOOLBOX | “toolboxes” do MATLAB |
| WORK | diretório de trabalho; |

PARA EXECUTAR O MATLAB

O prompt >>

 **No MATLAB, os comandos e variáveis escritos em letras maiúsculas são diferentes daqueles escritos em letras minúsculas ! Todos os comandos devem ser digitados em letras minúsculas. No *help* os comandos aparecem grafados em letras maiúsculas para melhorar a sua legibilidade.**

Sair do MATLAB, digitar o comando >> `quit` ou >> `exit` ou fechando o aplicativo como qualquer aplicativo Windows.

Para executar uma instrução do sistema Windows sem sair do MATLAB, digitar o comando precedido pelo sinal de exclamação. Exemplos:

```
>> !dir           Visualiza o diretório corrente
>> !format a:    Formata um disquete na unidade A
>> !copy arquivo a: Copia um arquivo para o disquete
>> !ren arq1 arq2 Muda o nome do arquivo arq1 para arq2
>> !del arquivo  Apaga o arquivo
>> !md pasta     Cria um subdiretório chamado pasta
```

ELEMENTOS BÁSICOS

Operações aritméticas (exemplos):

| | | |
|--|--|---|
| <pre>>> 1900/81 ans = 23.4568</pre> | <pre>>> A(2) ans = 2</pre> | <pre>>> sqrt(4) + 1 ans = 3</pre> |
| <pre>>> exp(0) ans = 1</pre> | <pre>>> A(1:2) ans = 1 2</pre> | <pre>>> pi ans = 3.1416</pre> |
| <pre>>> log(1) ans = 0</pre> | <pre>>> A = [1 2 3;4 5 6] A = 1 2 3 4 5 6</pre> | <pre>>> abs(-pi) ans = 3.1416</pre> |
| <pre>>> x = 2^3 x = 8</pre> | <pre>>> A = [1 2 3 4 5 6] A = 1 2 3 4 5 6</pre> | <pre>>> i ans = 0 + 1.0000i</pre> |
| <pre>>> y = x + 2 y = 10</pre> | <pre>>> A = [1 2 3 4 5 6] A = 1 2 3 4 5 6</pre> | <pre>>> 1 + i ans = 1.0000 + 1.0000i</pre> |
| <pre>>> y y = 10</pre> | <pre>>> A(:,1) ans = 1 4</pre> | <pre>>> i^2 ans = -1.0000 + 0.0000i</pre> |
| <pre>>> A = [1 2 3] A = 1 2 3</pre> | <pre>>> A(1,:) ans = 1 2 3</pre> | <pre>>> s = 1 - 1/2 + 1/3 - ... 1/4 + 1/5 - 1/6 + ... 1/7 - 1/8 + 1/9 - 1/10 s = 0.6456</pre> |

Operadores aritméticos:

| | |
|---|--------------------|
| + | adição |
| - | subtração |
| * | multiplicação |
| / | divisão a direita |
| \ | divisão a esquerda |
| ^ | potenciação |

Observação: $1/4 = 4 \setminus 1$ (= 0,25)

Exemplos de formatos de entrada de números:

```
3          -58          .14          -.84
0.0001
9.63754    1.60210E-2    6.02252e3
```

Formatos de saída de números:

Formato padrão de saída (format short)

Exemplo:

```
>> x = [4/3 1.234e-6]
x =
    1.3333    0.0000
```

Outros formatos:

```
>> format short e
>> x
x =
    1.3333e+000    1.2345e-006

>> format long
>> x
x =
    1.333333333333333    0.00000123450000

>> format long e
>> x
x =
    1.333333333333333e+000    1.234500000000000e-006
```

Gravação do espaço de trabalho:

```
>> save
```

salva todas as variáveis do espaço de trabalho corrente no arquivo chamado **matlab.mat**.

```
>> save dados
```

guardará os dados no formato binário no arquivo **dados.mat**. A extensão **.mat** é a padrão para arquivos de dados do Matlab.

A recuperação das variáveis após uma nova entrada no MATLAB se faz através da execução da instrução

```
>> load
```

Para recuperar as variáveis numa outra sessão de MATLAB, executar a instrução **load temp**.

A instrução **save temp X Y Z** salva o conteúdo das variáveis X, Y e Z no arquivo **temp.mat**.

O formato de arquivo **.mat** é binário. Se quisermos salvar o arquivo de dados no formato ASCII (texto puro) para utilizar em outros programas, digitar o comando

```
>> save dados.dat X Y -ascii
```

para salvar as variáveis X e Y no arquivo **dados.dat** no formato ASCII.

Quando o Matlab grava os dados no formato ASCII, ele utiliza o formato `short e`. No caso de carregamento de arquivo ASCII o Matlab cria uma única variável no espaço de trabalho com o nome do arquivo sem extensão.

CÁLCULO MATRICIAL

| Transposição | Adição e subtração | Multiplicação |
|--|--|---|
| <pre>>> A = [1 2 3;4 5 6;7 8 0] A = 1 2 3 4 5 6 7 8 0</pre> | <pre>>> A A = 1 2 3 4 5 6 7 8 0</pre> | <pre>>> A*B ans = 14 32 23 32 77 68 23 68 113</pre> |
| <pre>>> B = A' B = 1 4 7 2 5 8 3 6 0</pre> | <pre>>> B B = 1 4 7 2 5 8 3 6 0</pre> | <pre>>> b = A*x b = 5 8 -7</pre> |
| <pre>>> x = [-1 0 2] x = -1 0 2</pre> | <pre>>> C = A + B C = 2 6 10 6 10 14 10 14 0</pre> | <pre>>> pi*x ans = -3.1416 0 6.2832</pre> |
| <pre>>> x' ans = -1 0 2</pre> | <pre>>> D = C - B D = 1 2 3 4 5 6 7 8 0</pre> | <pre>>> 3*A - 5*B ans = -2 -14 -26 2 -10 -22 6 -6 0</pre> |
| <pre>>> x*x' ans = 1 0 -2 0 0 0 -2 0 4</pre> | <pre>>> x x = -1 0 2</pre> | |
| <pre>>> x'*x ans = 5</pre> | <pre>>> y = x - 1 y = -2 -1 1</pre> | |

Divisão de matrizes

Divisão à esquerda: $X = A \setminus B$ é a solução de $A * X = B$, pois $X = A^{-1} * B$

Divisão à direita: $X = B / A$ é a solução de $X * A = B$, pois $X = B * A^{-1}$

Exemplos:

| | | |
|--|--|--|
| <pre>>> A A = 1 2 3 4 5 6 7 8 0</pre> | <pre>>> A*z ans = 5 8 -7</pre> | <pre>>> S = y/x S = 0 0 -1.0000 0 0 -0.5000 0 0 0.5000</pre> |
| <pre>>> det(A) ans = 27</pre> | <pre>>> x x = -1 0 2</pre> | <pre>>> S^2 ans = 0 0 -0.5000 0 0 -0.2500 0 0 0.2500</pre> |
| <pre>>> b b = 5 8 -7</pre> | <pre>>> y y = -2 -1 1</pre> | <pre>>> S*S ans = 0 0 -0.5000 0 0 -0.2500 0 0 0.2500</pre> |
| <pre>>> z = A \ b z = -1 0 2</pre> | <pre>>> s = x \ y s = 0.8000</pre> | |

Operações elementares sobre matrizes

Dado $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 0]$

Calcular:

- poly - polinômio característico
- det(A) - determinante
- trace(A) - traço
- rank(A) - *rank*
- roots(p) - raízes do polinômio característico
- inv(A) - inversa da matriz
- eig(A) - auto-valores e auto-vetores

Comandos para visualização das informações do espaço de trabalho.

Exemplos:

```
>> who
```

```
Your variables are:
```

```
S ans s x y
```

```
>> whos
```

| Name | Size | Bytes | Class |
|------|------|-------|--------------|
| S | 3x3 | 72 | double array |
| ans | 3x3 | 72 | double array |
| s | 1x1 | 8 | double array |
| x | 3x1 | 24 | double array |
| y | 3x1 | 24 | double array |

```
Grand total is 25 elements using 200 bytes
```

OPERAÇÕES SOBRE TABELAS

Inicialmente, tratamos de operações sobre os elementos individuais das tabelas consideradas.

Adição e subtração de tabelas:

Funcionam como adição e subtração de matrizes.

Produto e divisão de tabelas (elemento por elemento), adicionar o ponto:

$A.*B$ $A./B$ $A.\backslash B$

Exemplos:

```
>> x = [1 2 3]; y = [4 5 6];
```

```
>> z = x.*y
```

```
z =
```

```
4 10 18
```

```
>> z = x.\y
```

```
z =
```

```
4.0000 2.5000 2.0000
```

Potência de tabelas:

```
>> z = x.^y
z =
    1    32   729
-----
>> z = x.^2
z =
    1    4    9
```

```
>> z = 2.^[x y]
z =
    2    4    8   16   32   64
```

Concatenação de tabelas e matrizes

```
>> a = [1 2 3]'
a =
    1
    2
    3
-----
>> b = [4 5 6]'
b =
    4
    5
    6
```

```
>> c = [a b]
c =
    1    4
    2    5
    3    6
-----
>> a'
ans =
    1    2    3
-----
>> b'
ans =
    4    5    6
```

```
>> [a';b']
ans =
    1    2    3
    4    5    6
```

MANIPULAÇÃO DE VETORES E MATRIZES

Criação de vetores:

```
>> x = 1:5
x =
    1    2    3    4    5

>> y = 0:.1:1
y =
Columns 1 through 7
    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
Columns 8 through 11
    0.7000    0.8000    0.9000    1.0000
```

Exemplos:

```
>> z = 0:pi/4:pi
z =
    0    0.7854    1.5708    2.3562    3.1416

>> u = 6:-1:1
u =
    6    5    4    3    2    1
```

```
>> u = (6:-1:1)'
u =
    6
    5
    4
    3
    2
    1
```

Criação de vetores por número de pontos: função linspace

```
>> k = linspace(-pi,pi,4)
k =
   -3.1416   -1.0472    1.0472    3.1416
```

Manipulação de matrizes:

| | | |
|--|--|--|
| <pre>>> A = [1 2 3;4 5 6;7 8 9] A = 1 2 3 4 5 6 7 8 9</pre> | <pre>>> b = A(:) b = 1 4 7 2 5 8 3 10</pre> | <pre>>> A(:) = 1:9 A = 1 4 7 2 5 8 3 6 9</pre> |
| <pre>>> A(3,3)=A(1,3)+A(3,1) A = 1 2 3 4 5 6 7 8 10</pre> | | <pre>>> A(:,[2,3]) = zeros(3,2) A = 1 0 0 4 0 0 7 0 0</pre> |

Matriz vazia: `x = []`

Exemplo de remoção de linhas e colunas de uma matriz:

| | |
|---|--|
| <pre>>> A(:) = 1:9 A = 1 4 7 2 5 8 3 6 9</pre> | <pre>>> A(:,[2 3])=[] A = 1 2 3</pre> |
|---|--|

Exemplos:

| | | |
|--|---|---|
| <pre>>> diag(1:4) ans = 1 0 0 0 0 2 0 0 0 0 3 0 0 0 0 4</pre> | <pre>>> diag(A) ans = 1 5 9</pre> | <pre>>> 3*ones(3) ans = 3 3 3 3 3 3 3 3 3</pre> |
| <pre>>> diag(1:4,1) ans = 0 1 0 0 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 4 0 0 0 0 0</pre> | <pre>>> diag(A,-1) ans = 4 8</pre> | <pre>>> eye(2,3) ans = 1 0 0 0 1 0</pre> |
| <pre>>> diag([1 -1 -8]) ans = 1 0 0 0 -1 0 0 0 -8</pre> | <pre>>> diag(A,2) ans = 3</pre> | <pre>>> eye(3) ans = 1 0 0 0 1 0 0 0 1</pre> |
| <pre>>> zeros(2,3) ans = 0 0 0 0 0 0</pre> | <pre>>> zeros(2) ans = 0 0 0 0</pre> | <pre>>> -1*eye(3) ans = -1 0 0 0 -1 0 0 0 -1</pre> |
| <pre>>> A = [1 2 3;4 5 6;7 8 9] A = 1 2 3 4 5 6 7 8 9</pre> | <pre>>> rand(4,3) ans = 0.2190 0.9347 0.0346 0.0470 0.3835 0.0535 0.6789 0.5194 0.5297 0.6793 0.8310 0.6711</pre> | <pre>>> rand(4,3) ans = 0.0077 0.6868 0.5269 0.3834 0.5890 0.0920 0.0668 0.9304 0.6539 0.4175 0.8462 0.4160</pre> |
| | | <pre>>> ones(3) ans = 1 1 1 1 1 1 1 1 1</pre> |

Construção de matrizes a partir de outras matrizes.

Exemplos:

```
>> A = [1 2;3 4] | >> C = [A eye(2);ones(2) A^2] | >> size(C)
A = | C = | ans =
    1 2 |    1 2 1 0 |    4 4
    3 4 |    3 4 0 1 |
    1 1 |    1 1 7 10 |
    1 1 |    1 1 15 22 |
```

GRÁFICOS NO MATLAB

Um dos recursos mais utilizados no Matlab é a sua capacidade de geração de gráficos em duas (2D) e três dimensões (3D), concomitante ou após o cálculo numérico.

O comando para desenhar gráficos x-y 2D é `plot(x,y)`, no qual `x` e `y` são vetores contendo os valores das variáveis `x` (eixo horizontal) e `y` (eixo vertical).

Exemplo:

Desenho de uma função senoidal com amplitude unitária, frequência de 1000 Hz e ângulo de fase igual a zero.

```
>> A = 1; f = 1000; p = 0; T = 1/f;
>> t = 0:T/100:5*T; s = A*sin(2*pi*f*t + p);
>> plot(t,s);
>> grid; title('Onda senoidal'); xlabel('Tempo (s)'); ylabel('Tensao (V)');
```

PROGRAMAÇÃO NO MATLAB

O Matlab pode ser também utilizado como linguagem de programação com uma construção sintática semelhante à linguagem C. Ela possui declarações como laços, testes e desvios condicionais, manipulação de arquivos e objetos gráficos. Os programas são criados como scripts "m-files".

Laços e Testes Condicionais

O Matlab tem os testes condicionais padrões if-elseif-else. Por exemplo:

```
>> t = rand(1);
>> if t > 0.75
    s = 0;
elseif t < 0.25
    s = 1;
else
    s = 1-2*(t-0.25);
end
>> s
s =
    0
>> t
t =
    0.7622
```


Os operadores lógicos no Matlab são: <, >, <=, >=, == (igualdade lógica) e ~= (não igual). Estes são operadores binários (tratados como números inteiros) que retornam os valores 0 (falso) e 1 (verdadeiro):

```
>> 5>3
ans =
     1
>> 5<3
ans =
     0
>> 5==3
ans =
     0
```

A forma geral da declaração if é:

```
if expr1
    declarações
elseif expr2
    declarações
    .
    .
else
    declarações
end
```

Uma declaração if é sempre encerrada com uma declaração end. A indentação é facultativa, porém, é recomendada o seu uso para tornar legível o programa.

O Matlab possui dois tipos de laços: o laço `for` padrão (comparável ao laço `for` da linguagem C) e o laço condicional `while`.

O laço `for` repete as instruções dentro do laço até que o índice contador do laço alcance a condição final:

```
>> for i=[1,2,3,4]
    disp(i^2)
end
     1
     4
     9
    16
```

(Observe o uso da função `disp`, que exhibe na tela o conteúdo do seu argumento). O laço `for`, tal como o bloco `if`, deve ser terminado com a instrução `end`. Este laço poderia ser expresso na forma mais comum:

```
>> for i=1:4
    disp(i^2)
end
     1
     4
     9
    16
```

(lembre-se que `1:4` é equivalente a `[1,2,3,4]`).

O laço `while` é executado enquanto a condição `expr` for verdadeira:

```
>> x=1;
```

```
>> while 1+x > 1
      x = x/2;
    end
>> x
x =
    1.1102e-16
```

Seleção de Casos (Switch)

O comando `switch` é utilizado quando desejamos selecionar (chavear) condicionalmente expressões porém, na forma de lista. O formato geral do comando `switch` é:

```
SWITCH expr
  CASE caso1,
    declaração, ..., declaração
  CASE {expr_caso1, expr_caso2, expr_caso_3,...}
    declaração, ..., declaração
  ...
  OTHERWISE,
    declaração, ..., declaração
END
```

As declarações que seguem o primeiro `CASE` que corresponde à condição dada por `expr` são executadas. Quando uma expressão é um vetor (como no segundo `CASE` acima), basta um `caso_expr` concordar com o `expr` da instrução `switch`. Se nenhum dos `CASES` concordar com o `expr` do `switch`, então o caso `OTHERWISE` é executado (se existir). Apenas um `CASE` é executado e a seqüência de execução do programa prossegue após o `END`. Apenas as instruções entre o `CASE` concordante e o próximo `CASE`, `OTHERWISE` ou `END` são executadas. Diferentemente da linguagem C, a instrução `SWITCH` não executa o `CASE` seqüencialmente (de modo que não há necessidade da instrução `BREAK`).

```
switch ajuste(METODO)
  case 'linear'
    disp('Ajuste linear')
  case 'exponencial'
    disp('Ajuste exponencial')
  case 'potencial'
    disp('Ajuste potencial')
  otherwise
    disp('Ajuste desconhecido')
end
```

Scripts e Funções

O script (também chamado de arquivo `m` ou *m-file*) é uma coleção de comandos Matlab escrito em arquivo texto cujo nome termina com a extensão “.m”. Digitando-se o nome do arquivo `m` sem a extensão na área de trabalho do Matlab faz com que os comandos descritos nele sejam executados seqüencialmente no Matlab. Para isso, é necessário que o arquivo `m` esteja em um subdiretório listado no caminho (`path`) do ambiente Matlab.

Por exemplo, suponha que o arquivo `plotseno.m` contenha as seguintes linhas:

```
x = 0:2*pi/N:2*pi;
y = sin(h*x);
plot(x,y)
```

Então, a seqüência de comandos na área de trabalho do Matlab:

```
>> N = 100; h = 5;  
>> plotseno
```

gerará a Figura 1.

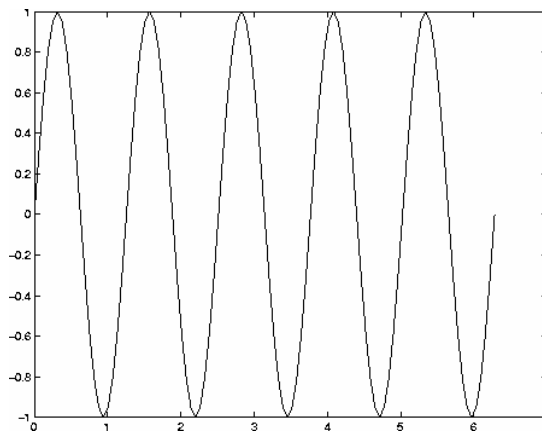


Fig. 1 – Resultado da execução do arquivo m `plotseno`

Como este exemplo mostrou, os comandos no script pode se referir a variáveis já definidas na área de trabalho do Matlab (as variáveis `N` e `h` chamadas em `plotseno.m`).

Além dos scripts, as Funções são um tipo de arquivo m que permitem ao usuário criar novos comandos do Matlab. Uma função é definida como um arquivo m que começa com um cabeçalho na seguinte forma:

```
function [saida1, saida2,...] = nome_funcao(var1, var2,...)
```

O restante do arquivo m consiste de comandos do Matlab que utilizam os valores das variáveis `var1`, `var2`, etc., para calcular o valor das variáveis `saida1`, `saida2`, etc. É importante observar que quando uma função é chamada, o Matlab cria um espaço de trabalho separado das variáveis da área de trabalho. Dessa forma, os comandos na função não podem se referir às variáveis (interativas) da área de trabalho, a menos que elas sejam passadas como argumentos da função (no lugar das variáveis `var1`, `var2`, etc.). Inversamente, as variáveis criadas dentro da função são apagadas quando a execução da função é terminada, exceto se elas forem passadas como argumentos de saída da função (variáveis `saida1`, `saida2`, etc.).

Um exemplo de uma função que calcula $f(x) = \sin(x^2)$. Os comandos são gravados no arquivo `fcn.m` (o nome da função no Matlab é o nome do arquivo m sem extensão).

```
function y = fcn(x)  
y = sin(x.^2);
```

(Observe o operador escalar `.` de modo que a função `fcn` retorna um vetor de mesma dimensão da variável `x`). Com esta função definida, podemos usar `fcn` como se fosse uma função intrínseca do Matlab:

```
>> x = (-pi:2*pi/100:pi)';  
>> y = sin(x);  
>> z = fcn(x);  
>> plot(x,y,x,z)  
>> grid
```

O gráfico mostrado na Fig. 2 é o resultado do cálculo das função intrínseca do Matlab `sin(x)` e da função personalizada `fcn(x)`. Observe o uso do comando `grid` para desenhar uma grade pontilhada no gráfico.

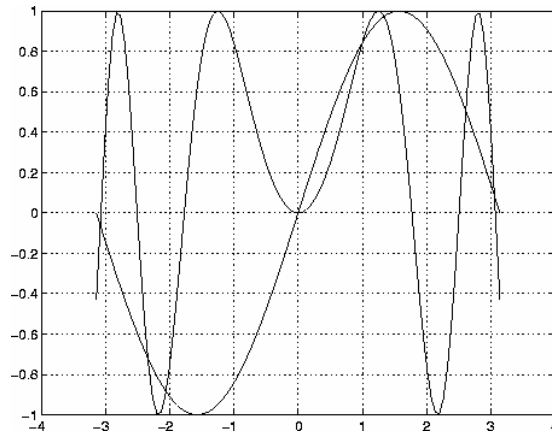


Fig. 2 – Gráfico contendo duas curvas

Um Exemplo de Aplicação de Função

Pode-se observar na Fig. 2 que a função $f(x) = \sin(x^2)$ possui uma raiz entre 1 e 2 (na realidade, a raiz exata é $x = \sqrt{\pi}$, mas vamos ignorar esta informação neste momento). Um algoritmo que poderia ser utilizado para o cálculo da raiz dessa função não-linear seria o método da bisseção, que avalia a função e estabelece um intervalo no qual a função muda de sinal e, repetidamente, divide pela metade o intervalo até que a raiz esteja contida num intervalo tão pequeno quanto se desejar.

Uma função implementando o método da bisseção ilustra algumas das principais técnicas de programação no Matlab. A primeira técnica importante, sem a qual uma rotina de bisseção não funcionaria, é a capacidade de passar o nome de uma função a outra função. Neste caso, a função `bissecao` (observar o nome da função sem acento til e cedilha) precisa saber o nome da função cuja raiz deseja-se calcular. Este nome é passado como uma variável texto e a função interna `feval` do Matlab é usada para se avaliar numericamente a função cujo nome seja fornecido como argumento da função `feval`. Assim, interativamente,

```
>> fcn(2)
ans =
    -0.7568
```

e

```
>> feval('fcn',2)
ans =
    -0.7568
```

são equivalentes (observe que são usados apóstrofes para delimitar uma variável texto). Uma variável também pode ser usada para atribuir o nome da função:

```
>> str = 'fcn'
str =
f
>> feval(str,2)
ans =
    -0.7568
```

Para mais informações sobre como o Matlab manipula variáveis texto, digite `help strings`.

O seguinte programa Matlab usa algumas facilidades de manipulação de texto do Matlab para passar o nome da função para `bissecao`. O sinal `%` indica que o restante da linha é comentário (e, portanto, ignorado na execução do programa).

```

function c = bissecao(fn,a,b,tol)

% c = bissecao('fn',a,b,tol)
%
% Esta funcao calcula a raiz da funcao fn no intervalo [a,b]
% com tolerancia tol. Admite-se que a funcao tenha uma única raiz
% no intervalo [a,b].

% Avalia a funcao nos extremos e verifica se o sinal muda
fa = feval(fn,a);
fb = feval(fn,b);

if fa*fb >= 0
    error('A funcao deve ter sinais opostos em a e b')
end

% A variavel OK e usada para sinalizar que o valor da raiz foi obtido
% antes de comecar o metodo.
OK = 0;

% Divide o intervalo pela metade
c = (a+b)/2;

% Laco principal
while abs(a-b) > 2*tol & ~OK % Observe o caractere de negacao ~
    % Avalia a funcao no valor medio
    fc = feval(fn,c);
    if fa*fc < 0 % A raiz esta a esquerda de c
        b = c;
        fb = fc;
        c = (a+b)/2;
    elseif fc*fb < 0 % A raiz esta a direita de c
        a = c;
        fa = fc;
        c = (a+b)/2;
    else % Chegamos na raiz
        OK = 1;
    end
end
end

```

Considerando que o nome do arquivo m é bissecao.m, ele pode ser executado como:

```

>> x = bissecao('fcn',1,2,1e-6)
x =
    1.7725
>> erro = sqrt(pi)-x
erro =
   -4.1087e-07

```

Podemos criar não somente novos comandos no Matlab usando arquivos m, mas também um sistema de ajuda (*help*) automático, colocando as linhas de comentário (%) no cabeçalho do arquivo m. O comando `help` imprimirá as linhas de comentário do cabeçalho do arquivo m:

```

>> help bissecao

% c = bissecao('fn',a,b,tol)
%
% Esta funcao calcula a raiz da funcao fn no intervalo [a,b]
% com tolerancia tol. Admite-se que a funcao tenha uma única raiz
% no intervalo [a,b].

```

Observe que o nome da função 'fn' foi colocada entre apóstrofos no argumento da função bissecao.m. Na verdade, a variável fn já é uma variável texto, não necessitando estar delimitada por apóstrofos. O uso dos apóstrofos é um lembrete para que a variável seja passada como variável texto no ambiente de trabalho do Matlab. Observe também o uso da função error no início do programa. Esta função intrínseca do Matlab exibe o texto passado como seu argumento e encerra a execução do programa.

Manipulação de Arquivos

A manipulação de arquivos na área de trabalho do Matlab normalmente é realizada pelos comandos save e load. Em programas, o Matlab possui comandos equivalentes aos da linguagem C. (fopen, fclose, fprintf e fscanf).

Para abrir um arquivo de leitura (entrada de dados), utilize o seguinte comando:

```
fid = fopen('arquivo.dat','r')
```

no qual id é uma variável inteira que contém o endereço lógico do arquivo de leitura ('r') 'arquivo.dat'.

Para ler os dados, utiliza-se o comando fscanf. Supondo que o arquivo de dados contenha uma coluna de valores no formato real; eles serão lidos de forma seqüencial pelo comando:

```
x = fscanf(fid, '%f');
```

Terminada a leitura dos dados do arquivo fecha-se o arquivo com o comando:

```
fclose(fid);
```

Para gravar um arquivo de dados, emprega-se a seguinte seqüência de comandos:

```
fid = fopen('arquivo_gravacao.dat','w');
fprintf(fid, '%f', x);
```

Observe que o status do arquivo_gravacao.dat é de escrita (gravação) ('w' - write).

Ao terminar a leitura dos dados ou quando precisar ler novamente os dados, fecha-se o arquivo com o mesmo comando visto anteriormente:

```
fclose(fid);
```

TABELAS DE MATRIZES E FUNÇÕES ESPECIAIS NO MATLAB

| Matrizes especiais | |
|--------------------|---------------------|
| diag | diagonal |
| hankel | Hankel |
| pascal | Triângulo de Pascal |
| toeplitz | Toeplitz |
| vander | Vandermonde |

| Matrizes utilitárias | |
|----------------------|-------------------------------------|
| zeros | Matriz com zeros |
| ones | Matriz constante |
| rand | Matriz aleatória |
| eye | Matriz identidade |
| linspace | Vetores linearmente espaçados |
| logspace | Vetores com espaçamento logaritmico |

OPERADORES E FUNÇÕES BÁSICAS

| Operadores relacionais | |
|------------------------|------------------|
| < | menor que |
| <= | menor ou igual a |
| > | maior que |
| >= | maior ou igual a |
| == | igual |
| ~= | não igual a |

| Operadores lógicos | |
|--------------------|-----|
| & | and |
| | or |
| ~ | not |

| Funções relacionais e lógicas | |
|-------------------------------|--|
| any | condições lógicas |
| all | condições lógicas |
| find | acha índices de tabelas de valores lógicos |
| exist | checa a existência de variáveis |
| isnan | detecta NaN (Not a Number) |
| finite | detecta quantidades infinitas |
| isstr | detecta variáveis alfanuméricas |
| strcmp | compara variáveis alfanuméricas |

| Funções matemáticas elementares | |
|---------------------------------|-------------------------|
| abs | valor absoluto ou norma |
| angle | ângulo de fase |
| sqrt | raiz quadrada |
| real | parte real |
| imag | parte imaginária |
| conj | conjugado complexo |
| round | valor arredondado |
| sign | função sinal |
| isempty | detecta matrizes vazias |
| exp | função exponencial |
| log | logaritmo natural |
| log10 | logaritmo de base 10 |

| Funções trigonométricas | |
|-------------------------|---------------------------|
| sin | seno |
| cos | cosseno |
| tan | tangente |
| asin | arco-seno |
| acos | arco-cosseno |
| atan | arco-tangente |
| sinh | seno hiperbólico |
| cosh | cosseno hiperbólico |
| tanh | tangente hiperbólica |
| asinh | arco-seno hiperbólico |
| acosh | arco-cosseno hiperbólico |
| atanh | arco-tangente hiperbólico |

| Funções especiais | |
|-------------------|------------------------------------|
| bessel | Função de Bessel |
| gamma | Funções Gama completa e incompleta |
| rat | Aproximação racional |
| erf | Função erro |
| inverf | Função erro inversa |
| ellipk | Integral elíptica de primeiro tipo |
| ellipj | Função elíptica jacobiana |
| laguer | Função de Laguerre |