Capítulo 9
Operações lógicas e relacionais

Considerações iniciais

- Operações lógicas e relacionais retornam *Verdadeiro* ou *Falso*;
- Dados de entrada de expressões lógicas e relacionais:
  - Verdadeiro: qualquer valor diferente de zero;
  - Falso: zero.
- Saída:
  - Verdadeiro: 1;
  - Falso: 0.
- Quando a saída da operação ocorre em um vetor (neste caso composto de zeros e uns) este vetor é chamado de *vetor lógico*.

Operações relacionais

- Os operadores relacionais são os usuais:
  - $<$ menor que;
  - $<=$ menor ou igual a;
  - $>$ maior que;
  - $>=$ maior ou igual a;
  - $==$ igual a;
  - $\sim=$ diferente de;
- Podem ser usados para comparar dois vetores de mesmo tamanho, ou um escalar com um vetor (o escalar é comparado a todos os elementos do vetor);
- O resultado da operação é retornado em um vetor, onde a i-ésima posição do vetor da saída é o resultado da comparação das i-ésimas posições dos vetores de entrada (ou do escalar com esta, no caso de escalar$\times$ vetor).

Exemplos de operações relacionais

```
>> A = 1:9
A =
  1    2    3    4    5    6    7    8    9

>> B = fliplr(A)
B =
  9    8    7    6    5    4    3    2    1

>> T = (A == B)
T =
  0    0    0    0    1    0    0    0    0

>> T = (A >= B)
T =
  0    0    0    0    1    1    1    1    1

>> T = (B < 4)
T =
  0    0    0    0    0    0    1    1    1
```

Comutatividade da adição

- É preciso cuidado com o problema da comutatividade da adição no caso de teste de igualdade:

```
>> soma1 = -0.08 +0.5 -0.42
soma1 =
     0

>> soma2 = 0.5 -0.42 -0.08
soma2 =
   1.3878e-17

>> T = soma1 == soma2
T =
     0

>> T = (soma1 == soma2) < eps
T =
     1
```

Usando expressões relacionais

- Podemos combinar expressões matemáticas e relacionais:

```
>> disp(A), disp(B)
  1    2    3    4    5    6    7    8    9
  9    8    7    6    5    4    3    2    1

>> A = B - 2*(A>5)
A =
  9    8    7    6    5    2    1    0   -1

>> A = A + (A==0)*eps
A =
  Columns 1 through 5
  9.0000   8.0000   7.0000   6.000    5.0000
  Columns 6 through 9
  2.0000   1.0000   0.0000  -1.0000

>> disp(A(8)==eps)
     1
```

Divisão por zero

- O exemplo anterior $A = A + (A == 0) * eps$ é uma forma de substituir elementos iguais a zero de um vetor pelo número `eps`, o que pode ser útil para evitar divisões por zero.

```
>> X = (-1:1)/2
X =
   -0.5000         0    0.5000

>> sin(X)./X
Warning: Divide by zero.
ans =
    0.9589       NaN    0.9589

>> X = X +(X==0)*eps
X =
   -0.5000    0.0000    0.5000

>> disp(sin(X)./X)
    0.9589    1.0000    0.9589
```

Divisão por zero II

- O enfoque anterior evita divisões por zero, mas não é a única maneira de fazer isso. Selecionar componentes para se fazer cálculo (ou para evitar fazê-lo) é um enfoque muito utlizado no **MATLAB**.

```
>> X = (-2:2)/2
X =
  -1.0000  -0.5000        0   0.5000   1.0000

>> Y = ones(size(X))
Y =
     1     1     1     1     1

>> L = X~=0 % vetor lógico para indexar  X
L =
     1     1     0     1     1

>> Y(L) = sin(X(L))./X(L)
Y =
   0.8415   0.9589   1.0000   0.9589   0.8415
```

Uma função útil: *logical*

```
>> lookfor logical

... LOGICAL Convert numeric values to logical ...
```

- Compare o exemplo abaixo com o seguinte a ele:

```
>> c=[1 0 1]
c =
     1     0     1
>> d=logical(c)
d =
     1     0     1
>> d==c % comparação é legal
ans =
     1     1     1
>> disp(class(c)), disp(class(d))
double
logical
```

Uma função útil: *logical*

```
>> x=-3:3
x =
    -3    -2    -1     0     1     2     3

>> abs(x)>1
ans =
     1     1     0     0     0     1     1

>> y=x(abs(x)>1)
y =
    -3    -2     2     3

>> y=x([1 1 0 0 0 1 1]) % operação não é legal
??? Subscript indices must either be
    real positive integers or logicals.

>> y=x(logical([1 1 0 0 0 1 1]))
y =
    -3    -2     2     3
```

Operadores lógicos

- Os operadores lógicos são os usuais:
  - $\&$ AND;
  - $|$ OR;
  - $\sim$ NOT.
- A precedência destes operadores também é a usual.

```
>> A = 1:5; B = fliplr(A);

>> disp(~A>3) % atenção para a preced.
     0     0     0     0     0

>> disp(~(A>3))
     1     1     1     0     0

>> disp((A>2) & (A<4))
     0     0     1     0     0
```

Um exemplo de gráfico

- Digite os seguintes comandos no **MATLAB** (sem os comentários):

```
>> X = linspace(0,10,100); % gera dados.
>> Y = sin(X);
>> Z = (Y>=0).*Y; % zera elem. neg. do seno.
>> Z = Z+0.5*(Y<0); % soma 1/2 aos elem. zerados.
>> Z = (X<=8).*Z; % zera elem. maiores que 8.
>> plot(X,Z)
```

- Agora um por vez, digite os comandos a seguir e observe as modificações no gráfico.

```
>> xlabel('X')
>> ylabel('Z = f(X)')
>> title('Figura: Um sinal descontínuo')
```

Valores não numéricos

- Quando temos resultados que são indefinidos o **MATLAB** retorna NaN, indicando que o resultado é um valor não numérico.
- NaNs requerem um tratamento especial e deve-se ter cuidado especialmente em expressões lógicas e relacionais.

```
>> A = [1 2 NaN inf NaN];

>> disp(2*A) % operaçoes com NaNs resultam em NaNs
     2     4   NaN   Inf   NaN

>> disp(A==NaN) % Dois NaN's são diferentes
     0     0     0     0     0
>> disp(A~=NaN)
     1     1     1     1     1

>> disp(isnan(A)) % usar a função para testar
     0     0     1     0     1
```

Matrizes vazias

- São variáveis do **MATLAB** com tamanho zero em uma ou mais dimensões.

```
>> size([])
ans =
     0     0

>> zeros(0,5) % 0 linhas e 5 colunas
ans =
   Empty matrix: 0-by-5

>> disp(length(ans)) % tamanho=0 apesar de 4 cols
     0

>> disp(size(ones(4,0))) % 4 linhas e 0 colunas
     4     0
```

A função $find$

- Algumas funções retornam matrizes vazias quando nenhum outro resultado é apropriado. A função $find$ é uma delas.
- $find$(expressão): Retorna os índices do vetor para os quais uma expressão relacional é verdadeira.

```
>> X = -3:3
ans =
     -3   -2   -1    0    1    2    3
>> V = find(abs(X)> 1)
V =
      1    2    6    7
>> Y = X(abs(X)>1) % endereçamento lógico
Y =
     -3   -2    2    3
```

A função $find$ com matrizes

```
>> A=reshape(1:9,3,3);

>> [i,j] = find(A>5)
i =
     3
     1
     2
     3
j =
     2
     3
     3
     3

>> I = find(A>5)
I =
   6
   7
   8
   9
```

```
>> A(I) = 0
A =
     1   4   0
     2   5   0
     3   0   0

>> A=reshape(1:9,3,3);
>> disp(A(i,j))
     6   9   9   9
     4   7   7   7
     5   8   8   8
     6   9   9   9
%Note: diag(A(i,j))=A(I)
% A(i,j)=A([3 1 2 3],
%           [2 3 3 3])

>> A(i,j) = 0;
A =
     1   0   0
     2   0   0
     3   0   0
```

Exemplo: matriz vazia e $find$

- Recapitulando: o **MATLAB** retorna uma matriz vazia quando não possui um resultado mais apropriado como no exemplo a seguir.

```
>> X = -2:2
X =
    -2    -1    0     1     2

>> Y = find(X>2)
Y =
     []

>>disp(isempty(Y))
     1
```

Algumas considerações finais

- Evite comparar `X == []` para testar se uma matriz é vazia porque isto pode gerar resultados incorretos devido às dimensões variáveis de uma matriz vazia. Dê preferência ao uso da função $isempty$.
- Na hora de escrever expressões, não esquecer de considerar a precedência dos operadores. A seção 9.3 do livro possui uma tabela completa.
- O **MATLAB** possui diversas funções que retornam valores lógicos, algumas delas nós vimos em exemplos precedentes (tais como $isempty$ acima). A seção 9.4 do livro traz uma tabela completa com a descrição destas funções.