Relational and Logical Operators

Lecture: 7 (Dr. Ajeet Kumar)

Table of Contents

Relational Operators

Relational Operators are used to compare the scalers, vectors, Matrices and Strings.

Output of Relation Operator is always a logical 0 and 1.

"1" corresponds to **TRUE** condition and **"0"** corresponds to **FALSE** condition.

Operation	Mathematical Symbol	Matlab Operator
Equal to	=	==
Not equal to	≠	~=
Greater than	>	>
Greater than or equal to	≥	>=
Less than	<	<
Less than or equal to	≤	<=

Scaler to Scaler

Output is always logical 0 or 1 depending on the applied condition is flase or true.

Example:

```
% When both operands are Scaler: Scaler to Scaler
```

```
x=5,y=10,
```

```
x = 5
y = 10
```

```
z1=(x==y), z2=(x<y),z3=(x>y),z4=(x<=y),z5=(x>=y)
```

```
z1 = logical
   0
z2 = logical
   1
z3 = logical
   0
z4 = logical
   1
z5 = logical
   0
```

Scaler to Vector/Matrix

Each element of Vector/Matrix is compared with the Scaler. As a result you get a Vector/Matrix of logical 0 and 1.

```
% When one operand is scaler and other one is vector or Matrix: Scaler to Vector/Matrix
x=5, u=[ 1 2 3 5 10 12]    % x is scaler and u is row vector
```

```
x = 5
u = 1×6
     1     2     3     5    10    12
```

```
z1=(x>=u), z2=(x~=u)
```

```
z1 = 1×6 logical array
   1   1   1   1   0   0
z2 = 1×6 logical array
   1   1   1   0   1   1
```

```
x=10, A=[ 11 10 13; 15 4 6; 4 18 20 ]
```

```
x = 10
A = 3×3
    11    10    13
    15     4     6
     4    18    20
```

```
z2=(x==A),z3=(x>A)
```

```
z2 = 3×3 logical array
   0   1   0
   0   0   0
   0   0   0
z3 = 3×3 logical array
   0   0   0
   0   1   1
```

```
    1   0   0
```

Row Vector to Row Vector

Condition: Dimension of both row vectors should be same.

As a result you get the a row vector having logical 0 and 1 elements

```
%   When both operands are row vector: Row Vector to Row Vector
u_1=[15 17 13 18], u_2=[ 10 14 16 19]
```

```
u_1 = 1×4
    15    17    13    18
u_2 = 1×4
    10    14    16    19
```

```
z1=(u_1>u_2), z2=(u_1~=u_2)
```

```
z1 = 1×4 logical array
   1   1   0   0
z2 = 1×4 logical array
   1   1   1   1
```

Column Vector to Column Vector

Condition: Dimension of both column vectors should be same

As a result you get the a column vector having logical 0 and 1 elements

```
% When both operands are column vector: Column Vector to Column Vector
v_1= [1;2;3; 16],v_2=[10; 1; 17; 23]
```

```
v_1 = 4×1
     1
     2
     3
    16
v_2 = 4×1
    10
     1
    17
    23
```

```
z1=(v_1<=v_2), z2=(v_1==v_2)
```

```
z1 = 4×1 logical array
   1
   0
   1
   1
z2 = 4×1 logical array
   0
   0
```

```
     0
     0
```

Row Vector to Column Vector

When a row vector **"u"** of dimension 1-by-n is compared with a column vector **"v"** of dimension m-by-1, each element of the row vector are compared with each element of column vector. As a result you get a logical Matrix of dimension mxn.

```
% When one operand is row vector and other is column vector: Row Vector to Column Vector
u=[1 4 5 6], v=[2; 5; 7]
```

```
u = 1×4
     1     4     5     6
v = 3×1
     2
     5
     7
```

```
z1=(u<=v), z2=(u>v)
```

```
z1 = 3×4 logical array
   1   0   0   0
   1   1   1   0
   1   1   1   1
z2 = 3×4 logical array
   0   1   1   1
   0   0   0   1
   0   0   0   0
```

Column Vector to Row Vector

When a column vector **"v"** of dimension p-by-1 is compared with a row vector **"u"** of dimension 1-by-q, each element of the row vector is compared with each element of column vector. As a result you get a logical Matrix of dimension p-by-q.

```
% When one operand is column vector and other is row vector: Column Vector to Row Vector
v=[23; 34; 46; 11], u=[22 45 57]
```

```
v = 4×1
    23
    34
    46
    11
u = 1×3
    22    45    57
```

```
z1=(v>=u), z2=(v~=u)
```

```
z1 = 4×3 logical array
   1   0   0
   1   0   0
   1   1   0
   0   0   0
```

```
z2 = 4×3 logical array
   1   1   1
   1   1   1
   1   1   1
   1   1   1
```

Row Vector to Matrix

Condition: When you are comparing a row vector **"u"** of dimension 1-by-n with a matrix **"A"** of dimension m-by-n, number of columns should be same.

Comparison is made column wise.

As a result you get a logical matrix of m-by-n.

```
% When one operand is row vector and other is matrix: Row Vector to Matrix
u=[11 17 20 23], A=[11 14 12 10; 14 16 34 23]
```

```
u = 1×4
    11    17    20    23
A = 2×4
    11    14    12    10
    14    16    34    23
```

```
z1=(u<A), z2=(u>=A)
```

```
z1 = 2×4 logical array
   0   0   0   0
   1   0   1   0
z2 = 2×4 logical array
   1   1   1   1
   0   1   0   1
```

Column Vector to Matrix

Condition: When you are comparing a column vector **"v"** of dimension m-by-1 with a matrix **"A"** of dimension m-by-n, number of rows should be same.

Comparison is made row wise.

As a result you get a logical matrix of m-by-n.

```
% When one operand is column vector and other is matrix: Column Vector to Matrix
v=[22; 34; 11;],A=[11 35; 25 22; 45 10]
```

```
v = 3×1
    22
    34
    11
A = 3×2
    11    35
    25    22
    45    10
```

```
z1=(v>=A),z2=(v<A)
```

```
z1 = 3×2 logical array
   1   0
   1   1
   0   1
z2 = 3×2 logical array
   0   1
   0   0
   1   0
```

Matrix to Matrix

Condition: dimension of both matrices should be same.

Comparison is made element-by-element wise.

As a result you get a logical matrix of the dimension of given matrix.

```
% When both operands are matrix: Matrix to Matrix
A=[21 2 3; 4 7 9; 11 3 45], B=[11 34 23; 34 67 11; 23 34 67]
```

```
A = 3×3
    21     2     3
     4     7     9
    11     3    45
B = 3×3
    11    34    23
    34    67    11
    23    34    67
```

```
z1=(A>=B),z2=(A==B)
```

```
z1 = 3×3 logical array
   1   0   0
   0   0   0
   0   0   0
z2 = 3×3 logical array
   0   0   0
   0   0   0
   0   0   0
```

Important Points

- Be careful while using "==" operator. There is clear difference between "=" and "==".
- = refer to assign the value in new variable
- == refer to compare the operands
- '==' operator gives logical output 1 when both the operands are **exactly** same. So one should be very careful while using operator.
- In case of Complex number, the operators >, <, >=, and <= use only the real part of the operands in performing comparisons, while operators == and ~= test both real and imaginary parts of the operands.

```
% Important points to remeber using relational operators (Decision-Making program)
x1=4.0000,x2=4.000010   % while equating two variables sometimes it is better to
```

```
x1 = 4
x2 = 4.0000
```

```
% check abs(x1-x2)<= accuracy limit rather than x1==x2 (Wein's law varification)
z1=(x1==x2),z2=abs(x2-x1)<=10^-3    % See the difference in output
```

```
z1 = logical
   0
z2 = logical
   1
```

```
x=sin(2*pi),x==0
```

```
x = -2.4493e-16
ans = logical
   0
```

```
% compare the complex no.
z1=2+3i;z2=2+3j;
z3=10+10i,z4=10+50i,
```

```
z3 = 10.0000 + 10.0000i
z4 = 10.0000 + 50.0000i
```

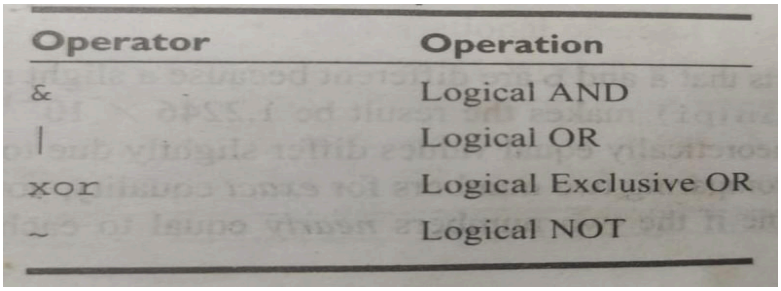
```
z1==z2     % check both real and imag part
```

```
ans = logical
   1
```

```
z1>z2      % only check real part
```

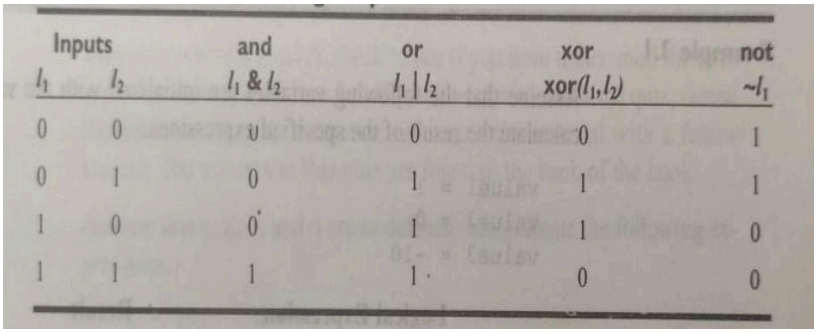
```
ans = logical
   0
```

Logical Operators



Operator	Operation
&	Logical AND
|	Logical OR
xor	Logical Exclusive OR
~	Logical NOT

Truth Table of Logical Operators



Inputs		and	or	xor	not
I_1	I_2	I_1 & I_2	$I_1 \mid I_2$	xor(I_1,I_2)	~I_1
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Refer input 1 = TRUE, 0 = False

Logical Operators are also known as Boolean Operators.

For purpose of logical operations, Matlab treats an operand as true (logical 1), if it is non-zero value, and false if it is zero (logical 0).

Similar rules of operation are applied on operands as in case of relational operators

```
x=[0 5 3 7],y=[0 -2 8 7],
```

```
x = 1×4
     0     5     3     7
y = 1×4
     0    -2     8     7
```

```
m=(x>y)&(x>4)
```

```
m = 1×4 logical array
   0   1   0   0
```

```
n=x|y
```

```
n = 1×4 logical array
   0   1   1   1
```

```
z=~(x|y)
```

```
z = 1×4 logical array
   1   0   0   0
```

```
p=xor(x,y)
```

```
p = 1×4 logical array
   0   0   0   0
```

```
% find the elements of the array that satisfies special condition
x=[1 2 5 7 90 34 10 11 15 19 20]
```

```
x = 1×11
     1     2     5     7    90    34    10    11    15    19    20
```

```
% find out all the elements of vector x having values more than 2 and less than 15
x((x>2)&(x<15))
```

```
ans = 1×4
     5     7    10    11
```

```
A=[1 2 3; 23 45 12; 21 4 14],
```

```
A = 3×3
     1     2     3
    23    45    12
    21     4    14
ans = 2×1
    12
    14
```

```
% find out all the elements of matrix A having value more than 4 and less than or equal to 14
A((A>4)&(A<=14))
```