Relational and Logical Operations

These operations and functions provide answers to **True-False** questions.

One important use of this capability is to control the flow or order of execution of a series of MATLAB commands (usually in an M-file) based on the results of true/false questions.

As inputs to all relational and logical expressions, MATLAB considers any nonzero number to be true, and zero to be False. The output of all relational and logical expressions produces ***one for True*** and ***zero for False***, and the array is flagged as *logical*. That is, the result contains numerical values 1 and 0 that can be used in mathematical statement, but also allow logical array addressing.

Relational Operations

Operation	Description
= =	Equal
~ =	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

Example:

```
>> a=1:9; b=9-a;
>> t=a>4 %finds elements of (a) that
are greater than 4.
t =
0 0 0 0 1 1 1 1 1
```

Zeros appear where a ≤ 4, and ones where a > 4.

`>> t= (a==b) %`finds elements of (a) that are equal to those in (b).

```
t =
0 0 0 0 0 0 0 0 0
```

Logical Operation

Operation	Description
&	Logical AND **and(a,b)**
|	Logical OR **or(a,b)**
~	Logical NOT
xor (a,b)	Logical EXCLUSIVE OR

Example:

```
>> a = [0 4 0 -3 -5 2];
>> b = ~a
b =
1 0 1 0 0 0
>> c=a&b
c =
0 0 0 0 0 0
```

Example: let `x=[ 2 -3 5 ;0 11 0],` then

a) find elements in x that are greater than 2

solution

```
a)
>> x>2
ans =
0 0 1
0 1 0
```

Logical Functions

MATLAB has a number of useful logical functions that operate on scalars, vectors, and matrices. Examples are given in the following list:-

Function	Description
`any(x)`	True if any element of a vector is a nonzero number or is logical 1 (TRUE)
`all(x)`	True if all elements of a vector are nonzero.
`find(x)`	Find indices of nonzero elements
`isnan(x)`	True for Not-a-Number
`isinf(x)`	True for infinite elements.
`isempty(x)`	True for empty array.

Example: Let A=[4 9 7 0 5]

```
>> any(A)
ans = 1
>> all(A)
ans = 0
>> find(A)
ans = 1 2 3 5
```

To remove zero elements from matrix

```
>> B=A(find(A));
>> B
B = 4 9 7 5
```

Precedence	Operation
1 (highest)	Parentheses (if nested parentheses exist, inner ones have precedence)
2	Exponentiation
3	Logical NOT (~)
4	Multiplication, division
5	Addition, subtraction
6	Relational operators (>, <, >=, <=, ==, ~=)
7	Logical AND (&)
8 (lowest)	Logical OR (|)

```
>> 3&7
```
3 AND 7.

```
ans =
     1
```
3 and 7 are both true (nonzero), so the outcome is 1.

```
>> a=5|0
```
5 OR 0 (assign to variable a).

```
a =
     1
```
1 is assigned to a since at least one number is true (nonzero).

```
>> ~25
```
NOT 25.

```
ans =
     0
```
The outcome is 0 since 25 is true (nonzero) and the opposite is false.

```
>> t=25*((12&0)+(~0)+(0|5))
```
Using logical operators in a math expression.

```
t =
    50
```

```
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
```
Define two vectors x and y.

```
>> x&y
ans =
     1     0     0     1     0     1
```
The outcome is a vector with 1 in every position where both x and y are true (nonzero elements), and 0s otherwise.

```
>> z=x|y
z =
     1     1     1     1     0     1
```
The outcome is a vector with 1 in every position where either or both x and y are true (nonzero elements), and 0s otherwise.

```
>> ~(x+y)
ans =
     0     0     0     1     1     0
```
The outcome is a vector with 0 in every position where the vector x + y is true (nonzero elements), and 1 in every position where x + y is false (zero elements).

```
>> 5>8
```
Checks if 5 is larger than 8.

```
ans =
     0
```
Since the comparison is false (5 is not larger than 8) the answer is 0.

```
>> a=5<10
```
Checks if 5 is smaller than 10, and assigns the answer to a.

```
a =
     1
```
Since the comparison is true (5 is smaller than 10) the number 1 is assigned to a.

```
>> y=(6<10)+(7>8)+(5*3==60/4)
```
Using relational operators in math expression.

Equal to 1 since 6 is smaller than 10.

Equal to 0 since 7 is not larger than 8.

Equal to 1 since 5*3 is equal to 60/4.

```
y =
     2
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
```
Define vectors b and c.

```
>> d=c>=b
```
Checks which c elements are larger than or equal to b elements.

```
d =
     0     1     1     0     1     1     0
```
Assigns 1 where an element of c is larger than or equal to an element of b.

```
>> b == c
```
Checks which b elements are equal to c elements.

```
ans =
     0     0     1     0     0     1     0
>> b~=c
```
Checks which b elements are not equal to c elements.

```
ans =
     1     1     0     1     1     0     1
>> f=b-c>0
```
Subtracts c from b and then checks which elements are larger than zero.

```
f =
     1     0     0     1     0     0     1
>> A=[2 9 4; -3 5 2; 6 7 -1]
```
Define a 3 × 3 matrix A.

```
A =
     2     9     4
    -3     5     2
     6     7    -1
>> B=A<=2
```
Checks which elements in A are smaller than or equal to 2. Assigns the results to matrix B.

```
B =
     1     0     0
     1     0     1
     0     0     1
```

```
>> r = [8 12 9 4 23 19 10]
```

Define a vector r.

```
r =
     8    12     9     4    23    19    10
```

```
>> s=r<=10
```

Checks which r elements are smaller than or equal to 10.

```
s =
     1     0     1     1     0     0     1
```

A logical vector s with 1s at positions where elements of r are smaller than or equal to 10.

```
>> t=r(s)
```

Use s for addresses in vector r to create vector t.

```
t =
     8     9     4    10
```

Vector t consists of elements of r in positions where s has 1s.

```
>> w=r(r<=10)
```

The same procedure can be done in one step.

```
w =
     8     9     4    10
```

```
>> 3+4<16/2

ans =

     1

>> 3+(4<16)/2

ans =

    3.5000
```

+ and / are executed first.

The answer is 1 since 7 < 8 is true.

4 < 16 is executed first, and is equal to 1, since it is true.

3.5 is obtained from 3 + 1/2.

```
>> x=-2; y=5;
```

Define variables x and y.

```
>> -5<x<-1
ans =
     0
```

This inequality is correct mathematically. The answer, however, is false since MATLAB executes from left to right. $-5 < x$ is true (=1) and then $1 < -1$ is false (0).

```
>> -5<x & x<-1
ans =
     1
```

The mathematically correct statement is obtained by using the logical operator &. The inequalities are executed first. Since both are true (1), the answer is 1.

```
>> ~(y<7)
ans =
     0
```

$y < 7$ is executed first, it is true (1), and ~1 is 0.

```
>> ~y<7
ans =
     1
```

~y is executed first. y is true (1) (since y is nonzero), ~1 is 0, and $0 < 7$ is true (1).

```
>> ~((y>=8)|(x<-1))
ans =
     0
```

$y >= 8$ (false), and $x < -1$ (true) are executed first. OR is executed next (true). ~ is executed last, and gives false (0).

```
>> ~(y>=8)|(x<-1)
ans =
     1
```

$y >= 8$ (false), and $x < -1$ (true) are executed first. NOT of ($y >= 8$) is executed next (true). OR is executed last, and gives true (1).

Equal (==)

```
a = 5;
b = 5;
result = (a == b);
```

Checks if a is equal to b. Returns true (1) if they are equal.

Not Equal (~=)

```
a = 5;
b = 3;
result = (a ~= b);
```

Checks if a is not equal to b. Returns true if they are different.

Greater Than (>)

```
a = 10;
b = 7;
result = (a > b);
```

Checks if a is greater than b. Returns true if yes.

Less Than (<)

```
a = 3;
b = 8;
result = (a < b);
```

Checks if a is less than b. Returns true if yes.

Greater Than or Equal (>=)

```
a = 4;
b = 4;
result = (a >= b);
```

Checks if a is greater than or equal to b.

Less Than or Equal (<=)

```
a = 2;
b = 5;
result = (a <= b);
```

Checks if a is less than or equal to b.

Logical AND (&)

```
a = true;
b = false;
result = a & b;
```

Returns true only if both a and b are true.

Logical OR (|)

```
a = true;
b = false;
result = a | b;
```

Returns true if either a or b is true.

Logical NOT (~)

```
a = true;
result = ~a;
```

Returns the opposite logical value of a.

MATLAB Programming Basics

Flow Control with Examples

If - Else - Elseif

```
x = 5;

if x > 0
    disp('Positive number')
elseif x < 0
    disp('Negative number')
else
    disp('Zero')
end
```

This structure allows decision making. The program checks if x is positive, negative, or zero, and displays the result.

For Loop

```
for i = 1:5
    disp(['Value: ', num2str(i)])
end
```

The 'for' loop repeats code a specific number of times. Here, it displays numbers from 1 to 5.

While Loop

```
i = 1;
while i <= 5
    disp(['i = ', num2str(i)])
    i = i + 1;
end
```

The 'while' loop continues as long as the condition is true. In this case, it runs until i becomes greater than 5.

Break and Continue

```
for i = 1:10
    if i == 5
        break;
    end
    if mod(i,2) == 0
        continue;
    end
    disp(i)
end
```

'break' stops the loop completely. 'continue' skips to the next iteration. This loop prints odd numbers until it hits 5.

Switch - Case

```
day = 3;

switch day
    case 1
        disp('Sunday')
    case 2
        disp('Monday')
    case 3
        disp('Tuesday')
    otherwise
        disp('Unknown day')
end
```

The 'switch' statement selects one of many code blocks to run, depending on the value of a variable (day).

Example 1: if statement

```
x = 10;
if x > 5
    disp('x is greater than 5');
elseif x == 5
    disp('x is equal to 5');
else
    disp('x is less than 5');
end
```

Example 2: for loop

```
for i = 1:5
    disp(['Iteration number: ', num2str(i)]);
end
```

Example 3: while loop

```
i = 1;
while i <= 5
    disp(['i = ', num2str(i)]);
    i = i + 1;
end
```

Example 4: switch statement

```
day = 'Monday';
switch day
    case 'Monday'
        disp('Start of the week!');
    case 'Friday'
        disp('Weekend is near!');
    otherwise
        disp('Just a regular day.');
end
```

Example 5: if with compound condition

```
a = 5; b = 7;
if a < b && b > 6
    disp('Both conditions are true.');
end
```

Example 6: Sum numbers using for loop

```
sum = 0;
for i = 1:10
    sum = sum + i;
end
disp(['Sum = ', num2str(sum)]);
```

Example 7: while loop with break

```
n = 1;
while true
    disp(n);
    if n == 5
        break;
    end
    n = n + 1;
end
```

Example 8: switch with numeric values

```
x = 2;
switch x
    case 1
        disp('One');
    case 2
        disp('Two');
    otherwise
        disp('Other');
end
```

Exercise 1: Even or Odd Checker

Goal: Write a program to check whether a number is even or odd using **if**

```
n = input('Enter a number: ');
if mod(n, 2) == 0
    disp('The number is even.');
else
    disp('The number is odd.');
end
```

Exercise 2: Display Numbers from 1 to 10

```
for i = 1:10
   disp(i);
end
```

Exercise 3: Grade Evaluation

Goal: Use switch to print a message based on a letter grade

```
grade = input('Enter your grade (A, B, C, D, F): ', 's');
switch grade
   case 'A'
      disp('Excellent!');
   case 'B'
      disp('Good!');
   case 'C'
      disp('Fair');
   case 'D'
      disp('Poor');
   case 'F'
      disp('Fail');
   otherwise
      disp('Invalid grade');
end
```

Exercise 4: Factorial of a Number

Goal: Use a for loop to calculate the factorial of a number

```
n = input('Enter a number: ');
fact = 1;
for i = 1:n
    fact = fact * i;
end
disp(['Factorial = ', num2str(fact)]);
```

Exercise 5: Count Down Using while

Goal: Use `while` to count down from a number to 1.

```
n = input('Enter a number: ');
while n >= 1
   disp(n);
   n = n - 1;
End
```

Exercise 6: Use break inside for loop

Goal: Stop printing numbers once you reach number 5.

```
for i = 1:10
   if i == 5
      break;
   end
   disp(i);
end
```

Exercises

1- if `q=[1 5 6 8 3 2 4 5 9 10 1],x=[ 3 5 7 8 3 1 2 4 11 5 9],`

then:

a) find elements of (q) that are greater than 4.

b) find elements of (q) that are equal to those in (x).

c) find elements of (x) that are less than or equal to 7.

2- If `x=[10 3 ; 9 15], y=[10 0; 9 3], z=[-1 0; -3 2],` what is the output of the following statements:

```
a) v = x > y
b) w = z >= y
c) u = ~z & y
d) t = x & y < z
```

3- let `x=[ 2 -3 5 ;0 11 0]`

find the number of nonzero elements in x

Sol:-

```
>> t=~(~x);
>> sum(sum(t))
ans =
4
```

Any question????

Thanks