MATLAB
Operators, control flow and scripting

Edited by Péter Vass

Operators

An operator is a symbol which is used for specifying some kind of operation to be executed.
An operator is always the member of an expression and refers to at least one but mostly two operands.
An operand can be a variable, a constant or an embedded expression between a pair of round brackets.
An expression is made up of variables, operators
Operators in the MATLAB is designed to work not only on scalars but also on matrices and arrays.

The following groups of operators are distinguished in the MATLAB:

- arithmetic operators,
- relational operators,
- logical operators,
- bitwise operators,
- set operators.

Operators

Most of the arithmetic operators together with matrices and arrays have already been introduced.

Relational operators

They are used for comparing two operands in an expression.
The result of the comparison is a logical value.
There are only two different logical values:
0 which means "false",
1 which corresponds "true"
If the operands are not scalars but vectors or matrices, the comparison is performed element by element.
So, the result of a relational expression will be a vector or matrix of logical values.

Operators

operator	meaning
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to

Example:

```
>>a = 10;
>>b = 20;
>>a >= b
>>a~=b
```

Operators

Logical operators

The results of the logical operations are always logical values (true or false, that is 1 or 0)

There are two types of logical operators in the MATLAB

- element-wise logical operators,
- short-circuit logical operators.

<u>Element-wise logical operators</u>

Their operands are vectors or matrices.
All numbers hold "true" boolean value, except for 0 itself which is "false".
The logical operation is performed element by element.

operator	logical operation
&	AND
|	OR
~	NOT

Operators

Truth table of logical AND

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

Truth table of logical OR

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

Truth table of logical NOT

a	~a
0	1
1	0

Operators

Example

```
>> X=round(10*rand(2,3))
>> Y=eye(2,3)
>> X&Y
>> X|Y
>> ~Y
```

Remark:

the dimensions of the operands must fit.

Try the floor and ceil functions instead of round.

floor(X) rounds down the elements of X to the nearest integers.

ceil(X) rounds up the elements of X to the nearest integers.

round(X) rounds the elements of X to the nearest integers.

Operators

Short-circuit logical operators

The operands of these operators are scalars and/or logical expressions (not vectors an matrices).
The result of an evaluated logical expression is a logical value.
The attribute "short-circuit" is related to the way of evaluation of an expression with logical AND operation (expr1 AND expr2).
When the right side of the expression is "false", the left side of the expression will not be evaluated and the result will be "false" (see the truth table of AND operation).
In such a way, the evaluation of the expression shortens.

operator	logical operation
&&	AND
||	OR

Operators

Example

```
>>x=5;
>>y=0;
>>(y~=0) && (x>2)
```

Both the relational and logical operators are primarily used in conditional expressions of different control statements (if, while etc.)

The application of different control statements are connected to programming in MATLAB language.

Script

It was shown that MATLAB works as a high-performance calculator which is able to execute the commands entered by the user.
However, MATLAB can also be used for programming because it provides a powerful programming language, as well as an interactive computational environment.
In the MATLAB terminology, a source code of a program called script.
So, the activity of writing a program is often mentioned as scripting.
Script is a program file with .m extension and it contains a series of commands.
Unlike a function, it does not accept inputs and does not return any output.
But, it can use the variables existing in the actual workspace in its specified computations.
In addition, the new variables created by the script remain in the workspace, so we can use them in later computations.

Script

Creating a script file

A script file can be created in any text editor, but it is suggested using the built-in MATLAB editor.

The MATLAB editor can be opened in two ways:

- from the command line by typing the command edit or edit *filename*,
- from the tool bar of the MATLAB GUI by choosing New Script or New → Script on the Home tab.

Typing and saving the code of the script

We can use variables, constants, different types of expressions, commands, function calls, control structures etc.

After the % symbol, a line of text as a comment can be inserted in the code. The comment line is not interpreted by the MATLAB

Do not forget to save the script.

Script

Example

```
x = 0:pi/20:2*pi;
y = sin(x);
plot(x,y);
```

Running the script

After typing and saving the code, the script can be executed by means of
typing its name without m extension in the command line and pressing enter key,
or clicking the Run command on the tool bar of the MATLAB editor.

If you modify something in the code, do not forget to save its content before running.

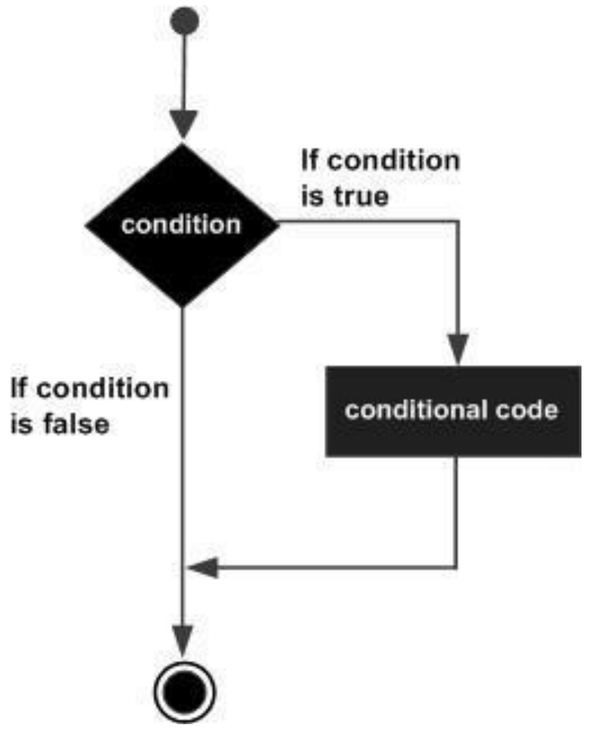
Control statements

Decision making statements

They are used for implementing conditional branches in the program.

Simple conditional execution

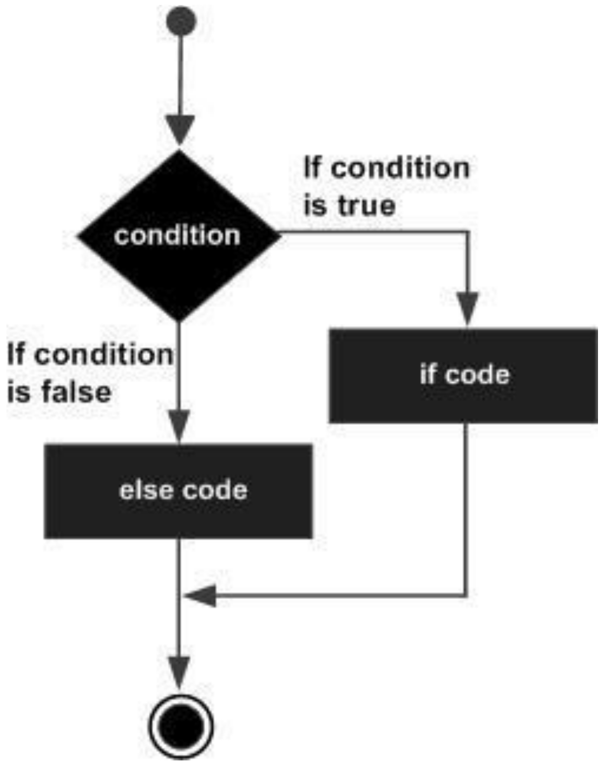
```
if expression
    instruction(s)
end
```



Control statements

<u>Two-way branching</u>

```
if expression
    instruction(s)
else
    instruction(s)
end
```



<u>Multiway branching</u>

```
if expression_1
    instruction(s)
elseif expression_2
    instruction(s)
else
    instruction(s)
end
```

Control statements

Example

```
a = 100;
if a == 10
    fprintf('Value of a is 10\n' );
elseif( a == 20 )
    fprintf('Value of a is 20\n' );
elseif a == 30
    fprintf('Value of a is 30\n' );
else
    fprintf('None of the values are matching\n');
    fprintf('Exact value of a is: %d\n', a );
end
```

Control statements

Nested decison making statements

```
if expression_1
    if expression_2
        instruction(s)
    else
         instruction(s)
    end
end
```

Control statements

Multiway branching with switch statement

```
switch switch_expression
    case case_expression_1
        instruction(s)
    case case_expression_2
        instruction(s)
    …
    …
    otherwise
        instruction(s)
end
```

Control statements

```
mark = 4;
switch(mark)
    case 5
        fprintf('Excellent\n' );
    case 4
        fprintf('Good' );
    case 3
        fprintf('Medium' );
    case 2
        fprintf('Passed' );
    case 1
        fprintf('Failed' );
    otherwise
        fprintf('Invalid mark\n' );
end
```

Control statements

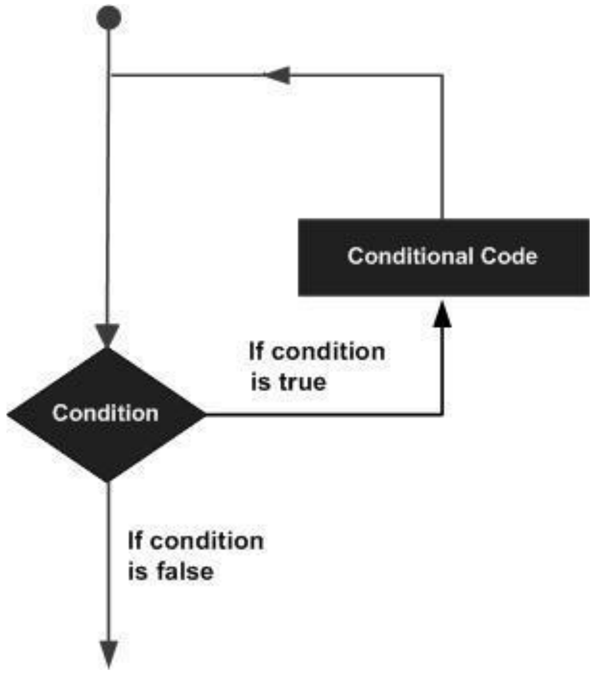
Loop statements

Loops are used to execute a block of code several number of times. The number of repetition is determined by the loop condition.

While loop

The while loop repeatedly executes statements while the loop condition (expression) is true.

```
while expression
    instruction(s)
end
```



Control statements

Example

```
a = 10;
% while loop execution
while( a < 20 )
    fprintf('value of a= %d\n', a);
    a = a + 1;
end
```

Control statements

For loop

The for loop is a control structure by which the number of repetition can be specified directly.

```
for index = values
    instruction(s)
end
```

The *values* part of the statement which specifies the number of repetition is generally given in the following two forms:

initval : *endval*	The value of the loop control variable starts with *initval* , and it is increased by 1 after each repetition of the instruction block. The process continues while its value is less than or equal to *endval.*

Control statements

initval:step:endval

The value of the loop control variable starts with *initval* , and it is increased by the value of *step* after each repetition of the instruction block.

The process continues while its value is less than or equal to *endval*.

If the *initval* is greater than the *endval* and the step is a negative value, the control loop variable is not increased but decreased.

Control statements

Example 1

```
for a = 5:10
    fprintf('value of a: %d\n', a);
end
```

Example 2

```
for a = 5: -0.5: 2
    disp(a)
end
```

Control statements

Nested loops are also used in programming practice

```
while expression_1
    while expression_2
        instruction(s)
    end
end

for index1 = values1
    for index2 = values2
        instruction(s)
    end
end
```

Control statements

Special control statements

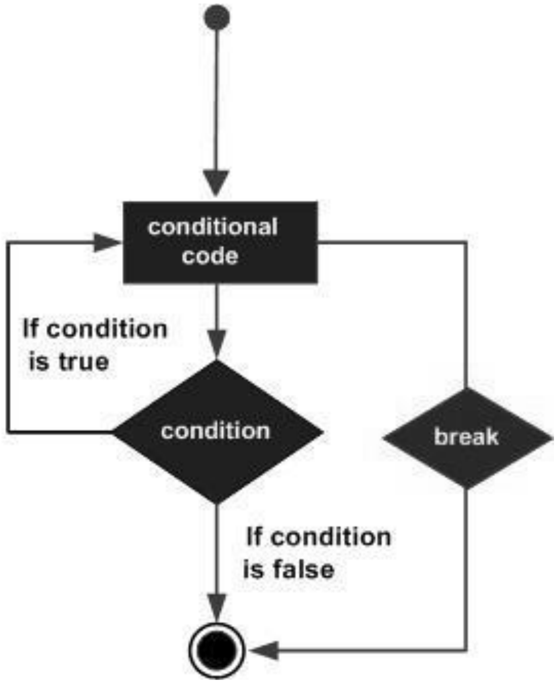
They are used in loops for modifying the normal execution of the loops.

The break statement

It terminates the execution of a loop if a certain condition is fulfilled.

Example

```
a = 10;
while (a < 20 )
    fprintf('value of a: %d\n', a);
    a = a+1;
    if( a > 15)
        break;
    end
end
```



Control statements

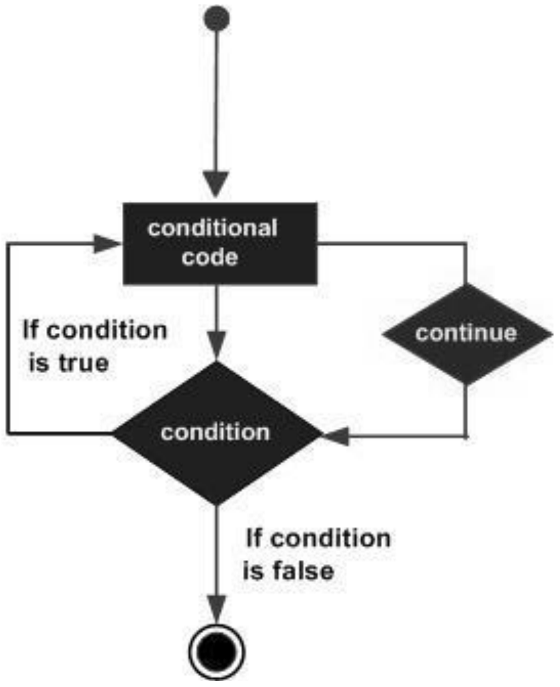
The continue statement

It is used for skipping the actual iteration of a loop if a certain condition is fulfilled.

The execution of the loop continues with the next iteration.

Example

```
a = 10;
while a < 20
    if a == 15
        a = a + 1;
        continue;
    end
    fprintf('value of a: %d\n', a);
    a = a + 1;
end
```


conditional code
If condition is true
continue
condition
If condition is false