Chapter 4 **Loops and Control flow**

1 Introduction

MATLAB is also a programming language. Like other computer programming languages, MATLAB has some decision making structures *for control* of command execution. These decision making or *control flow* structures include *for loops, while loops, and if-else-end* constructions. *Control flow structures* are often used in script M-files and function M-files.

MATLAB provides several tools that can be used to control the *flow* of a program (script or function). In a simple program, the commands are executed one after the other. Here the *flow control* structure that make possible to skip commands or to execute specific group of commands.

2. Loops

2.1 The ‘‘for...end’’ loop

In the ‘*for ... end’ loop*, the execution of a command is repeated at a fixed and predetermined number of times.

The syntax is

for variable = expression

statements

end

Usually, expression is a vector of the form “ *i:s:j* ”. A simple example of for loop is

```
for i=1:5
x=i*i
end
```

It is a good idea to indent the loops for readability, especially when they are nested. Note that MATLAB editor does it automatically.

Multiple for loops can be nested, in which case indentation helps to improve the readability. The following statements form the 5-by-5 symmetric matrix A with (i, j) element i/j for $j \geq i$:

```
n = 5; A = eye(n);
for j=2:n
    for i=1:j-1
        A(i,j)=i/j;
        A(j,i)=i/j;
    end
end
```

2.2 The ‘‘while...end’’ loop

This loop is used when the number of passes is not specified. The looping continues until a stated condition is satisfied. The while loop has the form:

while expression
statements
end

The statements are executed as long as *expression is true*.

Example

x = 1

while x <= 10

*x = 3*x*

end

It is important to note that if the condition inside the looping is not well defined, the looping will continue indefinitely. If this happens, we can stop the execution by pressing *Ctrl-C*.

2.3 Other flow structures

• The *break statement*. A *while loop* can be terminated with the break statement, which passes control to the first statement after the corresponding end. The break statement can also be used to *exit a for loop*.

• The *continue statement* can also be used to *exit* a *for loop* to pass immediately to the *next iteration* of the loop, skipping the remaining statements in the loop.

2.4 Operator precedence

We can build expressions that use any combination of arithmetic, relational, and logical operators. Precedence rules determine the order in which MATLAB evaluates an expression.

The precedence rules for MATLAB are shown in this list (Table 2), ordered from highest (1) to lowest (9) precedence level.

PRECEDENCE	OPERATOR
1	Parentheses ()
2	Transpose (.′), power (.^), matrix power (^)
3	Unary plus (+), unary minus (−), logical negation (∼)
4	Multiplication (.∗), right division (./), left division (.\), matrix multiplication (∗), matrix right division (/), matrix left division (\)
5	Addition (+), subtraction (−)
6	Colon operator (:)
7	Less than (<), less than or equal to (≤), greater (>), greater than or equal to (≥), equal to (==), not equal to (∼=)
8	Element-wise AND, (&)
9	Element-wise OR, (|)

Table. 1: Operator precedence

3. Control flow

MATLAB has four control flow structures: *the if statement, the for loop, the while loop, and the switch statement.*

3.1 The ‘‘if...end’’ structure

Syntax:

• *if … end*

• *if … else … end*

• *if … elseif … else … end*

The simplest form of the *if statement* is

```
if expression
  statements
end
```

Examples

```
1. discr = b*b - 4*a*c;
   if discr < 0
      disp('Warning: discriminant is negative, roots are imaginary');
  end
2. discr = b*b - 4*a*c;
```

```
if discr < 0
    disp('Warning: discriminant is negative, roots are imaginary');
else
    disp('Roots are real, but may be repeated')
end
```

3.
```
discr = b*b - 4*a*c;
if discr < 0
    disp('Warning: discriminant is negative, roots are imaginary');
elseif discr == 0
    disp('Discriminant is zero, roots are repeated')
else
    disp('Roots are real')
end
```

It should be noted that:

• elseif has no space between else and if (one word)

• No semicolon (;) is needed at the end of lines containing *if*, *else*, *end*

• the *end* statement is required

- **Relational and logical operators**

A *relational* operator compares two numbers by determining whether a comparison is *true or false*. Relational operators are shown in Table 2.

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
~=	Not equal to
&	AND operator
|	OR operator
~	NOT operator

Table.2: Relational and logical operators

Note that the "*equal to*" relational operator consists of two equal signs *(==)* (with no space between them), *since* = is reserved for the assignment operator.

3.2. Switch and Case Statement

The *switch statement* executes groups of statements based on the *value of a variable or expression*. The keywords *case* and otherwise delineate the groups. Only the first matching *case* is executed. There must always be an *end* to match the *switch*. The general syntax is as follows:

```
switch variable
    case case_value1
        statements1
    case case_value2
        statements2
        ...
    otherwise
        statements
end
```

Example:

```
n=2
switch(n)
    case 1
        M = eye(n)
    case 2
        M = zeros(n)
    case 3
        M = ones(n)
end
```