Lab-5

1.1 Control Flow and operators

MATLAB is also a programming language. Like other computer programming languages, MATLAB has some decision-making structures for control of command execution. These decisions 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. By creating a File with the extension .m, we can easily write and run programs. We do not need to compile the program since MATLAB is an interpretative (not compiled) language. MATLAB has thousands of functions, and you can add your own using m-Files. MATLAB provides several tools that can be used to control the Flow of a program (script or function). In a simple program as shown in the previous Chapter, the commands are executed one after the other. Here we introduce the Flow control structure that make possible to skip commands or to execute specific group of commands.

1.2 Control Flow

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

1.2.1 The ``if...end'' structure

MATLAB supports the variants of \if" construct.

```
if ... end
if ... else ... end
if ... elseif ... else ... end
```

The simplest form of the if statement is

```
if expression
statements
end
```

Here are some examples based on the familiar quadratic formula.

```
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
- indentation of if block is not required but facilitate the reading.
- the end statement is required

1.2.2 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 5.1.

Table **Error! No text of specified style in document.**-1: Relational and logical operators

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

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

1.2.3 The ``for...end'' loop

In the for ... end loop, the execution of a command is repeated at a Fixed and predeter-

mined 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 ii=1:5
x=ii*ii
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 ≥ 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
```

1.2.4 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.

```
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.

1.2.5 Other Flow structures

- The break statements. 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.
- Other control statements include return, continue, switch, etc. For more detail about these commands, consul MATLAB documentation.

1.2.6 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. We have already seen this in the \Tutorial Lessons".

Here we add other operators in the list. The precedence rules for MATLAB are shown in this list (Table 5.2), ordered from highest (1) to lowest (9) precedence level. Operators are evaluated from left to right.

Table **Error! No text of specified style in document.**-2: Operator precedence

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, (|)

1.2.7 Saving output to a file

In addition to displaying output on the screen, the command fprintf can be used for writing the output to a File. The saved data can subsequently be used by MATLAB or other softwares.

To save the results of some computation to a File in a text format requires the following steps:

1. Open a File using fopen
2. Write the output using fprintf
3. Close the File using fclose

Here is an example (script) of its use.

```
% write some variable length strings to a file
op = fopen('weekdays.txt','wt');
fprintf(op,'Sunday\nMonday\nTuesday\nWednesday\n');
```

```
fprintf(op,'Thursday\nFriday\nSaturday\n');
fclose(op);
```

This File (weekdays.txt) can be opened with any program that can read .txt File.