EXPERIMENT-FOUR

FLOW CONTROL

MATLAB has several flow control constructs:

- if statement
- if- else statement
- if- elseif statement
- switch and case statements.
- for loops
- while loops
- continue statement
- break statement

1- if statement

The if statement evaluates a logical expression and executes a group of statements when the expression is *true*. The if statement form is:

if *condition*

statement

end

or in simplest form we can put it in a single line:

if *condition statement,* end

where *condition* is usually a logical *expression* , i.e. an expression containing a *relational operator,* and which is either *true* or *false.* These *relational operators* are:

<	less than
<=	less than or equal
==	equal
~=	not equal
>	greater than

>= greater than or equal

If *condition* is true, statement executed, but if condition is false, nothing happens. More complicated *logical expressions* can be constructed using the three *logical operators*:

& and
| or
~ not

Example: The quadratic equation:

$$ax^2 + bx + c = 0$$

has equal roots, given by $-b/2a$, provided that $b^2 - 4ac=0$ and a $\neq$ 0. This translates to the following MATLAB statements:

```
a = input('enter the value of a:');
b = input('enter the value of b:');
c = input('enter the value of c:');
if (b^2 -4*a*c == 0) & (a ~= 0)
   disp('The quadratic equation has equal roots')
   x = -b / (2*a)
end
```

After running this program you will see that the command window wants you to enter a value for the variables a and b according to the input function that you put it in the first and second lines of the program. Then the command window will gives the result if the condition is satisfied.

condition may be a vector or matrix, so it is important to understand how relational operators and if statements work with matrices. When you want to check for equality between two variables, you might use

```
if A == B, ...
```

This is legal MATLAB code, and does what you expect when A and B are scalars. But when A and B are matrices, A == B does not test *if* they are equal, it tests *where* they are equal; the result is another matrix of 0's and 1's showing element-by-element equality. In fact, if A and B are not the same size, then A == B is an error.

The proper way to check for equality between two variables is to use the isequal function,

```
if isequal(A,B), ...
```

Several functions are helpful for reducing the results of matrix comparisons to scalar conditions for use with if, including

Isequal	True if arrays are numerically equal
Isempty	True for empty array.
all	True if all elements of a vector are nonzero.
any	True if any element of a vector is nonzero.

2- if-else statement

if-else statement keywords provide for the execution of alternate groups of statements. An end keyword, which matches the if, terminates the last group of statements. The groups of statements are delineated by the four keywords – no braces or brackets are involved. The basic form of if-else statement for use in a program file is:

```
if condition
   statement-1
else
   statement-2
end
```

or in simplest form :

```
if condition  statement-1, else statement-2,end
```

Example: the same example above will be repeated using if-else statement:

```
a = input('enter the value of a:');
b = input('enter the value of b:');
c = input('enter the value of c:');
if (b^2 -4*a*c == 0) & (a ~= 0)
```

```
    disp('The quadratic equation has equal roots')
    x = -b / (2*a)
else
    disp('The quadratic equation did not have equal roots')
end
```

3- if-elseif statement

The if-elseif statement executes groups of statements based on different expressions. So if our comparison contains many statements for many conditions then we must use the if-elseif statement. The basic form of this statement is:

```
if condition-1
   statement-1
elseif condition-2
   statement-2
.
.
.
elseif condition-N
   statement-N
else
  statement-N+1
end
```

Example: Suppose the random bank offers 9% interest on balances of less than $5000, 12% for balances of $5000 or more but less than $10000, and 15% for balances of $10000 or more. The following program calculates a customer's new balance after one year according to this scheme:

```
bal = input (' Enter bank balance:' );
if bal < 5000
   rate = 0.09;
elseif bal < 10000
```

```
   rate=0.12;
else
   rate = 0.15;
end
newbal = bal + rate * bal;
format bank
disp ('New balance is:')
disp (newbal)
```

4- switch and case statements

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 form of a while statement is:

```
switch variable
   case case_number_1,
      statement-1
   case case_number_2,
      statement-2
   .
   .
   .
   case case_number_N,
      statement-N
    otherwise,
      statement-N+1
 end
```

So the statements following the case statement are executed when the case number matches the variable value entry with the switch statement.

Note: Unlike the C language switch statement, MATLAB's switch does not fall through. If the first case statement is true, the other case statements do not execute. So, break statements are not required.

Example: Write a script file to enter an integer random numbers from 1 to 10 in a (3×3) matrix (named A). Find the requirements below depending on your entry from 1 to 4:

1. The transpose of matrix A.
2. The determinant of matrix A.
3. The inverse of matrix A.
4. The eigen values of matrix A

```
A=fix(rand(3,3)*10);
disp('your matrix is:')
A
n=input('Enter your choice from 1 to 4:')
switch n
   case 1
      disp('The transpose of matrix A is:')
      A'
   case 2
      disp('The determinant of matrix a is:')
      det(A)
   case 3
      disp('The inverse of matrix A is:')
      inv(a)
   case 4
      disp('The eigen values of matrix A is:')
      eig(a)
   otherwise
      disp('wrong number, enter another number')
end
```