Attia, John Okyere. “Control Statements .”
Electronics and Circuit Analysis using MATLAB.
Ed. John Okyere Attia
Boca Raton: CRC Press LLC, 1999

CHAPTER THREE

CONTROL STATEMENTS

3.1 FOR LOOPS

"**FOR**" loops allow a statement or group of statements to be repeated a fixed number of times. The general form of a for loop is

```
for index = expression
        statement group X
end
```

The expression is a matrix and the statement group X is repeated as many times as the number of elements in the columns of the expression matrix. The index takes on the elemental values in the matrix expression. Usually, the expression is something like

```
m:n  or m:i:n
```

where m is the beginning value, n the ending value, and i is the increment.

Suppose we would like to find the squares of all the integers starting from 1 to 100. We could use the following statements to solve the problem:

```
sum = 0;
for i = 1:100
   sum = sum + i^2;
end
sum
```

For loops can be nested, and it is recommended that the loop be indented for readability. Suppose we want to fill 10-by-20 matrix, b, with an element value equal to unity, the following statements can be used to perform the operation.

```
%
n = 10;              % number of rows
m = 20;              % number of columns
for i = 1:n
    for j = 1:m
        b(i,j) = 1;  % semicolon suppresses printing in the loop
    end
end
```

```
b                    % display the result
%
```

It is important to note that each for statement group must end with the word **end**. The following program illustrates the use of a for loop.

Example 3.1

The horizontal displacement $x(t)$ and vertical displacement $y(t)$ are given with respect to time, *t,* as

$$x(t) = 2t$$
$$y(t) = \sin(t)$$

For t = 0 to 10 ms, determine the values of $x(t)$ and $y(t)$. Use the values to plot $x(t)$ versus $y(t)$.

Solution*:*

MATLAB Script

```
%
for i= 0:10
    x(i+1) = 2*i;
    y(i+1) = 2*sin(i);
end
plot(x,y)
```

Figure 3.1 shows the plots of $x(t)$ and $y(t)$.

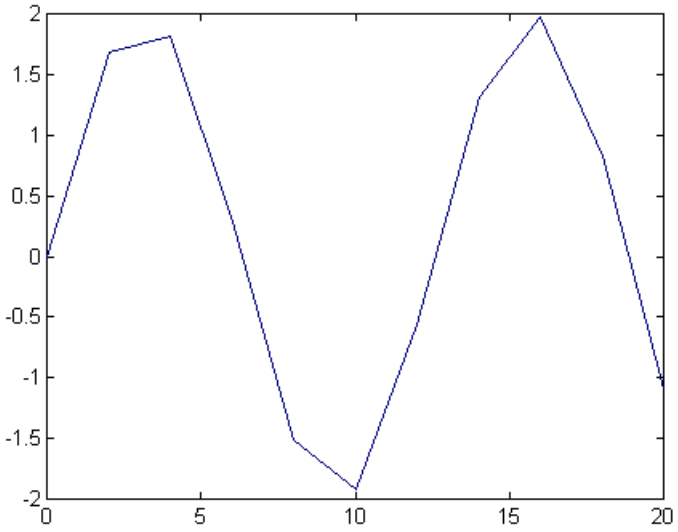

Figure 3.1 Plot of x versus y.

3.2 IF STATEMENTS

IF statements use relational or logical operations to determine what steps to perform in the solution of a problem. The relational operators in MATLAB for comparing two matrices of equal size are shown in Table 3.1.

Table 3.1
Relational Operators

RELATIONAL OPERATOR	MEANING
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
~=	not equal

When any of the above relational operators are used, a comparison is done between the pairs of corresponding elements. The result is a matrix of ones and zeros, with one representing TRUE and zero FALSE. For example, if

```
a = [1 2 3 3 3 6];
b = [1 2 3 4 5 6];
a == b
```

The answer obtained is

```
ans =
     1     1     1     0     0     1
```

The 1s indicate the elements in vectors *a* and *b* that are the same and 0s are the ones that are different.

There are three logical operators in MATLAB. These are shown in Table 3.2.

Table 3.2
Logical Operators

LOGICAL OPERATOR SYMBOL	MEANING
&	and
!	or
~	not

Logical operators work element-wise and are usually used on 0-1 matrices, such as those generated by relational operators. The & and ! operators compare two matrices of equal dimensions. If A and B are 0-1 matrices, then A&B is another 0-1 matrix with ones representing TRUE and zeros FALSE. The NOT(~) operator is a unary operator. The expression ~C returns 1 where C is zero and 0 when C is nonzero.

There are several variations of the IF statement:

- simple if statement
- nested if statement
- if-else statement

- if-elseif statement
- if-elseif-else statement.

- The general form of the **simple if statement** is

```
if logical expression 1
        statement group 1
end
```

In the case of a simple if statement, if the logical expression1 is true, the statement group 1 is executed. However, if the logical expression is false, the statement group 1 is bypassed and the program control jumps to the statement that follows the end statement.

- The general form of a **nested if statement** is

```
if logical expression 1
                statement group 1
        if logical expression 2
                statement group 2
        end
                statement group 3
end
statement group 4
```

The program control is such that if expression 1 is true, then statement groups 1 and 3 are executed. If the logical expression 2 is also true, the statement groups 1 and 2 will be executed before executing statement group 3. If logical expression 1 is false, we jump to statement group 4 without executing statement groups 1, 2 and 3.

- The **if-else statement** allows one to execute one set of statements if a logical expression is true and a different set of statements if the logical statement is false. The general form of the if-else statement is

```
if logical expression 1
                statement group 1
        else
                statement group 2
end
```

In the above program segment, statement group 1 is executed if logical expression 1 is true. However, if logical expression 1 is false, statement group 2 is executed.

- **If-elseif statement** may be used to test various conditions before executing a set of statements. The general form of the if-elseif statement is

```
if logical expression 1
                statement group1
        elseif logical expression 2
                statement group2
        elseif logical expression 3
                statement group 3
        elseif logical expression 4
                statement group 4
end
```

A statement group is executed provided the logical expression above it is true. For example, if logical expression 1 is true, then statement group 1 is executed. If logical expression 1 is false and logical expression 2 is true, then statement group 2 will be executed. If logical expressions 1, 2 and 3 are false and logical expression 4 is true, then statement group 4 will be executed. If none of the logical expressions is true, then statement groups 1, 2, 3 and 4 will not be executed. Only three elseif statements are used in the above example. More elseif statements may be used if the application requires them.

- **If-elseif-else statement** provides a group of statements to be executed if other logical expressions are false. The general form of the if-elseif-else statement is

```
if logical expression 1
                statement group1
        elseif logical expression 2
                statement group 2
        elseif logical expression 3
                statement group 3
        elseif logical expression 4
                statement group4
        else
                statement group 5
end
```

The various logical expressions are tested. The one that is satisfied is executed. If the logical expressions 1, 2, 3 and 4 are false, then statement group 5 is executed. Example 3.2 shows the use of the if-elseif-else statement.

Example 3.2

A 3-bit A/D converter, with an analog input x and digital output y, is represented by the equation:

$$\begin{aligned} y = 0 &\quad x < -2.5 \\ = 1 &\quad -2.5 \leq x < -1.5 \\ = 2 &\quad -1.5 \leq x < -0.5 \\ = 3 &\quad -0.5 \leq x < 0.5 \\ = 4 &\quad 0.5 \leq x < 1.5 \\ = 5 &\quad 1.5 \leq x < 2.5 \\ = 6 &\quad 2.5 \leq x < 3.5 \\ = 7 &\quad x \geq 3.5 \end{aligned}$$

Write a MATLAB program to convert analog signal x to digital signal y. Test the program by using an analog signal with the following amplitudes: -1.25, 2.57 and 6.0.

Solution

MATLAB Script

```
diary ex3_2.dat
%
y1 = bitatd_3(-1.25)
y2 = bitatd_3(2.57)
y3 = bitatd_3(6.0)
diary

function Y_dig = bitatd_3(X_analog)
%
% bitatd_3 is a function program for obtaining
%       the digital value given an input analog
%       signal
%
% usage:  Y_dig = bitatd_3(X_analog)
%       Y_dig is the digital number (in integer form)
```

```
%       X_analog is the analog input (in decimal form)
%
if X_analog < -2.5
  Y_dig = 0;
elseif X_analog >= -2.5 & X_analog < -1.5
  Y_dig = 1;
elseif X_analog >= -1.5 & X_analog < -0.5
  Y_dig = 2;
elseif X_analog >= -0.5 & X_analog < 0.5
  Y_dig = 3;
elseif X_analog >= 0.5 & X_analog < 1.5
  Y_dig = 4;
elseif X_analog >= 1.5 & X_analog < 2.5
  Y_dig = 5;
elseif X_analog >= 2.5 & X_analog < 3.5
  Y_dig = 6;
else
  Y_dig = 7;
end
Y_dig;
end
```

The function file, bitatd_3.m, is an m-file available in the disk that accompanies this book. In addition, the script file, ex3_2.m on the disk, can be used to perform this example. The results obtained, when the latter program is executed, are

```
y1 =
     2

y2 =
     6

y3 =
     7
```

3.3 WHILE LOOP

A **WHILE** loop allows one to repeat a group of statements as long as a specified condition is satisfied. The general form of the WHILE loop is

```
while expression 1
        statement group 1
end
statement group 2
```

When expression 1 is true, statement group 1 is executed. At the end of executing the statement group 1, the expression 1 is retested. If expression 1 is still true, the statement group 1 is again executed. However, if expression 1 is false, the program exits the while loop and executes statement group 2. The following example illustrates the use of the while loop.

Example 3.3

Determine the number of consecutive integer numbers which when added together will give a value equal to or just less than 210.

Solution

MATLAB Script

```
diary ex3_3.dat
% integer summation
int = 1; int_sum = 0;
max_val = 210;
while int_sum < max_val
    int_sum = int_sum + int;
    int = int + 1;
end
last_int = int
if int_sum == max_val
  num_int = int - 1
  tt_int_ct =  int_sum
 elseif int_sum  > max_val
    num_int = int - 1
    tt_int_ct = int_sum - last_int
 end
end
diary
```

The solution obtained will be

```
last_int =
            21
```

```
num_int =
          20

tt_int_ct =
          210
```

Thus, the number of integers starting from 1 that would add up to 210 is 20. That is,

$$1 + 2 + 3 + 4 + \ldots + 20 = 210$$

3.4 INPUT/OUTPUT COMMANDS

MATLAB has commands for inputting information in the command window and outputting data. Examples of input/output commands are echo, input, pause, keyboard, break, error, display, format, and fprintf. Brief descriptions of these commands are shown in Table 3.3.

Table 3.3
Some Input/output Commands

COMMAND	DESCRIPTION
break	exits while or for loops
disp	displays text or matrix
echo	displays m-files during execution
error	displays error messages
format	switches output display to a particular format
fprintf	displays text and matrices and specifies format for printing values
input	allows user input
keyboard	invokes the keyboard as an m-file
pause	causes an m-file to stop executing. Pressing any key cause resumption of program execution.

Break

The **break** command may be used to terminate the execution of ***for*** and ***while*** loops. If the break command exits in an innermost part of a nested loop, the

break command will exit from that loop only. The break command is useful in exiting a loop when an error condition is detected.

Disp

The **disp** command displays a matrix without printing its name. It can also be used to display a text string. The general form of the disp command is

```
disp(x)
disp('text string')
```

disp(x) will display the matrix x. Another way of displaying matrix x is to type its name. This is not always desirable since the display will start with a leading "x = ". **Disp('text string')** will display the text string in quotes. For example, the MATLAB statement

```
disp('3-by-3  identity matrix')
```

will result in

```
3-by-3 identity matrix
```

and

```
disp(eye(3,3))
```

will result in

```
1    0    0
0    1    0
0    0    1
```

Echo

The echo command can be used for debugging purposes. The **echo** command allows commands to be viewed as they execute. The echo can be enabled or disabled.

```
echo on  -        enables the echoing of commands
echo off -        disables the echoing of commands
echo -            by itself toggles the echo state
```

Error

The **error** command causes an error return from the m-files to the keyboard and displays a user written message. The general form of the command is

```
error('message for display')
```

Consider the following MATLAB statements:

```
x = input('Enter age of student');
if  x < 0
        error('wrong age was entered, try again')
end
x = input('Enter age of student')
```

For the above MATLAB statements, if the age is less that zero, the error message 'wrong age was entered, try again' will be displayed and the user will again be prompted for the correct age.

Format

The **format** controls the format of an output. Table 3.4 shows some formats available in MATLAB.

Table 3.4
Format Displays

COMMAND	MEANING
format short	5 significant decimal digits
format long	15 significant digits
format short e	scientific notation with 5 significant digits
format long e	scientific notation with 15 significant digits
format hex	hexadecimal
format +	+ printed if value is positive, - if negative; space is skipped if value is zero

By default, MATLAB displays numbers in "short" format (5 significant digits). **Format compact** suppresses line-feeds that appear between matrix displays, thus allowing more lines of information to be seen on the screen. **For-**

mat loose reverts to the less compact display. Format compact and format loose do not affect the numeric format.

fprintf

The **fprintf** can be used to print both text and matrix values. The format for printing the matrix can be specified, and line feed can also be specified. The general form of this command is

```
fprintf('text with format specification', matrices)
```

For example, the following statements

```
cap = 1.0e-06;
fprintf('The value of capacitance is %7.3e Farads\n', cap)
```

when executed will yield the output

```
The value of capacitance is 1.000e-006 Farads
```

The format specifier %7.3e is used to show where the matrix value should be printed in the text. 7.3e indicates that the capacitance value should be printed with an exponential notation of 7 digits, three of which should be decimal digits. Other **format specifiers** are

```
%f  -    floating point
%g -     signed decimal number in either %e or  %f format,
         whichever is shorter
```

The text with format specification should end with **\n** to indicate the end of line. However, we can also use \n to get line feeds as represented by the following example:

```
r1 = 1500;
fprintf('resistance is \n%f Ohms \n', r1)
```

the output is

```
resistance is
1500.000000 Ohms
```

Input

The **input** command displays a user-written text string on the screen, waits for an input from the keyboard, and assigns the number entered on the keyboard as the value of a variable. For example, if one types the command

```
r = input('Please enter the four resistor values');
```

when the above command is executed, the text string 'Please, enter the four resistor values' will be displayed on the terminal screen. The user can then type an expression such as

```
[10 15 30 25];
```

The variable r will be assigned a vector [10 15 30 25]. If the user strikes the return key, without entering an input, an empty matrix will be assigned to r.

To return a string typed by a user as a text variable, the input command may take the form

```
x = input('Enter string for prompt', 's')
```

For example, the command

```
x = input('What is the title of your graph', 's')
```

when executed, will echo on the screen, 'What is the title of your graph.' The user can enter a string such as 'Voltage (mV) versus Current (mA).'

Keyboard

The **keyboard** command invokes the keyboard as an m-file. When the word **keyboard** is placed in an m-file, execution of the m-file stops when the word keyboard is encountered. MATLAB commands can then be entered. The keyboard mode is terminated by typing the word, "**return**" and pressing the return key. The keyboard command may be used to examine or change a variable or may be used as a tool for debugging m-files.

Pause

The **pause** command stops the execution of m-files. The execution of the m-file resumes upon pressing any key. The general forms of the pause command are

```
pause
pause(n)
```

Pause stops the execution of m-files until a key is pressed. **Pause(n)** stops the execution of m-files for n seconds before continuing. The pause command can be used to stop m-files temporarily when plotting commands are encountered during program execution. If pause is not used, the graphics are momentarily visible.

SELECTED BIBLIOGRAPHY

1. MathWorks, Inc., *MATLAB, High-Performance Numeric Computation Software*, 1995.

2. Biran, A. and Breiner, M., *MATLAB for Engineers*, Addison-Wesley, 1995.

3. Etter, D.M., *Engineering Problem Solving with MATLAB*, 2nd Edition, Prentice Hall, 1997.

EXERCISES

3.1 Write a MATLAB program to add all the even numbers from 0 to 100.

3.2 Add all the terms in the series

$$1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\ldots$$

until the sum exceeds 1.995. Print out the sum and the number of terms needed to just exceed the sum of 1.995.

3.3 The Fibonacci sequence is given as

1 1 2 3 5 8 13 21 34 …

Write a MATLAB program to generate the Fibonacci sequence up to the twelfth term. Print out the results.

3.4 The table below shows the final course grade and its corresponding relevant letter grade.

LETTER GRADE	FINAL COURSE GRADE
A	90 < grade ≤ 100
B	80 < grade ≤ 90
C	70 < grade ≤ 80
D	60 < grade ≤ 70
F	grade ≤ 60

For the course grades: 70, 85, 90, 97, 50, 60, 71, 83, 91, 86, 77, 45, 67, 88, 64, 79, 75, 92, and 69

(a) Determine the number of students who attained the grade of A and F.
(b) What are the mean grade and the standard deviation?

3.5 Write a script file to evaluate y[1], y[2], y[3] and y[4] for the difference equation:

$$y[n] = 2y[n-1] - y[n-2] + x[n]$$

for $n \geq 0$. Assume that $x[n] = 1$ for $n \geq 0$, $y[-2] = 2$ and $y[-1] = 1$.

3.6 The equivalent impedance of a circuit is given as

$$Z_{eq}(jw) = 100 + jwL + \frac{1}{jwC}$$

If L = 4 H and C = 1 μF,

(a) Plot $\left|Z_{eq}(jw)\right|$ versus w. (b) What is the minimum impedance?

(c) With what frequency does the minimum impedance occur?