Chapter 5

Control instructions (`if` and `switch` statements, `for` and `while` loops)

1. Introduction

In general, a computer program can be considered as a list of sections or statements. In this chapter we will provide an overview about the conditional or branching statements, in which, we can select among the statements whether are executed or not. In addition, if a statement is repeated more than three times, it is meaningful to use repetition or loop structures. MATLAB supports two types of loops: **`for`** and **`while`**, which are the subject of the second part of this chapter.

2. `if` and `switch` statements

In MATLAB environments, **`if-else`** and **`switch`** are important control tasks and conditional statements that help the users to take decisions based on various conditions. However, the **`if-else`** statements are also employed to generate two potential parts of a program, one to operate when a certain condition is met and another to run if the condition is false. In addition, the **`switch`** statement executes the appropriate block or a set of instruction based on the value of an expression.

2.1. `if-else` statement

The **`if-else`** statement is employed to execute various parts of a program based on whether a specified condition is true or false.

a. if statement

The **`if`** statement is utilized to test and examine a condition. If the condition is true, a program or a set of instructions is executed. The general form of the if statement is as follows.

```
if   condition
 <Instructions>
end
```

An example of using the if statement is indicated in this script file, in which we calculate the root square of a positive number.

SquareRoot.m

```
x = input('Please enter a number: '); % Let the user enter a number
if x >= 0                             % Check if the number is >= 0
    fprintf('The square root of %.2f is : %.2f\n', x, sqrt(x));
end
```

The command window bellow shows the execution of this script for two cases, positive and negative numbers, in which the instruction is not executed in the case of the negative number.

```
>> SquareRoot
Please enter a number: 4
The square root of 4.00 is: 2.00
>>
>> SquareRoot
Please enter a number: -4
>>
```

b. if-else statement

Generally, the **`if`** statement is followed by another statement called **`else`** that allows to run another code or program when the condition in the **`if`** statement is false.

Therefore, the general syntax for an if-else statement in MATLAB is as follows:

```
if  condition
 <Instruction 1>
else
 <Instruction 2>
end
```

The following example shows the use of an `if-else` statement to calculate the absolute value of a number.

absolute.m

```
x = input('Please enter a number: '); % Let the user enter a number
if x < 0                              % Check if x < 0
    fprintf('The absolute value of %.2f is %.2f \n',x, -x);
else
    fprintf('The absolute value of %.2f is %.2f \n',x, x);
end
```

The command window bellow shows the execution of this script.

```
>> absolute
Please enter a number: -5.36
The absolute value of -5.36 is 5.36
```

c. `if-elseif-else` statement

In some cases, **`if`** statement can be followed by one or more optional **`elseif`** and an **`else`** statement, which is very useful to test various conditions. The basic syntax of **`if-elseif-else`** statement is expressed as follows:

```
if condition 1
   <Instructions 1> % Executes the code when the condition 1 is true
elseif condition 2
   <Instructions 2> % Executes the code when the condition 2 is true
else
   <Instructions 3> % Executes the code when the above conditions are false
end
```

The next script is introduced to show the use of **`if-elseif-else`** statement to determine if a number is positive, negative or null.

numberType.m

```
x = input('Please enter a number: '); % Let the user enter a number
if x > 0                              % Check if the number is positive
    type = 'positive';
elseif x < 0
```

```
    type = 'negative';
else
    type = 'null';
end
fprintf('%.2f is a %s number.\n',x, type); % Print the number type
```

A set of executions of the above script for the three cases is indicated in the bellow command window.

```
>> numberType
Please enter a number: 4
4.00 is a positive number.
>> numberType
Please enter a number: -8
-8.00 is a negative number.
>> numberType
Please enter a number: 0
0.00 is a null number.
```

2.2. Switch statement

The mechanism of the **switch** statement is based on selecting one among various blocks to be executed, this is done according to the value of a specific expression. In other words, the **switch** statement allows to execute different blocks of code depending on the value of the expression.

The basic syntax of the switch statement is expressed as follows:

```
switch expression
case caseValue1
        % run this cod if the expression matches caseValue1
case caseValue2
        % run this cod if the expression matches caseValue2
    ...
otherwise
        % Code to execute if the expression doesn't match any case
end
```

The following example displays the grade of a student based on their numerical score.

grade.m

```
% Input the numerical grade
numGrade = input('Enter the numerical grade (0-20): ');
% Determine the grade using a switch statement
switch true
case numGrade >= 18
        Grade = 'Excellent';
case numGrade >= 15
        Grade = 'Good';
case numGrade >= 10
        Grade = 'Middle';
case numGrade >= 5
        Grade = 'Weak';
case numGrade >= 5
        Grade = 'Very weak';
otherwise
        Grade = 'not a valid numerical grade’;
end
% Display the letter grade
fprintf('The letter grade for %.2f is: %s\n', numGrade, Grade);
```

The command window bellow shows the execution of this script.

```
>> grade
Enter the numerical grade (0-20): 16
The letter grade for 16.00 is: good
```

The next program determines the type of a vehicle based on a category

vehicle.m

```
vehicleType = input('Enter the vehicle type : ', 's');

switch vehicleType

case {'car' , 'truck', 'bus'}

        disp('This is a road vehicle.');

case {'boat' , 'ship'}

        disp('This is a water vehicle.');

case {'plane' , 'helicopter'}

        disp('This is an air vehicle.');
```

```
otherwise

       disp('Unknown vehicle type.')

end
```

The below command window shows the execution of the vehicle script.

```
>> vehicle
Enter the vehicle type : plane
This is an air vehicle.
```

3. `for` and `while` loops

In MATLAB, we can distinguish two types of loop statements **counted loop** in which the statements are repeated with a specified number of times and **conditional loop** where the statements are executed until reaching defined goal or conditions become false. The counted loop is based on the **`for`** statement and the **`while`** statement is usually adopted for conditional loop. We present in what follows the **`for`** and the **`while`** statements.

3.1. `for` loop

In general, we use the **`for`** loop to repeat a set of statements with a known ahead times called actions of the loop. The general form of the **`for`** loop is given as indicated below.

```
for iterate_var = range
    <action>
end
```

where `iterate_var` represents the loop variable, `range` indicates the range in which the loop variable iterates and `<action>` represents the set of statements to be repeated until completing all iterations. In the below example we implement a script file to present the use of the **`for`** loop for printing three times 'Hello world'.

for_loop_msg.m

```
% Program print 'Hello world' three times using the for loop.
for itr = 1:3
    disp('Hello world');
end
```

The execution of the above script is indicated in the next command window.

```
>> for_loop_msg
Hello world
Hello world
Hello world
```

The next example shows the use of the **for** loop to calculate the square of the first five numbers in which the loop variable is used.

for_loop.m

```
% program calculating the square of the first three
% numbers using the for loop.
for itr = 1:3
    s = itr^2;
    fprintf('The square of %d is %d \n',itr,s)
end
```

The execution of this script is shown in the next command window.

```
>> for_loop
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
```

3.2. while loop

Instead of repeating statements with a known ahead times, the **while** loop repeats the actions as long as the condition is true. The general form of the **while** loop is defined as illustrated bellow.

```
while condition
    <action>
end
```

where the condition is evaluated in each iteration and if it is logically true, the action is executed until the condition becomes false. To avoid an infinite loop in which the execution of the action never halts, the condition must become false. In MATLAB, if the infinite loop occurs, we press **Ctrl-C** to exit the loop. A simple example depicts the use of the while loop concerns the printing of the first four integer numbers as indicated below.

iterate.m

```
m = 1;
while m < 5
    disp(m);
    m = m + 1;
end
```

The execution of this script is shown in the next command window.

```
>> iterate
     1

     2

     3

     4

```

The second example introduced to calculates the factorial of a number as showcased in the below script.

factorial.m

```
n = input('Please enter a positive number: '); % Let the user enter a
                                               % number
my_n = n; % Save n value to use it in fprintf function
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
fprintf('%d! = %d \n',my_n, f)
```

The below command window shows the execution of the factorial script.

```
>> factorial
Please enter a positive number: 8
8! = 40320
```

In some cases, we need to quit a **`for`** or a **`while`** loop before reaching the end, then we use the **`break`** statement to exit the loop. However, if we need to skip the rest of actions in the loop and begin the next iteration, we use the **continue** statement. In MATLAB, **break** is defined only inside a **for** or a **while** loop.

4. Practical work 5

if and switch statements

1. Write the following scripts and explain what they generate.

```
% Script 1
nbr = input('Enter a number : ');
if mod(nbr, 2)==0
    disp('The number is even')
else
    disp('The number is odd')
end


% Script 2
Day = input('Please enter a day name : ', 's');
switch Day
    case 'Sunday'
        disp('Start of the week.');
    case 'Monday'
        disp('Second day of the week.');
    case 'Tuesday'
        disp('Midweek.');
    case 'Wednesday'
        disp('Almost the weekend.');
    case 'Thursday'
        disp('Last workday of the week.');
    case {'Friday', 'Sunday'}
        disp('Weekend.')
    otherwise
        disp('Not a valid day.');
end
```

2. Using **if-else statements**, write a MATLAB program to calculate a student's average score of three exams and check if he has passed the exam or not, with the passing score set at 10 points.
3. Using **if-else statements**, write a MATLAB program to calculate the determinant of a second-degree equation and find its solutions
4. Rewrite the previous programs using **switch statement**

for and while loops

1. Write the following scripts and explain what they generate

```
% Script 1
sum = 0;
for i = 1:10
    sum = sum + i;
end
fprintf('Sum of first ten integers is: %d\n', sum);

% Script 2
nbs = [3, 5, 8, 7, 9];
for i = 1:length(nbs)
    nbs(i) = nbs(i) * 2;
end
disp(['Doubled array: ', num2str(nbs)])

% Script 3
sum = 0;
i = 1;
while sum <= 30
    sum = sum + i;
    i = i + 1;
end
disp(['The sum exceeded 30 and is: ', num2str(sum)]);
```

2. Using for loop, write a program to count the number of even numbers in an array.
3. Upgrade the above script to count the sum and the product of the even numbers.
4. Write a script to save in a vector 10 numbers entered by the user based on the for loop.
5. Repeat the above scripts using the while loop.
6. Write a scripts file to generate a multiplication table for a given number

5. Exercises

1. Write a script that asks the user for an integer number and displays whether it is even or odd.
2. Read two numbers *a* and *b* and display which one is larger or if they are equal.
3. Input a number from 0-9 and use switch to print it as a word (e.g. 0: “zero”, 1: “one”, etc.).
4. Using a for loop, Ask the user for N, and compute S = 1+2+⋯+N

5. Compute $S = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$ for a user-specified N.
6. Sum until threshold: Initialize mysum = 0. Continuously add random numbers between 0 and 1 until mysum exceeds 10 and count how many iterations were needed.
7. Write a loop that checks whether a given number n is prime.
8. Given a random vector v, use a while loop and indexing to compute the sum of only positive elements.
9. Menu-driven program: Display a menu:

   ```
   1. Add two numbers
   2. Multiply two numbers
   3. Exit
   ```

   Use a while loop and switch statement to handle user choices until they select Exit.

10. Area calculator: Display a menu:

    ```
    1. Circle
    2. Rectangle
    3. Triangle
    ```

    Ask the user to choose an option and input the necessary dimensions.
    Use switch to calculate and display the area.
11. BMI classification: Ask for height (m) and weight (kg), compute $BMI = weight / height^2$, and classify using:
    - Underweight: < 18.5
    - Normal: 18.5 – 24.9
    - Overweight: 25 – 29.9
    - Obese: ≥ 30
12. Guess the number game: Generate a random integer between 1 and 100.
    Ask the user to guess the number and display: "Too high" or "Too low" until they find it. Count the number of attempts.
13. Armstrong number: An Armstrong number is one where the sum of the cubes of its digits equals the number itself (e.g. 153 : $1^3+5^3+3^3 = 153$).
    Write a while loop that checks this.