Functions

Functions in Scilab are programs developed to obtain mathematical outputs upon their execution. Due to their mathematical outputs, it becomes more appropriate to use function for further mathematical use computations. For example, if outputs of programs 4 and 5 of Exercise 3.3 are in the form of a string, then it is difficult to use these outputs for further processing of information drawn through their outputs. Rather, if the output of both the programs be obtained in terms of numbers or Boolean variables, then the further processing of outputs becomes convenient. To understand this, let us test whether a given number is a Palindrome and Armstrong too. If the output of both the programs are expressed in terms of numbers or Booleans instead of messages conveyed as strings, then their conclusions can further be treated through mathematical or logical operators. Knowing the usefulness of function programs, let us learn first about how to call a function and use its output(s).

3.1 Calling a function

There are three main components of call of a Scilab function, as listed below.

- name of the function
- input argument(s)
- output argument(s).

The calling sequence of any function in most general form is as following:

```
[y1, ... ,yn] = function_name(x1, ... , xn)
```

Remarks: In the calling sequence of a function,

1. the function name is function_name;
2. input arguments are x1, ... , xn;
3. output arguments are y1, ... ,yn;
4. if there are multiple input or output arguments, then they should to be separated by comma (,);
5. input argument(s) are to be enclosed in parenthesis (generally called round brackets), *i.e.*, “(” and “)”;
6. input argument(s) are to be enclosed in square brackets (or simply called brackets), *i.e.*, “[” and “]”;

Caution: Any space should not be given after the function name and after the left parenthesis.

Built-in functions

There are multiple built-in functions in Scilab. Some of them we have discussed in Section 1.4 of Chapter 1. Built-in functions are simply to be called for getting the values of their output arguments by simply entering their correct calling sequence.

4.2 Defining a function

Scilab has been developed so that it can be extended by the users. Users can develop their own programs as functions for better mathematical application. In this section, let us learn the procedure and syntax of coding a function program in Scilab. The coding of a function program is majorly like any general program except some typical differences, which are as following.

1. The code of any function program has to start with the keyword function and end with the keyword endfunction.
2. As function programs are intended to give mathematical outputs, so the desirable outputs are needed to be assigned to the variables written as output arguments.
3. As function programs, once developed, are used for multiple values of inputs arguments, therefore whichever variables are used as inputs they should be included as input arguments.

Syntax of a general function program code is as following:

```
        function [y1, ... ,yn] = function_name(x1, ... , xn)
    Scilab Command 1
    Scilab Command 2
    ...
    y1 = assign value
    ...
    yn = assign value
endfunction
```

Remarks:

1. Space must be provided after output arguments and the assignment operator sign "=".
2. The name of the function must abide the rules of defining a valid variable name in Scilab.
3. User defined function names should be avoided to be same as built-in function names.

4. All the output variables must be assigned values in the code of the function program.
5. A used defined function program is required to be executed before being called for the first time. Execution is required again before calling the function program for every time when the code is edited.
6. General Scilab programs are saved in the permanent memory of the computer with the file extension ".sce". Whereas, the function programs in Scilab are saved with the file extension ".sci". Over that, the file name of the program file must be same as the function name.

Let us learn this through an example of a function program developed by modifying the program in Example 3.3 in Chapter 3.

```
function [y]=signum(x)          if (x ==
0) then         y = 0;     elseif (x > 0)
then     y = 1;     else         y = -1;
end
endfunction
```

This function can be executed and then called for different inputs as demonstrated below.

```
-->exec('D:\SCILAB\signum.sci', -1)

-->[y] = signum(5)  y  =

         1.

-->[y] = signum(-10)  y  =

   - 1.   -->[y] = signum(0)  y

=

         0.
```

Remark: Variable names used in input and output arguments are local to the function program. The interpretation of this concept demonstrated through the above-discussed function program is given below.[1]

1. Although, x is the input variable for the above function program but it can be called using any other variable name also. For example, for the above function program:

```
-->a = 5; -->[y] = signum(a)

y  =

          1.
```

This feature is due to a mechanism of Scilab explained as following. During the call of function program signum with variable a used as input argument, Scilab assigns the value of variable a to the local variable x and then computes the value of variable y pertaining to the output argument.

2. Similarly, any variable name can be used as output variable during the call of a function program. For example, for the above function program:

```
-->[b] = signum(a)  b  =

    1.
```

This feature is due to another mechanism of Scilab explained as following. During the call of function program signum with input argument assigned value directly or through variable, Scilab computes the value of variable y pertaining to the output argument. It then assigns the computed value of variable y to the variable b, because the call of the function program uses output argument as b.

Exercise 3

1. Write a Scilab program to solve a Quadratic Equation $ax^2 + bx + c = 0$. The input to the function are the values "a, b, c" and the output of the function should be in the variable names "p, q" appropriately declared.
2. Write a Scilab program to compute sum of first 'n' natural numbers.
3. Write a Scilab program to compute factorial of a natural number 'n'.
4. Write a Scilab program using for loop to compute the sum of two given matrices, if they are of comparable order.
5. Write a Scilab program to compute the number of permutations & number of combinations for given values of 'n' and 'r'.
6. Write a Scilab program to compute sum of digits of a natural number 'n'.
7. Write a Scilab program to find the number of digits of a natural number 'n'.

8. Write a Scilab program to obtain a number with digits as the reverse of a given natural number ‘n’.
9. Write a Scilab program to test whether a given number is Palindrome.
10. Write a Scilab program to test whether a given number is Armstrong number.
11. Write a Scilab program to obtain the binary equivalent of a given decimal number.
12. Write a Scilab program to obtain the decimal equivalent of a given binary number.
