3. SS Lab 3

In the third lab of Signals and Systems, creating functions, control structures and constructs would be explored in the MATLAB environment.

Suggestions for improvement or correction of the manuscript would be appreciated.

3.1 Lab Objectives

In this lab, the following topics would be covered:

- Functions creation
- Control structures
- Relational constructs
- Logical constructs
- Branching constructs
- Looping constructs

3.2 Function creation

A function can be created by the following syntax:

```
Function [output1,output2,...]
= function_name(input1,input2,...)
```

A function is a reusable piece of code that can be called from a program to accomplish some specified functionality. A function takes some input arguments and returns some output. To create a function that adds two numbers and stores the result in a third variable, type in it the following code in the MATLAB Editor:

```
function add

x = 3;

y = 5;

z = x + y
```

Save the file by the name of add (in work folder, which is chosen by default), go to the Command Window and write:

```
>> add
```

You will see that the sum z is displayed in the Command Window.

```
z =

   8
```

Now go to the Editor to write a new program as follows:

```
function addv(x,y)

z = x + y
```

Save the above program with a new name addv, go to the Command Window and type the following:

```
>> addv(3, 5)
```

The result would be:

```
z =

   8
```

Then type:

```
>> addv(5, 5)
```

The result would be:

```
z =

   10
```

We have actually created a function of our own, called it in the Command Window and given values to the variables `(x, y)`.

Go to the Editor and modify the program as follows:

```
function adv(x, y)
%-------------------------------------------------
% This function takes two values as input,
% finds its sum, & displays the result.
% inputs: x & y
% output: z
% Example: addv(3,6)
% Result: z=9
%-------------------------------------------------
z = x + y
```

Save the program with the name adv, go to Command Window, type the following:

```
>> help adv
```

What did you notice?

3.2.1 Script vs. Function

Table 3.1 depicts the key differences between script and function.

3.3 Control Structures

Control-of-flow in MATLAB programs is achieved with logical/relational constructs, branching constructs, and a variety of looping constructs.

3.3.1 Relational and Logical Constructs

The relational operators in MATLAB are:

```
Operator Description
==================================
<     less than
```

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

Note that = is used in an assignment statement while == is used in a relation.

Table 3.1: Differences between script and function

Script	Function
A script is simply a collection of MATLAB commands in an M-file. Upon typing the name of the file (without the extension), those commands are executed as if these have been entered via the keyboard.	Functions are used to create user-defined MATLAB commands.
A script can have any file name.	A function file is stored with the name specified after keyword function.
The commands in the script can refer to the variables already defined in MATLAB, which are said to be in the global workspace.	When a function is invoked, MATLAB creates a local workspace. The commands in the function cannot refer to variables from the global (interactive) workspace unless they are passed as inputs. By the same token, variables created as the function executes are erased when the execution of the function ends, unless they are passed back as outputs.

Relations may be connected or quantified by the logical operators.

```
Operator Description
==================================
&     and
|     or
~     not
==================================
```

When applied to scalars, a relation is actually the scalar 1 or 0 depending on whether the relation is true or false (indeed, throughout this section you should think of 1 as true and 0 as false). For example:

```
>> 3 < 5

ans =

   1

>> a = (3 == 5)

a =

   0
```

When logical operands are applied to matrices of the same size, a relation is a matrix of 0's and 1's giving the value of the relation between corresponding entries. For example:

```
>> A = [ 1 2; 3 4 ];

>> B = [ 6 7; 8 9 ];

>> A == B
```

```
ans =
   0  0
   0  0
>> A < B
ans =
   1  1
   1  1
```

To see how the other logical operators work, you should also try:

```
>> ~A
>> A&B
>> A & ~B
>> A | B
>> A | ~A
```

3.3.2 Branching Constructs

MATLAB provides several language constructs for branching a program's control of flow.

If-end Construct:

This is the most basic construct.

```
if <condition>
   <program>
end
```

Here the condition is a logical expression that will evaluate to either true or false (i.e., with values 1 or 0). When a logical expression evaluates to 0, program control moves on to the next program construction. You should keep in mind that MATLAB regards `A==B` and `A<=B` as functions with values 0 or 1.

```
>> a = 1;
>> b = 2;
>> if a < b
c = 3;
end
>> c
c =
   3
```

If-else-end Construct:

This construct is used quite frequently in MATLAB programming.

```
if <condition1>
   <program1>
```

```
else
   <program2>
end
```

In this case, if the condition is 0, then `program2` is executed.

If-elseif-end Construct:

Another variation is that if one condition is false, check another condition.

```
if <condition1>
   <program>
elseif <condition2>
   <program2>
end
```

Now if `condition1` is not 0, then `program1` is executed, if `condition1` is 0 and if `condition2` is not 0, then `program2` is executed, and otherwise control is passed on to the next construction.

3.3.3 Looping Constructs

For Loop:

A `for` loop is a construction of the form as follows:

```
for I = 1 : n
   <program>
end
```

This will repeat `program` once for each index value `i = 1, 2 .... n`. Here are some examples of MATLAB `for` loop capabilities:

A basic `for` loop could be like:

```
>> for i = 1 : 5
   c = 2*i
end
c =
   2
..... lines of output removed ...
c =
   10
```

This code computes and prints `c = 2*i` for `i = 1, 2, ... 5`.

`For` looping constructs may be nested. Here is an example of creating a matrix content inside a nested `for` loop

```
>> for i = 1:10
```

```
  for j = 1:10
    A(i, j) = i / j;
  end
end
```

There are actually two loops here, with one nested inside the other; they define `A(1,1), A(1,2), A(1,3) ... A(1,10), A(2,1), A(2,2) ... A(2,10), ... A(10,1), A(10,2) ... A(10,10)` in that order.

MATLAB will allow you to put any vector in place of the vector `1:n` in this construction. Thus, the following construction is perfectly valid:

```
>> for i = [2,4,5,6,10]
  <program>
end
```

In this case, the program will execute 5 times and the values for the variable `i` during execution are successively `2, 4, 5, 6, 10`.

While Loop:

A `while` loop is a construction of the form:

```
c
m = 2;
while m<=n
  l=l+1;
  m=2*m;
end
```

3.4 Tasks

Perform the following tasks:

3.4.1 Task 01

Write a function that accepts temperature in degrees `F` and computes the corresponding value in degrees `C`. The relation between the two is:

$$T_C = \frac{5}{9}\,(T_F - 32\,)$$

Make sure that you test your function with three different input values.

3.4.2 Task 02

For the arrays `x` and `y` given below, write MATLAB code to find all the elements in `x` that are greater than the corresponding elements in `y`.

```
x = [-3, 0, 0, 2, 6, 8]
y = [-5, -2, 0, 3, 4, 10]
```

3.4.3 Task 03

Using the branching constructs, for $0 < a \leq 16$, find the values of C defined as follows:

$$C = \begin{cases} 4ab & \text{for} \quad 1 \leq a \leq 8 \\ ab & \text{for} \quad 8 < a \leq 16 \end{cases}$$

and $b = 12$.

3.4.4 Task 04

For the values of integer a going from 1 to 10, using separately the methods of branching constructs and the Boolean alternative expressions, find the values of C if:

$$C = \begin{cases} a^2 & \text{for} \quad a < 3 \\ a + 3 & \text{for} \quad 3 \leq a \leq 7 \\ a & \text{for} \quad a > 7 \end{cases}$$

3.4.5 Task 05

Rewrite the following statements to use only one `if` statement.

```
if x < y
   if z < 5
      w = x*y*z
   end
end
```

3.4.6 Task 06

Using `for` loop, generate the cube of the first ten positive integers.

3.4.7 Task 07

Add the following two matrices using `for` loop.

$$A = \begin{bmatrix} 5 & 12 & 3 \\ 9 & 6 & 5 \\ 2 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 2 & 1 & 9 \\ 10 & 5 & 6 \\ 3 & 4 & 2 \end{bmatrix}$$

3.4.8 Task 08

Write MATLAB function that creates a special square matrix that has ones in the first row and first column, and whose remaining elements are the sum of two elements i.e. the element above and the element to the left, if the sum is less than 20. Otherwise, the element is the maximum of those two element values. Name the function `specmat` with a single input defining the matrix dimension. A sample program run is:

```
>>specmat(3)
ans =
   1  1  1
   1  2  3
   1  3  6
```

3.4.9 Task 09

Consider the following script file. Fill in the lines of the following table with the values that would be displayed immediately after the `while` statement if you run the script file. Write in the values that the variables have each time the `while` statement is executed. You might need more or fewer lines in the table. Then type in the file, and run it to check your answers.

```
k = 1; b = -2; x = -1; y = -2;
while k <= 3
  k, b, x, y
  y = x^2 -3;
  if y < b
    b = y;
  end
  x = x + 1;
  k = k + 1;
end
```

Pass	k	b	x	y
First				
Second				
Third				
Fourth				
Fifth				

3.4.10 Task 10

Create an M-file that inputs a number from user and then finds out the factorial of that number.

3.4.11 Task 11

Create an M-file that takes two vectors from user. Make sure that the second vector taken is of the same size as the first vector (Hint: use `while` loop). In a `while` loop, generate a third vector that contains the sum of the squares of corresponding entries of both the vectors.

3.4.12 Task 12

Perform the following commands on various matrices and comment on each.

```
>> ~A
>> A&B
>> A & ~B
>> A | B
>> A | ~A
```

3.4.13 Task 13

Design a function `Fib(N)` that takes `N` as an input and generates a Fibonacci sequence for `N`. Fibonacci sequence is a tile of squares whose side lengths are successive or each number is the sum of the previous number.

3.4.14 Task 14

Write a user-defined MATLAB function `Calculate`, with two input and two output arguments that determines the height in centimeters (cm) and mass in kilograms (kg) of a person from his height in inches (in.) and weight in pounds (lb).

a) Determine the height and mass of a 5 ft. 15 in. person in SI units who weighs 180 lb.
b) Determine your own height and weight in SI units.

3.4.15 Task 15

File handling in MATLAB. Create files of different formats in MATLAB. Use the following commands to create a text file using MATLAB commands:

1. Open a file using `fopen`

```
op = fopen('weekdays.txt','wt');
```

2. Write the output using `fprintf`

```
fprintf(op,'Sunday\nMonday\nTuesday\nWednesday\n');
fprintf(op,'Thursday\nFriday\nSaturday\n');
```

3. Close the file using `fclose`

```
fclose(op);
```

3.4.16 Task 16

Implement any sorting and searching algorithm of your choice by creating user-defined functions in MATLAB.