Signals & Systems Laboratory

CSE- 301L

Lab # 03

OBJECTIVES OF THE LAB

In this lab, we will get an understanding of the following topics:

- *Making Functions*
- *Control Structures*
- *Relational Constructs*
- *Logical Constructs*
- *Branching Constructs*
- *Looping constructs*

3.1 MAKING FUNCTIONS

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 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:

```
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 back to the command window and write

```
>> add
z =
   8
```

You see that the sum z is displayed in the command window. Now go back to the editor/debugger and modify the program as follows

```
function addv(x,y)
z = x + y
```

Save the above program with a new name **addv**, go back to the command window and type the following:

```
>> addv(3, 5)
z =
   8
>> addv(5, 5)
z =
  10
```

We have actually created a function of our own and called it in the main program and gave values to the variables (x, y).

Now go back to the editor/debugger 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 same name **addv**, go back to command window, type the following >> help addv

---

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

3.1.1 Script Vs Function

1) 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 they had been entered at the keyboard.
   Functions are used to create user-defined Matlab commands.

2) A script can have any name.
   A function file is stored with the name specified after keyword function.

3) 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.

----------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

$$T_c = \frac{5}{9}(T_c - 32)$$

Be sure to test your function.

3.2 CONTROL STRUCTURES

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

3.2.1 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.

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
```

----------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]

3.2.2 Branching constructs

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

i. **if-end Construct :** The most basic construct is:

```
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.

Example:

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

ii. **If-else-end Construct:** Frequently, this construction is elaborated with

```
if <condition1>
      <program1>
else
      <program2>
end
```

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

iii. **If-elseif-end Construct:** Another variation is

```
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.

----------TASK 03----------

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.

----------TASK 04----------

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

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

----------TASK 05----------

Rewrite the following statements to use only one if statement.

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

3.2.3 Looping constructs

i. **For Loop :** A for loop is a construction of the form

```
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's for loop capabilities:

Example: The basic for loop >> for i

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

computes and prints "c = 2*i" for i = 1, 2, ... 5.

Example: 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.

Example: MATLAB will allow you to put any vector in place of the vector 1:n in this construction. Thus the construction

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

is perfectly legitimate.

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

----------TASK 06----------

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

----------TASK 07----------

Add the following two matrices using for loop.

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

----------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 single input defining the matrix dimension. A sample program run is:

```
>>specmat(3)
ans =
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$$

ii. While Loops

A while loop is a construction of the form

```
while <condition>
      <program>
end
```

where condition is a MATLAB function, as with the branching construction. The program will execute successively as long as the value of condition is not 0. While loops carry an implicit danger in that there is no guarantee in general that you will exit a while loop. Here is a sample program using a while loop.

Example:

```
function l = twolog(n)
% l = twolog(n). l is the floor of the base 2
% logarithm of n.
l = 0;
```

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

----------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 ran the script file. Write in the values 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				

----------TASK 10----------

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

----------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.
