

## EE 301 Signals & Systems I MATLAB Tutorial with Questions

Under the content of the course EE-301, this semester, some MATLAB questions will be assigned in addition to the usual theoretical questions. This MATLAB tutorial has been prepared to serve as a means for teaching basic MATLAB skills (such as array generation, graph generation etc.) and help the students through the process of practicing MATLAB.

**Fact:** The most favorite commands of the MATLAB professionals are ‘**help**’ and ‘**lookfor**’.

To get help about the usage of any function in MATLAB, use the command

```
>> help function_name
```

To learn the function name which does something, use the command

```
>> lookfor keyword
```

**Q.1.** In this question, you will learn how to create and observe variables in MATLAB. Creating variables in MATLAB is done by just selecting a name for the variable and equating it to its initial value.

For example, creating an integer named x and a floating number y is done as shown in the following commands.

```
>> x=301;
```

```
>>%An integer named x with an initial value of 430 is created.
```

```
>>%Note that comments are specified to MATLAB using % at the beginning of the line
```

```
>> y=0.301;
```

```
>>%A floating number named y with an initial value 0.301 is created.
```

Note the semi-colons at the end of each command. Putting a semi-colon at the end of a statement in MATLAB means that you want the output of the statement not to be shown in the command window. For example, when you do not write the semi-colons of the statements in the command history above the output of MATLAB appears to be as below

```
>> x=301
```

```
x =
```

```
301
```

```
>> y=0.301
```

```
y =
```

```
0.3010
```

```
>>%No semi-colon is used, outputs of the commands are written to the window
```

Creating the arrays and matrices in MATLAB is similar. The following command histories illustrates the generation of arrays and matrices

```
>> a=[3 0 1]
```

```
a =
```

```
3 0 1
```

```
>>%A vector of size 1x3 was generated with name a
```

```
>>%Note that square brackets are used for vector and matrix declaration
```

```
>> b=[3; 0; 1]
b =
     3
     0
     1
>>%A vector of size 3x1 was generated with name b
>>%Note that semi-colons are used instead of spaces. This tells the MATLAB to pass
to the next row of
>>%the matrix
```

```
>> c=[3 0 1; 1 3 0; 0 1 3]
c =
     3     0     1
     1     3     0
     0     1     3
>>%A matrix of size 3x3 was generated with name c
>>%Note the place of the semi-colons which specify the place to jump to the next
row of the matrix
```

Reaching the elements of the arrays and matrices created is done using brackets.

```
>> x1=c(3,1)
x1 =
     0
>>%A variable named x1 is created with its initial value equal to 1st element of the 3rd
row of matrix c
```

- a. Create an identity matrix of size 4x4 with a name MyMatrix in MATLAB.
- b. In MATLAB, the colon ( : ) operator has a special meaning in the indexing of the matrices and vectors. The special meaning assigned to this operator is “ALL”. In this part write and execute the following commands in MATLAB and observe the results to understand what the colon operator does in the commands.

```
>>r1=MyMatrix(2,:)
>>c2=MyMatrix(:,3)
```

- c. Concatenation of MATLAB variables, arrays, and matrices is possible by means of simple commands. In this part you will observe these capabilities. For this purpose, first, create two 1x3 vectors [4 5 6] and [3 2 1] with names v1 and v2 respectively. Then observe the result of the following commands.

```
>>v3=[v1 v2]
>>v4=[v1; v2]
>>v5=[v1 v2; v2 v1]
>>v6=[v1 c]
```

Which commands worked, and which ones did not? What is the reason if any of them did not work?

**Q.2.** Every signal either in time or frequency domain is represented in the computer by a sequence of numbers. Being a popular numerical analysis package, MATLAB has efficient ways to generate such sequences. In this question, you will deal with these generation techniques.

The easiest way of generating a sequence is to use the colon notation of MATLAB. Below is a MATLAB command history to generate an array of integers named x from 1 to 10.

```
>> x=[1:10]
x =
    1    2    3    4    5    6    7    8    9   10
>>%x is an array of integers with an increment 1 between the consecutive elements
```

In the history above, the increment between the consecutive elements of the array is 1 by default. Using the colon notation in a different way, one can also specify the increment between the elements. Below is an example

```
>> x=[1:0.4:3]
x =
    1.0000    1.4000    1.8000    2.2000    2.6000    3.0000
>>%Now x is an array from 1 to 3 with increments of 0.4
```

When the increment (or, in a way, the sampling period) is specified, the colon notation is the best way to create an array in an interval. However, at many times one may need to take a specified number of samples in an interval. For example, if we need to take 7 samples in the interval [1, 3], calculating the required sampling period (i.e. the increment) may be a cumbersome job for most of the time. Therefore, a second way of generating such an array exists in MATLAB. The method uses the built-in MATLAB function `linspace()`.

The usage of the function is given below,

```
x=linspace(Beginning of the Interval, End of the Interval, # of Samples Needed);
```

Below is a sample execution,

```
>> x=linspace(1,3,7)
x =
    1.0000    1.3333    1.6667    2.0000    2.3333    2.6667    3.0000
>>%Now, x contains 7 elements in the interval [1,3]
```

**a.** As a first step of this question, generate an array named as t in MATLAB which represents time in the interval  $[0, 4\pi]$  using both of the methods illustrated above. For the colon notation operation choose the time increment as 0.1 and for the `linspace()` function, take the number of samples as 200. Observe the resulting array values.

**Hint:** In MATLAB, the number  $\pi$  is represented by the built-in variable “pi”.

**b.** Now, execute the following command in MATLAB.

```
>>y=sin(t)
```

This command takes the sine of the array t and equates the resulting array to a new array named y in MATLAB. Observe the resulting values of the array y.

c. Now, plot the resulting sine array y using the following command.

```
>>plot(y)
```

Observe the resulting plot. What do you think the x-axis values of the plot represent?

d. In this part, use the plot() function in the previous part with two arguments as shown below,

```
>>plot(t,y)
```

What is the difference between x-axis of this plot and that of the one in the previous section?

e. In this part, repeat the parts c and d now using the stem(.) function instead of the plot(.) function. What is the difference of the stem(.) function?

f. You should always title and label your plots. Use the following commands to make your plots more informative.

```
>>title('type your text here')
```

```
>>xlabel('type your text here')
```

```
>>ylabel('type your text here')
```

Observe the effects of each command on your plots.

**Q.3.** Indexing of arrays and vectors can be done using arrays in the indices' position. Using the arrays v3 and v5 you have created in question 1, observe the effect of indexing with an array by running the following commands,

```
>>a=v3(2:4)
```

```
>>b=v5(:,3:5)
```

```
>> c=v5(:,1:2:6)
```

Considering the elements of arrays v3, v5, a, b and c, comment on how the indexing is done using an array.

**Q.4.** In this question of the assignment, you will generate more complex functions than the single sinusoid of the previous question.

a. In this part, generate again a time sequence named t using the linspace() function in the interval [1,20] with 10000 samples.

b. Now, using the generated time sequence t, generate and plot the function defined below,

$$y(t) = \exp\left(-\frac{t-1}{10}\right) + \sin(t) \quad \text{for } 1 \leq t \leq 20$$

For this purpose, generate two sequences y1 and y2 as below,

```
>>y1=exp(-(t-1)/10);
```

```
>>y2=sin(t);
```

Then the sequence y can be easily obtained by summing the two sequences y1 and y2 and equating the result to the array y.

c. You are now required to generate and plot the function defined below,

$$y(t) = \exp\left(-\frac{t-1}{10}\right)\sin(t) \quad \text{for } 1 \leq t \leq 20$$

For this purpose, you must multiply the two arrays y1 and y2 generated in the previous part in an element by element manner. Note that since MATLAB always makes matrix operations the command

```
>>y=y1*y2;
```

will not work and return an error. This is because the sizes of the arrays y1 and y2 are incompatible for matrix multiplication. Execute this command and see the error message returned. For this type of element by element matrix operations MATLAB has built-in operators such as “.\*” for element by element multiplication, “./” for element by element division etc. Now try the following command and plot the resulting function.

```
>>y=y1.*y2;
```

**d.** Now, using the information of the previous parts, generate and plot the function given below,

$$y(t) = \sin(500 / t) \quad \text{for } 1 \leq t \leq 20$$

Don't forget to title and label your plot.

## Information About Control Structures and Functions

### For Loop Examples:

```
for i=1:N
```

```
    statements;
```

```
end
```

```
%Equates i to elements of the array [1:N] one by one
```

```
for i=10:0.2:20
```

```
    statements;
```

```
end
```

```
%Equates i to elements of the array [10:0.2:20] one by one
```

```
for i=100:-1:1
```

```
    statements;
```

```
end
```

```
%Equates i to elements of the array [100:-1:1] one by one
```

```
for i=[3 2 1 5 8 56 3 6 8 0 6 4]
```

```
    statements;
```

```
end
```

```
%Equates i to elements of the array [3 2 1 .....0 6 4] one by one
```

### If Statement

```
if <condition>
```

```
    statements;
```

```
elseif (condition) %optional
```

```
    statements;
```

```
else %optional
```

```
    statements;
```

```
end
```

### While Statement

```
while <condition>
```

```
    statements;
```

```
end
```

### User Defined Function Structure

```
function [out1 out2]=FunctionName(inp1, inp2)
```

%Function Name should be the same as the file name in which the function is saved  
*statements*;  
out1=*statements*;  
out2=*statements*;

Example of calling the function above from the command window:

```
>>  
>>[A,B]=FunctionName(inp1,inp2);  
>>%A has now the value of out1  
>>%B has now the value of out2  
>>
```