TBMT 19/33/37 INTRODUCTORY COMPUTER LAB

In this introduction to Matlab, you will create a short script which can calculate the mean and standard deviation of a set of random numbers and compare it to other data. While creating this script, you will learn some basic commands, how to do vector operations and loops as well as how functions in Matlab are written and run.

First, start Matlab.

Suppose you have some data, and you want to analyze it. To be able to use the data in Matlab, you need to create a vector with the information. Create a vector containing the numbers in the table below.

6	3	6	4	8	5	6	9	4	1

Write in the Command Window in Matlab. Vectors are created using square brackets. Label the vector "data1".

```
>> data1 = [6      3      6      4      8      5      6      9      4      1]

data1 =

     6      3      6      4      8      5      6      9      4      1
```

A very useful symbol is the semicolon, which simplifies your work tremendously by suppressing print-out.

```
>> data1 = [6      3      6      4      8      5      6      9      4      1];
```

Ah much better! Whenever you want to see your output, just skip the semicolon.

Think linear algebra: You've now created a 1x10 row vector with the first entry being the number 6 and so on.

To view the vector, simply type its label (without semicolon).

To look selectively at a position in your data1 vector, each entry of data1 can be called specifically.

```
>> data1(1,3)
```

Returns the value 6.

Note that once you created the vector, the label "data1" appeared in the workspace, top right of the Matlab window.

If you prefer to have the data as a column vector, use semicolons anywhere you want to change row. Change this now.

```
>> data1 = [6;     3;     6;     4;     8;     5;     6;     9;     4;     1];
```

But you don't. An easier way to change back is to use the apostrophe to transpose the vector.

```
>> data1 = data1';
```

You want to compare this randomized vector with a vector with number 1 to 10. For repetitive patterns it is convenient to use colons. For step size 1:

```
>> data2 = [1:10];
```

If you had wanted a step size of 0.5:

```
>> data2 = [1:0.5:10];
```

But you don't, so change back.

Now you know how to create the data vectors that will be used in the script you will create. To continue, you need to open a new script file where you can write and save your sequence of commands to be used whenever.

Open a new file. To start fresh every time the script is used, start with commands to clear all information in the workspace.

```
clear all
```

Insert commands to create the two vectors data1 and data2.

Save the script as myScript.m.

Now, you can try running the script. Either type myScript in the Command Window, or drag the file myScript.m from the directory to the left in the Matlab window to the Command Window. Note that data1 and data2 appear in the Workspace.

It may be better to assign the data to one single matrix, a 2 by 10 matrix where one row corresponds to one data set. This can be done with the information you already have learnt.

```
data = [data1; data2];
```

Next, you want to create a function to which you can send your two data sets, and which returns means and standard deviations from your data sets.

Open a new file and save it as myFunction.m.

The difference between a script and a function is simply that a script is only a series of commands while a function has inputs and outputs.

The first row in a function contains information of what outputs and inputs the function has. End the function by writing "end".

```
function [output] = myFunction(input)
…
end
```

Remember that the input now is a matrix where every row is a data set. In a similar manner, you want to create an output matrix with the means in the first column and the standard deviations in the second. To create a matrix of zeros, in which the output can be stored, define the matrix "out" as a 2 by 2 matrix. The Matlab function "zeros" is convenient here.

```
output = zeros(2,2);
```

This created a 2 by 2 matrix where all entries are zeros.

Now, you have two choices. Either, you do the same thing twice to get both means and both standard deviations. However, this becomes terribly tiresome if the number of data sets increases. Instead, you remember Ada and all the smart loops you created there!

```
for i = 1:2
…
end
```

In this loop, you want to calculate and save the mean and the standard deviation for data set *i*, where *i* goes from 1 to 2.

Now, use the Matlab functions “mean” and “std” to get the information you need! To get information about these functions, or any other function, simply write “help function”.

```
output(i,1) = mean(input(i,:));
output(i,2) = std(input(i,:));
```

To use this function, write in your script file ”x=myFunction(data)”.

Now, try the script! Checkpoint

To look selectively at your output, now stored in the matrix *x*, each entry of x can be called specifically.

```
>> x(2,:)
```

Shows the second row only, which corresponds to what?

Finally, you need one last touchup to your script. What if you want to send 4, 10 or 87 data sets to your function? Create a new data matrix with the function *randi*.

```
data = randi(10,[100,10]);
```

Here, the first number is the maximum integer size which can be randomized, and the second the size of the matrix you are creating. Thus, data now contains 100 data sets with 10 data points each.

Now, your functions must be changed to account for this!

```
[m,n] = size(input);
```

This function saves the number of rows (m) in the input as well as the number of data points in each set (n). Put this in the beginning of your function, and exchange the numbers for m and n where needed (should be 2 places total).

Try the script again!

What’s the mean of the means?