Workshop 4 - More with Loops, Vectors and 2D Arrays

- Due 16 Aug by 23:59
- Points 1

Workshop 4 - Working with loops in MATLAB

This workshop is designed to provide more practice with using loops both nested and un-nested in MATLAB and working through some key concepts from this week's video content. To prepare for this workshop you should create a folder in your account or on your computer called **workshop4**.

Each question has the pattern of problem definition, followed by tests, followed by coding. The Demonstrators will work through questions 1 and 2 with you and for the remaining questions you should be able to use code and ideas from previous questions in later questions.

During the demonstration, if possible it is good to type along with the demontrator or take notes *(bring pen and paper!)*. If you are sharing a terminal make sure that take turns at working during this practical so every group member has a chance to save some working code.

When you have finished your workshop you should make sure that you show the work that you have done to the laboratory supervisor so your participation can be marked. Be sure to check your participation is recorded before you leave your workshop session or you won't receive credit.

Coding

To start this workshop navigate to your workshop 4 folder in MATLAB and type

```
clear all
```

into the MATLAB console. This is a way to clear previous definitions that might get in the way of code working as required. In general, it is good to type this command at the beginning of a coding session and before you re-run your script.

Question 1: Calculating total element distances (to be done with demonstrators)

Definition

For this question we define a vector A of numbers and calculate another vector B that contains the sum of the distances from each element to every other element.

Testing

These are some example test cases :

```
A          B
[ ]        [ ]
[ 5 ]      [ 0 ]
[ 1 3 ]    [ 2  2 ]
[ 1 3 5 ]  [ 6  4  6 ]
```

(we would save these in a file called q1.txt)

Coding

One solution for this program is the script

```
% calculate the sum of distances from each element to
% every other element.

% define A
A = [1 3 5];

% initialisation
B = []; % start with empty B
lenA=length(A); % record length of A

% outer loop - to add each distance sum to B
for i = [1:lenA]
    sumDist = 0; % initialise sum of distances
    % inner loop - sum differences of elements
    % from current element.
    for j = [1:lenA]
        % add distance from current element A(i)
        % to other element A(j) to sum
        sumDist = sumDist + abs(A(i)-A(j));
    end

    % append sumDist to the result
    B = [B sumDist];
end

% display result
disp(B);
```

(we would save this script in a file called q1.m). Note the use of abs() to help calculate the absolute values of the distances.

Question 2: Right-shifting a vector

Definition

In this question we define a vector A and create a new vector B containing the elements of A shifted one index to the right. The first element of B should be zero. Thus A and B will be of the same length.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
A           B
[ 1 2 3 4 ]  [ 0 1 2 3 ]
[ 5 3 8 ]    [ 0 5 3 ]
```

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q2.m that implements the definition above using a for-loop. **Hint:** your for-loop will need to iterate over all but the last element of A to copy them across to B.

Question 3: Right-rotating a vector

Definition

In this question we define a vector A and create a new vector B containing the elements of A shifted one index to the right. The first element of B should be the last element of A. Thus A and B will be of the same length.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
A           B
[ 1 2 3 4 ] [ 4 1 2 3 ]
[ 5 3 8 ]   [ 8 5 3 ]
```

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q3.m that implements the definition above using a for-loop. **Hint:** your solution to question 4 above could be a good starting point to answer this question.

Question 4: Left-rotating a vector

Definition

In this question we define a vector A and create a new vector B containing the elements of A shifted one index to the *left*. The last element of B should be the previous first element of A. Thus A and B will be of the same length.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
A           B
[ 1 2 3 4 ] [ 2 3 4 1 ]
[ 5 3 8 ]   [ 3 8 5 ]
```

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q4.m that implements the definition above using a for-loop. **Hint**: You can copy A to B starting with the second element of A.

Question 5: Summing neighbouring elements in a 2D array

Definition

In this question we define a 2D array A and create a new 2D array B where each element of B is a sum of the corresponding element of A and its neighbouring elements. As special cases the first element of B is the sum of the first two elements of A and the last element of B is the sum of the last two elements of A. Again A and B will have the same dimensions (same number of rows and columns). You can assume that A has at least two elements.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
test 1
A           expected B
1 2 3 4     3  6 9  7
5 3 8 6     8 16 17 14

test 2
A           expected B
0 5 6       5 11 11
1 3 2       4  6  5
8 9 10      17 27 19
```

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q5.m that implements the definition above using a nested for-loop. **Hint**: your for-loop might only iterate across the middle elements of A - you can handle the first and last elements separately.

Question 6: Calculating a 2D array of the element distances

Definition

In this question we define a vector A and we use it to create a 2D vector B where each row contains the distances from the corresponding element of A to the other elements of A. Thus row 1 of B will contain the distances from the first element of A to the other elements of A and row 2 of B will contain the distances from the second element of A to all other elements and so on.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be:

```
A           B
1 2 3 4     0 1 2 3
            1 0 1 2
            2 1 0 1
            3 2 1 0

7 5 8       0 2 1
            2 0 3
            1 3 0
```

Note that we have left the brackets out of the examples above to make it easier to write.

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q6.m that implements the program defined in the definition above. **Hint**: the solution to **question 1** above is a good starting point for this question. Also note that this question will require you to vertically contentate vectors using the ":" (semicolon operator). As an example of this operator's use, the code:

```
A = [ 1 2 3 ];
B = [ 4 5 6 ];
C = [A; B];  % vertically concatenate A and B
disp(C);
```

will display the 2D vector:

```
1 2 3
4 5 6
```

Finishing up.

Make sure you let a practical supervisor see your work for this session.

End of Workshop.