Practical 3 - Working with Vectors

- Due 12 Aug by 23:59
- Points 16
- Submitting an external tool

This practical exercise is to be prepared for in your own time and in your computer exercise class.

These questions are intended to be completed **individually** but you are allowed to consult other resources to answer generic questions that help you complete small parts of each question.

Please consult the **rubric** of this practical assignment for information on the marking scheme.

For this exercise you must:

1. Complete the exercises below. Do what you can prior to class and come into class to have questions answered.
2. Prior to the due-date submit your answers (MATLAB scripts and text files) via this assignment link in Gradescope.
3. You can choose to have your tutor mark you in person (recommended for feedback) in your practical or workshop session

This week's topic

This week's topic is working with vectors. The exercises below require you to transform vectors, to filter vectors, and to work with vector indexing to re-arrange vector elements. Before you start this week's exercise you should make a folder called:

```
week3practice
```

to put all of the scripts and testing files described below.

Question 1.

For this problem, you have to create a program that creates transformed versions of existing vectors and plots the resulting vectors.

To start this exercise start a MATLAB script called q1.m and put, at the top, the vector definition:

```
A = [-10:0.5:10];
```

in this question, we will create code to create new vectors which are transformed versions of this vector.

Be sure to use this vector declaration when you submit your code.

Question 1a.

Problem Definition

For this question start by saving q1.m as q1a.m. In this script you will implement MATLAB code that uses a for-loop to create a new vector B that contains the cubes of the numbers in A. Note that the cube of any number is that number multiplied by itself three times. Thus the cube of 2 is 2x2x2=8 and the cube of 3 is 3x3x3=27. Note also, your program has to work for any contents of A in the range -100 to 100 (not just the numbers currently in A).

Testing

For this question you should, in a text file called, q1a.txt write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original        Expected Result   Actual Result
A               B
[1 2 3]         [1 8 27]          [1 8 27]
[-1 0 1 100]    [-1 0 1 1000000]  [-1 0 1 1000000]
```

Remember to write your test cases in the format above into your **q1a.txt** file.

Coding

Now code your solution to the problem above in your q1a.m file. Make sure you include some code at the end to display the values of the vector B **with 4 digits to the right of the decimal point. Please print the values on one line with a space between the values.**

Question 1b.

Problem definition

For this question start by saving q1a.m into q1b.m. Modify the code in your script so that you create another vector C that contains 2 raised to the power of each element of A. Note, that you can raise 2 to any value in MATLAB by using the pow2 function. So, for example, pow2(3) will produce the value 8. Again, your program should work for any values of A in the range -100 to 100 not just the current values of A.

Testing

For this question you should, in a text file called q1b.txt , write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original     Expected Result     Actual Result
A            C
[2 4]        [4 16]              [4 16]
[-1 0 1]     [0.5 1 2]           [0.5 1 2]
```

Coding

Now code your solution to the problem above in your q1b.m file. Make sure you include some code at the end to display the vector C **with 4 digits to the right of the decimal point**. **Please**

print the values on one line with a space between the values. In your coding, consider whether you can use the same for-loop to produce the vector C or whether you have to use a second for-loop.

Question 1c.

Problem definition

For this question start by saving your q1b.m file into a new script called q1c.m. Modify that code in your script so that you use MATLAB's plot command to plot both B and C on the same graph. Note, that some of the concepts needed to do this are new so see the coding section below to guide you.

Testing

The results of this problem are graphs. On paper, or in words, describe the expected shape of the graphs. If you use words save them in a file called q1c.txt.

Coding

In a video lecture this week you have seen some simple examples of plotting. For example we could plot vector B simply by typing

```
plot(B);
```

into MATLAB. However, if we then type:

```
plot(C);
```

the old plot of B will disappear. To plot both on the same graph we need to type:

```
close all;  %close all figure windows
hold on; % display plots on the same window for now
plot(B);
plot(C);
hold off;  % turn off plotting on the same window.
```

Look at the figures displayed. You might notice that the x-axis below might not have the correct values. If you want to use the values from A for the x-axis you can add the x-axis values as a parameter to plot. For example we might say:

```
plot(A,B);
```

to plot the values of B on an x-axis covered by A. Note, for this to work well you would need to have the elements of A in ascending order. Try plotting this way for your code above and save in q1c.m.

Question 2.

For this problem you have to create a program that defines two vectors of numbers: A and B and uses a for-loop to combine the values of A and B. Note that different parts of the question

below specify different orders for combining the elements.

To start this exercise start a MATLAB script called q2.m and put, at the top, the vector definition:

```
A = [1:10]
B = [11:21]
```

Question 2a:

Problem definition

Copy the file q2.m to q2a.m. For this problem you have to modify the code in your file so that, using a for-loop, you add the elements of B onto the end of A one element at a time.

Testing

For this question you should, in a text file called, q2a.txt write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original                Expected Result     Actual Result
A        B           new A
[1 2 4]  [5 6 7]     [1 2 4 5 6 7]       [1 2 4 5 6 7]
[-1 0 2] [7 3 1 4]   [-1 0 2 7 3 1 4]    [-1 0 2 7 3 1 4]
```

Coding

Now code your solution to the problem above in your q2a.m file. Make sure you include some code at the end to display the vector A after the loop has finished running. **Display the vector values as integers** - just the whole number (no decimal point or values after the decimal). **Please print the values on one line with a space between the values.**

Change your definitions of A and B at the start of your script to test it is working for your test cases.

Before submitting, change the declaration of A and B to:

```
A = [1:10]
B = [11:21]
```

Question 2b:

Problem definition

Copy the file q2a.m to q2b.m. For this problem, you have to modify the code in your file so that, using a for-loop, you add the elements of B onto the end of A one element at a time **in reverse order**.

Testing

For this question you should, in a text file called, q2b.txt write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original           Expected Result      Actual Result
A        B          new A
[1 2 4]  [5 6 7]    [1 2 4 7 6 5]        [1 2 4 7 6 5]
[-1 0 2] [7 3 1 4]  [-1 0 2 4 1 3 7]     [-1 0 2 4 1 3 7]
```

Coding

Now code your solution to the problem above in your q2b.m file. Make sure you include some code at the end to display the vector A after the loop has finished running. **Display the vector values as integers** - just the whole number (no decimal point or values after the decimal). **Please print the values on one line with a space between the values.**

As before, change your definitions of A and B at the start of your script to test it is working for your test cases.

Before submitting, change the declaration of A and B to:

```
A = [1:10]
B = [11:21]
```

Question 2c:

Problem definition

Copy the file q2b.m to q2c.m. For this problem you have to modify the code in your file so that, using a for-loop, you **interleave** the elements of A and B creating a new vector called C. You can assume that A and B are the **same** length.

Testing

For this question you should, in a text file called, q2c.txt write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original           Expected Result    Actual Result
A        B          C
[1 2 4]  [5 6 7]   [1 5 2 6 4 7]      [1 5 2 6 4 7]
[-1 0 2] [7 3 1]   [-1 7 0 3 2 1]     [-1 7 0 3 2 1]
```

Coding

Now code your solution to the problem above in your q2c.m file. Make sure you include some code at the end to display the vector C after the loop has finished running. **Display the vector values as integers** - just the whole number (no decimal point or values after the decimal). **Please print the values on one line with a space between the values.**

Change your definitions of A and B at the start of your script to test it is working for your test cases.

Before submitting, change the declaration of A and B to:

```
A = [1:10]
B = [11:20]
```

Question 3.

For this problem, you have to write a program that filters the leap years from a vector of years. The following parts will build up a solution one part at a time.

Question 3a.

Problem definition

For this problem, you have to write a MATLAB program that reads a number as input and prints out 'Divisible by four' if that number is divisible by four. To do this you will need to use an if-statement (you won't need a for-loop yet).

As an example of the pattern of computation that might be useful the following MATLAB code:

```
% get myAge from user input

% check if my age is greater than 40
if(myAge>40)
  disp('You are older than 40');
end
```

Will display 'You are older than 40' if myAge is greater than 40 (which it is in the code above).

Likewise the code:

```
% get myNumber from user input

% check if my number is odd
if(mod(myNumber,2) ~= 0)
   disp('Your number is odd');
end
```

Will display 'Your number is odd' if myNumber **is not** divisible by 2.

Note that the test used in this code:

```
mod(myNumber,2) ~= 0
```

is true if the remainder from dividing myNumber by 2 (mod(myNumber,2)) **is not** equal (~=) to zero.

Note also that, if I wanted to test if myNumber **is** divisible by two, I could write:

```
mod(myNumber,2) == 0
```

Testing

For this question you should, in a text file called, q3a.txt write a table containing three new test cases. As a reference example, three possible test cases for this are:

```
Input   Expected Result      Actual Result
3       no output
4       'Divisible by four'  'Divisible by four'
0       'Divisible by four'  'Divisible by four'
```

Coding

Now code your solution to the problem above in a file called q3a.m. Again, look at the code examples above as a guide to how you might write your if-statement and the tests you need.

Question 3b.

Problem definition

For this problem you need to write a MATLAB script, called q3b.m, that starts with the code:

```
years=[2015:2041];
```

and creates a new vector called leapYears that contains all of the years in *years* that are divisible by four.

Be sure to use this vector declaration for your submitted code.

Testing

For this question you should, in a text file called, q3b.txt write a table containing three new test cases. As a reference example, two possible test cases for this are:

```
Original                    Expected Result     Actual Result
years                       leapYears
[2015 2016 2017 2018 2019]   [2016]             [2016]
[2012 2013 2014 2015 2016]   [2012 2016]        [2012 2016]
```

Coding

Now code your solution to the problem above, using a for-loop, in a file called q3b.m. Use your code from q3a.m above to help with selecting the right elements. Note, that you may also find the examples in the filtering video from this week useful. **Please print the leap years on one line with a space between the values.**

End of Questions.

Assesment

Marks are awarded for

- **functionality** (the code runs and produces expected results) - 8 marks
  - NOTE - marks for q1.c will be added manually as tutor must check this. These marks will be added under the testing rubric in Gradescope.
- **style** 4 marks (the code adheres to **MATLAB Style Guidelines (https://myuni.adelaide.edu.au/courses/95033/files/15473409?wrap=1) (https://myuni.adelaide.edu.au/courses/95033/files/15473409/download)** for this practical we will be assessing adherence to:
  - **Naming Conventions**: for variables (all sections **other than** Structures, Functions.
  - **Statements**: sections Variables and Constants and Loops,

- **General**: sections *before* "Use the natural, straightforward form for logical expressions".
- **Layout, Comments and Documentation** : all aspects other than those that mention *functions*

- **testing** for this practical we are looking for the presence of unique tests cases (3 per question) - 4 marks

This tool needs to be loaded in a new browser window

Load Practical 3 - Working with Vectors in a new window