Faculty of Engineering and Information Science
ENGG100 Engineering Computing and Analysis
Tutorial problem sheet

UNIVERSITY OF WOLLONGONG IN DUBAI

Tutorial 4 – Week 4

Aims:

Upon successfully completing these tutorial exercises, students should be able to:

- Create and modify numerical arrays.
- Create and modify cell arrays.
- Perform arithmetic functions with vectors.

In these early tutorials, it may not seem like what you are learning is particularly useful. However, it is essential you learning the fundamentals first before you attempt more advanced programs (which you will able to do soon enough if you keep up!). Also keep in mind that most of the concepts learnt in MATLAB script are common amongst most programming languages. Once you understand one language, it is much easier to learn others.

1 Create and modify numerical arrays.

The most direct way to create a numerical array is by using the square brackets []. Last tutorial we used the square brackets to concatenate both strings and numbers. Here we will just look at numbers. If we want to create a one dimensional array with five values 1, 3, -2, 0 and 8 and store the array in a variable x:

```
x=[1,3,-2,0,8];
```

If you want to access an element of a numerical array, for example if you want to access the second element of x:

```
x(2)
```

If you want to delete the second element of x:

```
x(2)=[];
```

Remember if you want to see what is stored in x, simply enter x in the command window and press enter.

Handy hint: entering clc into the command window will clear the command window, however the variables will remain. You can see what variables exist by looking in the workspace or by typing who.

If you wish to create an array with an incremental relationship, for example an array containing all natural numbers from 1 to 8:

```
x=1:8;
```

If the increment size is not one, then another colon is required. For example, if you wish to begin at 5 and move in steps of 0.1 to 7:

```
x=5:0.1:7;
```

Negative increments are also possible:

```
x=10:-1:2;
```

Transposition of arrays may be achieved using the inverted comma ‘.

```
y=x’;
```

It is possible to access or delete any subset of an array. Try to predict the effect of the following:

```
a=x([2,5,6]);
b=x(1:3);
x(5:end)=[];
```

It is also possible to create multidimensional arrays. For example, to create a two dimensional array:

```
x=[1,2,3;4,5,6];
```

The comma (or a space) indicates a new column and a semicolon indicates a new row.

To access an element from a two dimensional array, use the row then column as an index inside parentheses. For example, if you want to change the element in the first row and second column to the number 7:

```
x(1,2)=7;
```

For you to do:
In a new script, save an array of 10 grades into a variable (make up the numbers yourself). Then request from the user the student number (between 1 and 10) and return the grade of that student. For example, let grades = [70,65,55,50,47,82,83,90,50,65]. If the user selects student 3, then the program should display, “Student 3’s grade is 55.”

Handy hint: If you want to display an inverter comma (apostrophe) to the user, you will have to use two inverted commas (not double inverted commas). Recall that one inverted comma represents the beginning/ending of a string. For example: disp('It''s done like this');

2 Create and modify cell arrays.

A cell array is an array which can contain different types of data (numerical arrays have only numbers, string arrays have only characters). When creating and accessing cell arrays, the braces {} should always be used. Let’s say you have five students and you want to save their names in the first column (string type) and their grades in the second column (numerical type) in the second column of the same cell array. Let’s call the cell array gradebook.

```
gradebook={'Jimmy',70;'Carl',64;'Ahmad',72;'Fadima',75;'Chris',58
};
```

After remarking Chris’ exam, we realise that two marks are missing. To increase his mark to 60:

```
gradebook{5,2}=gradebook{5,2}+2;
```

Be careful with the use of brackets in MATLAB, they are all used for different purposes. Braces {} are reserved for cell type arrays. Parentheses () are used with functions, accessing elements of non-cell arrays and for normal mathematical operations. Square brackets [] are used for concatenation (which means joining together).

For you to do:

Update Jimmy's name to James.

3 Perform arithmetic functions with vectors

Vectors are 1D numerical arrays within the context of MATLAB. Your knowledge of operations of vectors will depend on your Maths background. You will need a little for ENGG100, but not much.

To add two matrices together, simply add the corresponding elements together:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}_{b} + \begin{bmatrix}7 & 4\\ -1 & 0\end{bmatrix} = \begin{bmatrix}8 & 6\\ 2 & 4\end{bmatrix}_{a}$$

The same operation can be performed in MATLAB by:

```
a=[1,2;3,4]+[7,4;-1,0];
```

Matrix addition and subtraction works the same way:

```
b=a-[7,4;-1,0];
```

Should give `b=[1,2;3,4]`.

Multiplication is more complicated:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \times \begin{bmatrix}7 & 4\\ -1 & 0\end{bmatrix} = \begin{bmatrix}5 & 4\\ 17 & 12\end{bmatrix}$$

Notice how the row values in the first matrix are multiplied by the column values in the second matrix, and the results are then added. If this doesn't make sense, it is not essential for ENGG100 (although you will need to know this for future subjects). The result can be evaluated in MATLAB by:

```
a=[1,2;3,4]*[7,4;-1,0];
```

If instead you require an element wise multiplication where each element is the multiple of the corresponding elements:

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}.\times\begin{bmatrix}7 & 4\\-1 & 0\end{bmatrix}=\begin{bmatrix}1\times 7 & 2\times 4\\3\times(-3) & 4\times 0\end{bmatrix}=\begin{bmatrix}7 & 8\\-4 & 0\end{bmatrix}$$

For any element-wise multiplication, division or power operation, a full stop must be inserted before the normal operand, in this case:

```
a=[1,2;3,4].*[7,4;-1,0];
```

Try typing:

```
a^2
a.^2
a/a
a./a
```

Notice how the results are different for each. In the last case, one element of your Matrix should contain NaN, meaning not a number. This results occurs because 0/0 has no solution.

4 For you to do:
Create an array containing the grades of 10 students:

```
grades = [50,80,50,72,72,50,50,80,72,80];
```

Create a second array with scaling factors due to the fair contribution of each student.

```
FCS = [1.1, 1, 0.9, 1, 1.1, 0.9, 1, 1.1, 0.9, 1];
```

Multiply the grades by the FCS scaling factors to determine each student's final grade.
The faculty says that the scaling factors should go further to separate student's marks. They recommend that the scaling factor should first be squared and then multiplied by the students' original mark. Determine the new final grades.