Tutorial 9 – Week 10

Aims:

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

- Use vectorisation for array operations.
- Use break and continue inside loops.

1 Use vectorisation for array operations

When using logical operations, we have typically performed the operation of single values rather than arrays. However, within MATLAB it is possible to perform logical operation on arrays using vectorisation. For example:

```
r=1:5
r <= 3
```

Will return an array of binary values, a one will indicate that the expression was TRUE for this element and zero means FALSE. This technique will work when comparing a single value with an array, or comparing two arrays of equal sizes (but don't have two arrays of different size).

```
a=1:5;
b=[0,2,3,5,6];
a==b
```

This technique is useful when trying to remove or alter values when a certain condition is met. For example, what if you want to plot a sine wave, but you want all values below the axis to be set to 0.

```
x=0:0.001:4*pi;
y=sin(x).*(sin(x)>=0);
plot(x,y)
```

Here, if sin(x)>=0 is true, the element will be multiplied by 1 (therefore not changed). Otherwise, the element will be multiplied by 0 and therefore become 0 as desired.

Let's say we wish to calculate the amount of tax an Australian would pay in a financial for a certain amount of taxable income.

Taxable Income	Tax on this income
Up to $18200	0%

Income between \$18201 and \$37000	19% of each dollar over \$18200
Income between \$37001 and \$87000	\$3572 plus 32.5% of each dollar over 37000
Income between \$87001 and \$180000	\$19822 plus 37% of each dollar over 87000
Over \$180001	\$54232 plus 45% of each dollar over 180000

For example, if someone earned \$50000 in one financial year, they would pay:

3572+0.325*(50000-37000)

The function can be create using conditional statements:

```
function tax=aus_tax(income)
if income < 18200
    tax=0;
elseif income < 37000
    tax=0.19*(income-18200);
elseif income < 87000
    tax=3572+0.325*(income-37000);
elseif income < 180000
    tax=19822+0.37*(income-87000);
else
    tax=54232+0.45*(income-180000);
end
```

However, the same thing can be achieved using vectorisation. The vectorised version allows you to calculate many results at once. Try the following with a vector for the input:

```
function tax=aus_tax(income)

tax=0.19*(income-18200).*(income > 18200 & income <= 37000);
tax=tax+(3572+0.325*(income-37000)).*(income > 37000 & income <= 87000);
tax=tax+(19822+0.37*(income-87000)).*(income > 87000 & income <=
180000);
tax=tax+(54232+0.45*(income-180000)).*(income > 180000);
```

1.1 For you to do:

Create a function which calculates an electricity bill based on usage.

- If less than 500 KWh are used, the cost is 2 fils per KWh.
- If between 500 and 1000 kWh are used, the cost is 10 dirhams plus 5 fils for every kWh over 500.
- If more than 1000 kWh are used, the cost is 35 dirhams plus 10 fils for every kWh over 1000.
- In addition, there is a flat connection fee of 5 dirhams regardless of usage.

2 Use break and continue inside loops

We briefly looked at using break and continue statements inside loops (in Tutorial 5). Break may be used to terminate a loop, meaning when a break statement is executed in a loop, the program immediately moves to the end of the loop. Continue may be used to prematurely end an iteration of a loop and move to the next iteration. When a continue statement is executed in a loop, the program skips all remaining lines inside the loop and move to the start of the next iteration.

Let's write a function which checks if any two elements in a 1D numerical array are equal. If any elements are the same, the function should return 1, otherwise 0.

```
function a=same_found(b)
a=0;
for i=1:numel(b)
    for j=1:numel(b)
        if i == j
            continue
        end
        if b(i) == b(j)
            a=1;
            break
        end
    end
    if a == 1
        break
    end
end
```

The function numel() returns the number of elements in b, hence the function will work for an 1D numerical array of any length. Nested for loops are used to create indices for the two elements which are being compared. We do not want the same element to be compared with itself, hence we use the continue statement when i and j are the same. If i and j are different, the two elements of b are compared. If they are found to be the same, set a to 1 and we can break the loop (because we only need to find one match to know the output should be 1). The break statement will only interrupt the innermost loop. Hence, a second break is written for the outer loop that with be run when a has been set to 1.

A good programmer will make sure that his code will continue to function correctly even if the wrong information is entered by the user. For example, if we create a list of options for the user:

```
opt=input('1) Start a new game\n2) Load a saved game\n3) Quit\n');
```

If the user correctly enters 1, 2 or 3 then there is no problem. However, what if the user enters 4? Or anything else? We should then let the user know that their entry is invalid and ask again:

```
while 1
    opt=input('1) Start a new game\n2) Load a saved game\n3)
Quit\n','s');
    opt=str2num(opt);
    if numel(opt) ~= 1 || ~any(opt == [1,2,3])
        disp('Invalid entry.');
    else
        break
    end
end
```

Including a second parameter in the input() function 's' indicates that the data input with be stored in opt as a string type. This allows the user to input non-numerical data without creating an error. The string is then converted into a numerical type using str2num().

The conditional statement then considers an entry invalid if there isn't exactly one element in opt or if opt is not equal to either 1, 2 or 3.

The function any() will return one if at least one TRUE is found in the input. This code will only break the loop if a valid entry is added, other the loop will continue forever.

2.1 For you to do:

Write a script which randomly chooses two numbers between 1 and 100.

- Both numbers must be no more than 10 and no less than 3 apart from one another (i.e 92 and 95 is okay, but 50 and 65 is not).
- The user wins the game by guessing a number in between the two numbers.
- The user loses if one of the two numbers is guessed exactly. If the user is outside of range, the user may have another guess.
- The function randi(100) may be used to randomly generate a number between 1 and 100.
- If the user is out of range, display that the guess is too high or too low and ask for a new guess.
- If the user wins or loses, write an appropriate message and end the script.