Faculty of Engineering
MM1CPM

Worksheet 2

Q1 Complex numbers

This question is designed to help you practice using complex numbers in MATLAB. Understanding how to use complex numbers in MATLAB will enable you to check the answer to your complex number problems in your mathematics module (HG1M11).

Initialize the following complex variables in the work space:

a) Calculate the root of -1 in MATLAB using the **sqrt** function and store the result in a variable called a.

b) $b = 4 + 1i$

c) $c = 1000 + 1000\,i$

d) $d = (1+i)(1+i)$

e) $e = \dfrac{1+i}{100+77\,i}$

f) $f = (10+i)^2$

g) $d = \dfrac{(1+i)(1+i)(2+2\,i)(4+4\,i)}{100\,i}$

h) The built in functions **real()** and **imag( )** will return the real and imaginary part of any complex number. Extract the real and imaginary parts of the variables a,b and c.

i) The built in function **abs( )** will give you the absolute value of a complex number. Extract the absolute value of the variables a,b and c.

Q2 Making scripts

One of the hand outs in the lecture is entitled 'Making and saving MATLAB script files', this is a step by step guide on how to make MATLAB scripts. A script is another word for a computer program. Follow the instructions and make a script called 'math.m'. Write your name and e-mail address on the first line of 'math.m'. At the beginning of the line place a '%' sign this tells MATLAB that anything on this line is not MATLAB code, and to ignore it. Your script should look like figure 2. Notice how the text turns green to tell you that the line represents human readable text, not MATLAB code.

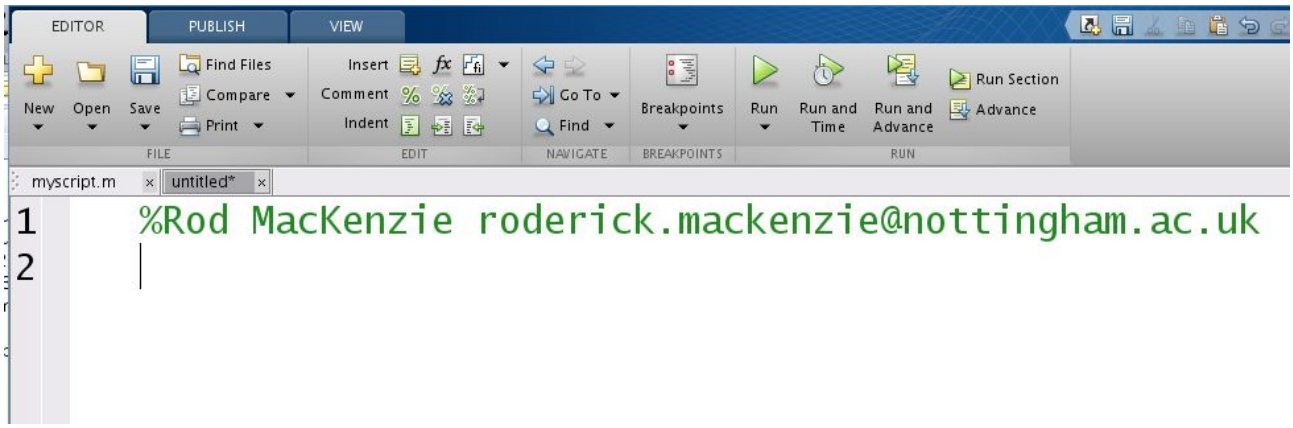

Figure 1: Your first script file

a) In the script file you just made, on line two type the word 'clear'. This will tell MATLAB to forget about any variables you previously defined. It is good practice to clear all old variables at the beginning of a script to so you are not left confused by which variables came from your script and which were previously defined – it also saves memory. On the third line define the variable 'a' as the number one, on line four defines the variable 'b' as the number two, and on line five define the variable 'c' as the number ten. On the final line of the script file define the variable 'z' as the sum of 'a', 'b', and 'c'. Now click on the 'run' arrow to run the script.

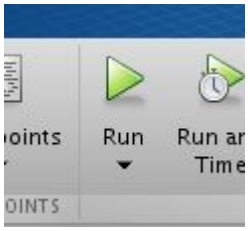

Figure 2: The run button, used to run the script.

If you look at the main 'MATLAB' window, you should see the value of z has been calculated. What's it's value? Another name for a script is a program. Congratulations, on writing your first full computer program!

b) A very important thing to remember about scripts is that the computer executes each line in the script one at a time, starting at the top and working it's way down to the bottom. With this in mind, now sketch out the flow diagram for your script you wrote above, each line in the program (except the line with your name) should be represented by it's own square box. Why would the script not work if the line z=a+b+c was at the top of the script?

Q3: Small arrays

Make a new MATLAB script file and call it Q3.m. Again, put your name and e-mail address in it. Putting your name in code is good practice so that people know who the copyright belongs to – this is important in industry. In the script, using the square bracket notation, define the following arrays one after each other on five different lines. You can find examples of how to do this in your lecture notes. After you have entered each line, press the 'run' button to check the script works. You should be able to see the output from the script in the main MATLAB command window.

a) $a=[1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10]$

b) $b=[-\pi \quad 0 \quad \pi]$

c) $c=[10 \quad 0 \quad 10 \quad 0 \quad 10 \quad 0 \quad 10 \quad 0 \quad 10 \quad 0]$

d) $d=[-\pi/2 \quad -\pi/4 \quad 0 \quad \pi/4 \quad \pi/2]$

e) $e=[-\sqrt{8} \quad -\sqrt{4} \quad 0 \quad \sqrt{4} \quad \sqrt{8}]$

f) $f=[\pi \quad 0 \quad -\pi]$

Q4: Bigger arrays
Make a new script file called 'Q4.m', again put your name and e-mail in the top of the script and the command clear on the second line. Typing arrays in by hand is fine if if they are small, but often in Science and Engineering the data sets you have to deal with run to thousands or billions of numbers. Use the linspace command (covered in the lecture – you can also find it in the help) to define the following arrays:

a) $a=[0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10]$

b) $b=[0 \ 0.5 \ 1 \ 1.5 \ 2 \ 2.5 \ 3 \ 3.5 \ 4 \ 4.5 \ 5 \ 5.5 \ 6 \ 6.5 \ 7 \ 7.5 \ 8 \ 8.5 \ 9 \ 9.5 \ 10]$

c) An array from 0 to 200 with 100 points.

d) An array from -200 to 200 with 100 points.

e) An array from -1000 to 1000 with 100 points.

f) An array from 0 to 1000 with 10000000 points.

Test your script after each new line you have added. Testing your script as you write it is also good practice, and will help you find any mistakes you have made.

Depending upon the speed of your computer, the last example (d), may take a very long time to finish printing to the screen. If you don't want to wait, press 'Crtl and c' together to stop the program. By default MATLAB prints the result of every command to the screen. Often you have so much data that it makes no sense to print it out. Try adding a ';' to the end of each line in the script and run it again. The semicolon stops MATLAB printing out the result of each mathematical operation. If you want to see the contents of the array just type the name of the array in the command line. Try typing 'a <enter>' in the command line in the main MATLAB window.

Q5: Mathematics on arrays
One of the very powerful features of MATLAB is that if arrays are the same size their values can be added or subtracted. This questions shows you how to do simple mathematical operations on arrays. Copy Q5.m to a new file called Q5.m. To make it easier to see what is happening place a semicolon ';' at the end of each line, this will reduce the number of numbers MATLAB prints out.

a) At the bottom of the script define a new variable 'g' which is the sum of 'a' and 'c' (i.e. write g=a+c). What has MATLAB done to form the new array?

b) Try performing g=b+f, and g=b-f, what happens?

c) Another very powerful feature of MATLAB is that you can multiply an entire array by a number or divide a whole array by a number. Try adding the line g=c*10 and g=c/10.0 to the bottom of the script. What happens?

We will be using the ability to multiply entire the arrays by numbers in later examples.

Q6: Plotting
Make a new MATLAB script called 'Q6.m', again put your name and e-mail address in the script,and the command clear as the second line. In this script define an array x from -2π to 2π with 10 points.

a) Generate a new array 'y' that is the sin of array x. You can copy the example done in the lecture.

b) Now calculate a new array 'c' that is the cosign of array x.

c) Using the plot command plot the array y.

d) Why does the plot of 'y' not look like a perfect sin wave?

e) Increase the number of points used to generate the x-axis from 10 to 100 and re-plot the graph.

f) Plot the array that contains the cos of x.

g) Add the array c to the array y forming a new array g.

h) For array 'g', calculate the maximum value, the minimum value, the standard deviation of the data, and the mean value. Why is the maximum of array g bigger than 1.0?

Practicing simple array manipulation
Q7: Copy the file Q6.m to a new file Q7.m, and open Q7.m using MATLAB. In the lecture we learnt how to extract a subset of data from an array. Look at your notes and revise this, then:
a) Use the command z=y(1:25) to extract the first 25 points of array y forming a new array z.

b) Plot the new array z.

c) Try changing the number 25 to 50 what happens. What happens if it gets bigger than 100, why is this? Try varying the number 1, what happens to the extracted data? What happens if you choose a number smaller than 1?

d) Define a new array 'n', which is generated from the the 15^{th} to the 35^{th} point of array y

e) Plot array n, what does it look like?

f) In the lecture we learn how to join arrays together forming new arrays. Define a new array 'nn'

which contains two copies of n, by adding the command "k=[n n]" to the end of your script.

h) Change the line "k=[n n]" to give ten and forty copies of n. Now plot k.

i) The array 'n' is formed by extracting the data between the 15th and 35th point of array 'y'. Can you change these values so that the array 'k' looks like the figure 3?

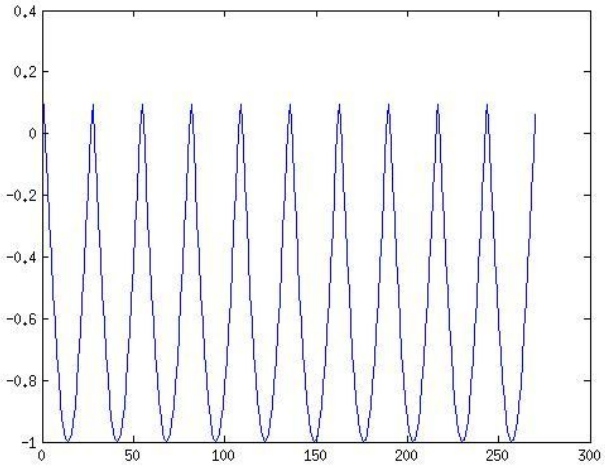

Figure 3: Can you make your plot look like this?

Q8: Using arrays and built in functions together

Copy the file Q7.m to a new file Q8.m, and open Q8.m using MATLAB. Delete everything in the script except the line with your name, and the line with the 'clear' command. In mathematics you will have solved quadratic equations using the quadratic equations

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where a,b and c are defined as:

$$ax^2+bx+c = 0$$

MATLAB has a built in function for solving quadratic equations called 'roots'. The function roots takes as its input an array containing a, b and c. For example if my quadratic equation was $1x^2-3x+1 = 0$, I would define the array as 'numbers=[1 -3 1]'. Then pass this array to the roots function by writing '**roots**(numbers)' in the script. Copy these two lines of code to your script and run it. What roots do you get? Unlike the quadratic equation MATLAB is not limited to only solving quadratics. For example if you wanted to solve the equation.

$$4x^4+5x^3+3x^2+1x+2 = 0$$

simply change the array numbers to read 'numbers =[4 5 3 1 2]'. Then call 'roots(numbers)' Try doing this in your script. This will work with a polynomial of any size, with real or imaginary numbers roots. Try making up a polynomial of order 10 or 20, and factoring it using the roots function.

The University of Nottingham

Q9: Detecting submarines

Ships use a system called Sonar to detect the depth of the water beneath them. It works by using a water proof loud speaker beneath the boat to generate a sound wave. This often sounds like a *ping*. This sound wave propagates away from the boat towards the bottom of the sea, when the sound wave hits the bottom of the sea , it reflects back towards the boat. The time the sound wave takes to travel from the boat, to the sea bed and back to the boat can be used to calculate the depth of the sea. Sonar can also be used for detecting the distance to other objects such as a submarine or a school of fish.

Figure 4: A nuclear powered submarine

No two sonar systems sound the same. For example, a sonar system from a fishing boat will sound very different from a sonar system from a submarine. From moodle download the zip file called 'sounds.zip' and unzip to a directory in your z: drive. This folder contains recordings from two sonar systems, one from a fishing boat and one from a submarine. By clicking on the folder icon shown in figure 5, change the current MATLAB directory to the directory where you have saved the dat files. If you can't do this ask a demonstrator.

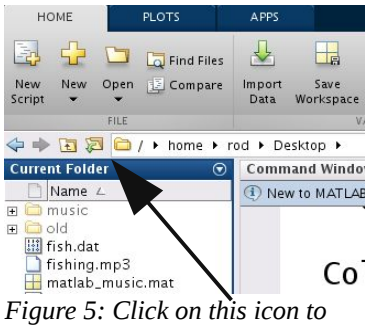

Figure 5: Click on this icon to change the working MATLAB directory.

a) Make a new script, as usual type your name at the beginning followed by the clear command. Load the by using the 'load' command make your script load the data file 'sub.dat' and plot a graph of it's wave form.

b) Edit your script to load the data file 'fish.dat' and again plot it's wave form.

c) Compare the plots of the sonar from the fishing boat and the submarine. What are the differences? If you want to display two figures at the same time you have to put the 'figure' command before each plot command, this will tell MATLAB to put the plot in a new window and not overwrite the old plot.

d) If you have brought head phones use the 'sound' command to listen to the different sonar sounds.

e) Calculate the maximum and minimum value in the sound file fish.dat and in the file sub.dat.

f) You are working for a company designing a system to differentiate between submarine and fishing boat sonar systems. To test if the system can differentiate between submarine sonar and fishing boat sonar, you are asked to make your computer produce five fishing boat sonar pulses

followed by one submarine pulse followed by another five fishing boat sound pulses. Generate a new array containing this data. Plot this array, and if you have brought your headphones listen to it.

f) You have now been asked to simulate the sound that would be heard by a microphone on board a submarine using a sonar system to measure the depth of water beneath it. You should generate an array 'z' containing just the submarine sonar sound. To represent the time it takes the sonar pulse to travel to the sea bed and back, you need append a large array of zeros to 'z'. You can do this by multiplying the original sonar signal by 0 and appending this to array 'z'. After the sonar signal has bounced off the sea floor it will be a much weaker signal than first heard. To simulate this append the original sonar sound to the end of the array 'z' multiplied by 0.33. Now plot the sonar signal. If you have headphones you can listen to this sound.

g) The electronic circuit that listens for the sonar pulse is not perfect and produces a background noise hissing sound. Your job is to simulate this hissing sound, this is done mathematically by adding a random signal to the top of the array 'z'. You array 'z' should be exactly 400896 elements long. You can generate a random array this long with the command 'r=rand(1,400896)'. Plot the array r and have a look at. Then add array 'r' to array 'z', and plot it again. What does it look like? Finally listen to the array 'z' and see if you can still hear the sonar pulse?

h) The electronic circuit has been improved by an engineer and now produces less noise, multiply array r by 0.1 to decrease the noise in the system. Can you now hear the sonar pulse?