Lab 1

AM 10 Winter 2020
Introduction to MATLAB and Complex Algebra

This lab is intended to get you familiar with accessing MATLAB on school computers and/or on your personal computer. You will also get a feel for inputting commands, saving .m (MATLAB script) files and outputs, and get familiar with the submission process.

Please submit your lab writeup through Canvas as a single PDF document with total filesize < 10 MB. You can type (and export to PDF) or hand-write (and scan) your submission. Make sure to double-check what the final PDF looks like – we can't give credit for things we can't read.

Some problems will ask you to submit your MATLAB code. For this you can submit the .m files themselves (separate from the PDF, this is the only thing that can be submitted separately). Other problems will ask you for calculations and/or figures produced by running your code. Calculations can be saved using the `diary` function (we'll go into detail later), and figures can be exported as .pdf files.

We recommend you start this lab early to make sure you have time to take care of any technical difficulties.

Warmup: Accessing MATLAB

First let's get familiar with how to access MATLAB at school and/or on your personal computer. On any school computer, log in with your CruzID (i.e. your UCSC email without the `@ucsc.edu`). MATLAB should be installed on all school computers. UCSC also provides a license so you can download and use MATLAB on your own computer for free. Please go to https://its.ucsc.edu/software/matlab.html.

Open up MATLAB. You should see something like Figure 1. In the *Command Window* you can type in commands (obviously). Try typing in

```
z1 = (3 - 1i * 4)^-3
```

This command creates a complex variable which has the value

$$z_1 = (3 - 4i)^{-3} \approx -0.0075 + 0.0028i. \tag{1}$$

Now try

```
z2 = z1 * 4 * (cos(2 * pi / 3) + 1i * sin(2 * pi / 3))
```

which should calculate a new variable

$$z_2 = z_1 \cdot 4[\cos(2\pi/3) + i \sin(2\pi/3)] \approx 0.0052 - 0.0316i. \tag{2}$$

Variables in MATLAB can be any combination of letters and numbers, as long as the variable doesn't start with a number (for example `x1` or `x123y` are allowed but `1x` is not). You can also update or overwrite the values of existing variables. For example, try

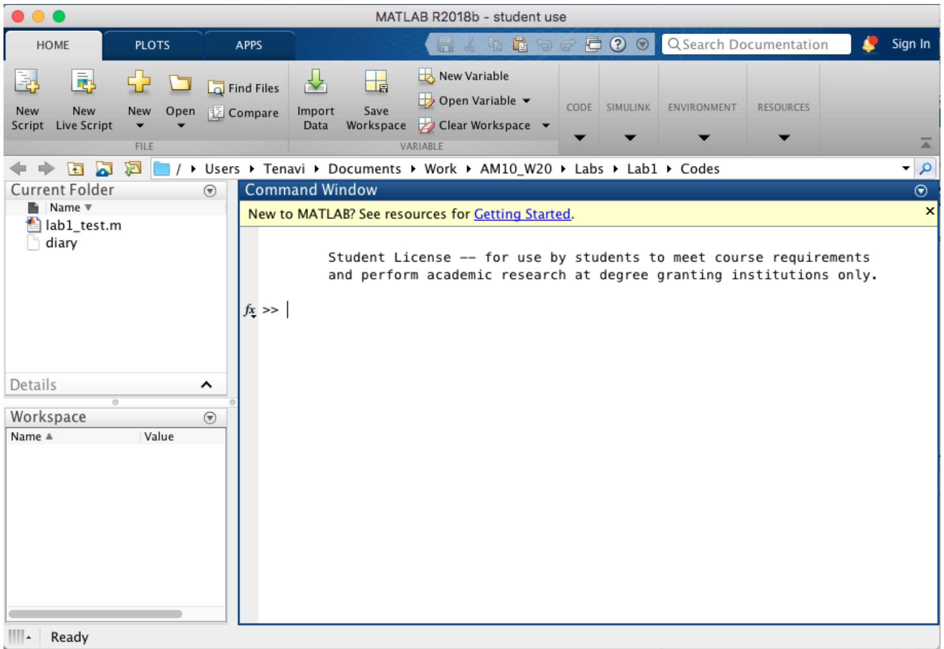

Figure 1: A blank MATLAB startup screen.

```
z1 = sqrt(-1)
z2 = z2 + exp(1.5)
```

You should end up with

$$z_1 = \sqrt{-1} = 0.000 + 1.000i \tag{3}$$

and

$$z_2 = z_2 + e^{1.5} \approx 4.4869 - 0.0316i. \tag{4}$$

If you didn't quite get the correct answers, make sure that your parentheses are in the correct places.

In the above calculations we used all the basic operations in MATLAB:

`+`	`-`	`*`	`/`	`^`
addition	subtraction	multiplication	division	exponentiation

and well as the built-in functions

`cos(z)`	`sin(z)`	`sqrt(z)`	`exp(z)`
$\cos(z)$	$\sin(z)$	$\sqrt{z}$	$\exp(z) = e^z$

Also note that to enter the imaginary number i, we actually typed in `1i`. This prevents us from mixing up imaginary numbers i with *variables* which are named `i`.

Problem 1 Saving inputs and outputs using `diary`

The `diary` command can be used to save Command Window inputs and outputs. To start recording, type

```
diary on
```

MATLAB will start recording the Command Window to a text file (called diary) to the current active folder. If you want to change the name of the file to record to, type

```
diary filename
```

where `filename` is the name of the file you want to save your diary in. Note that MATLAB will automatically record to the end of the existing file. To stop recording, type

```
diary off
```

You can open the diary file in any basic text editor. You can copy and paste parts of the diary into your lab report (if typing), or export the diary as a PDF and add the page into your submission (if submitting a scan of a handwritten document). ***Do not submit the entire diary! The diary will almost certainly have lines in there that don't show anything useful. Edit it down to what you really need it submit.***

Problem 1.1 Solutions of quadratic equations

Consider the quadratic equation

$$2z^2 - 2z + 1 = 0. \tag{5}$$

Use the quadratic formula to solve (5) by hand.

Problem 1.2 Quadratic formula in MATLAB

Now use MATLAB to double-check your hand calculations. Use the `diary` command to record your results and add them to your lab submission. We'll use MATLAB in two ways. First, if we write the polynomial on the left hand side of (5) as

$$p(z) = az^2 + bz + c, \tag{6}$$

figure out which real numbers correspond to a, b, and c. This should be straightforward since you should have already done this when using the quadratic formula in Problem 1.1. Store these as variables in MATLAB. For example

```
a = 2
```

Once these are all entered, compute the first root with the command

```
z1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a)
```

Modify this command to compute the other root, `z2`.

Problem 1.3 The `roots` function

Next we'll use MATLAB's built-in `roots` function to do this more easily. To do this, create a new variable

```
p = [a, b, c]
```

The variable `p` is a 1×3 *vector*, which is an *array* with one row and three columns. We'll learn all about vectors and arrays later in the course, so for now, just think of `p` as a convenient way to store all the polynomial coefficients. Compute the roots of (6) by

```
z = roots(p)
```

Do the roots match your answers from Problem 1.1 and Problem 1.2?

Problem 1.4 Higher-degree polynomials

The `roots` function also works for higher-degree polynomials. Use this function to compute solutions of the polynomial equation

$$z^7 + 2z^6 - z^5 - 3z^4 + 7z^2 + z - \cos(\pi) = 0. \tag{7}$$

In addition to submitting your `diary`, *write down* the roots in your report. How many roots are there? Are they real, imaginary, complex?

Problem 2 Writing and running .m files (MATLAB scripts)

Now look in the top left corner. You should see a *New Script* button. Click this to open up the *Editor* window and a blank .m file. You should see something like Figure 2. Creating a MATLAB script lets you run the same program repeatedly, without having to type in the same commands over and over again. First try entering a couple lines from the warmup into the blank file and run the script with the green play button.

Problem 2.2, Problem 2.4, and Problem 2.5 (extra credit) will ask you to write some simple code. For this problem, put all your answers into one script and submit the .m file along with your PDF lab report.

Problem 2.1 Complex algebra

Consider two complex numbers,

$$u_1 = (1 - 2i)(2 + i) \qquad \text{and} \qquad v_1 = e^{i\pi/4} + \frac{-3i}{1 + 2i}. \tag{8}$$

Put u_1 and v_1 into standard form and thereby find their real and imaginary parts. That is, find real numbers a and b such that $u_1 = a + bi$, and likewise for v_1.

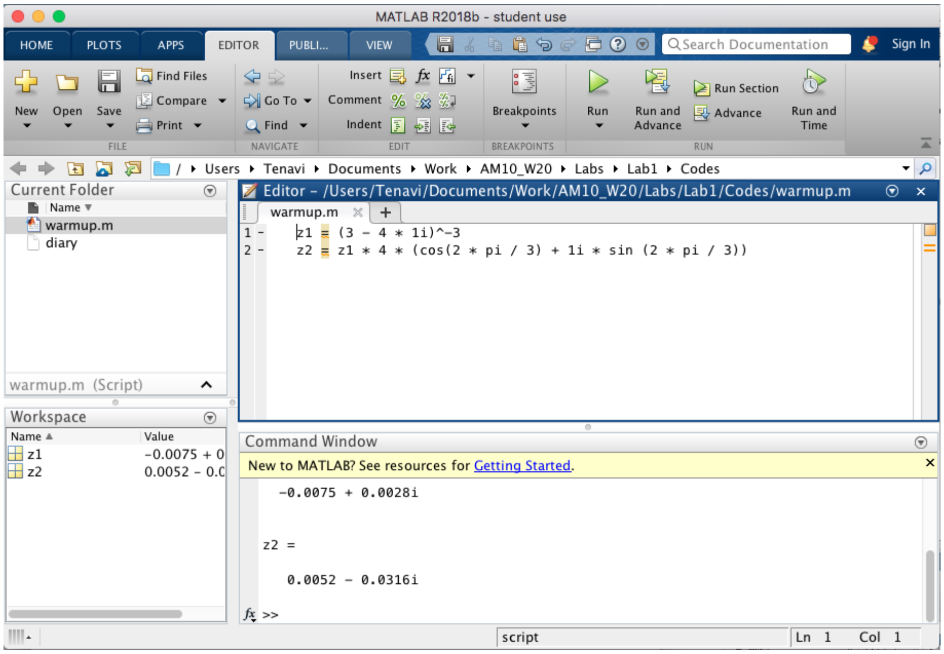

Figure 2: The MATLAB script editor.

Problem 2.2 Complex algebra in MATLAB

Now create a new script to compute u_1 and v_1 and find their real and imaginary parts. Once your code is working, use the `diary` to record the output of your script as your answer for this problem.

Hint: Use the `real` and `imag` functions. You can add the following lines to your program to display the real and imaginary parts of a variable `u1`:

```
re_u1 = real(u1)
im_u1 = imag(u1)
```

Problem 2.3 Converting between standard and polar forms

Consider two complex numbers,

$$u_2 = (-5 + 5i)e^{i3\pi/4} \qquad \text{and} \qquad v_2 = e^{-i\pi/3}\left[2\cos(\pi/6) + i \cdot 2\sin(\pi/6)\right]^7. \tag{9}$$

Put u_2 and v_2 into polar form and thereby find their moduli and arguments. That is, find real numbers r and θ such that $u_2 = re^{i\theta}$ and likewise for v_2.

Hint: Convert all the numbers to polar form before multiplying or taking powers.

Problem 2.4 Polar form in MATLAB

In your script, use the definition of the modulus,

$$r = |z| = \sqrt{[\mathrm{Re}(z)]^2 + [\mathrm{Im}(z)]^2}, \tag{10}$$

to compute $|u_1|$, $|v_1|$ for (8) and $|u_2|$, $|v_2|$ for (9). Do your answers for $|u_2|$ and $|v_2|$ match your answers from Problem 2.3?

There's an easier way to do this: the `abs` function computes the modulus or absolute value for any complex or real number. Compare your results using `abs` to those obtained using the definition (10).

Use the `angle` function to compute the arguments $\arg(u_1)$, $\arg(v_1)$ for (8) and $\arg(u_2)$, $\arg(v_2)$ for (9) in MATLAB. Do your answers for $\arg(u_2)$ and $\arg(v_2)$ match your answers from Problem 2.3? Remember that angles are not unique: every time you add or subtract 2π to or from θ you go around in a full circle and come back to the same place.

Along with the (short) discussions asked for above, use the `diary` to record the output of your script.

Problem 2.5 *Extra credit:* principal arguments

Recall that Shores defines the principal angle (or argument) as

$$\theta_0 = \mathrm{Arg}(z) = \arctan\left(\frac{\mathrm{Im}(z)}{\mathrm{Re}(z)}\right) \in [0, 2\pi). \tag{11}$$

On the other hand, MATLAB's `angle` function returns $\arg(z) \in [-\pi, \pi]$. Both definitions are valid, but people usually define the principle argument as $\mathrm{Arg}(z) \in [0, 2\pi)$.

If in Problem 2.4 you got any results which are outside of $[0, 2\pi)$, how would you change these to get the principle arguments? Discuss your answer before writing any code.

Add an if statement to your code from Problem 2.4 to make sure that all the angles lie in $[0, 2\pi)$. Use the `diary` to record the output of your script. An if statement has the following general structure:

```
if <statement is true> then
    do something
end if
```

Hint: The syntax in MATLAB is a little different than the pseudocode above – please see the documentation and the example below. It is fine to assume that the angle MATLAB gives you is in $[-\pi, \pi]$, and if an angle is already in the correct range, don't do anything to it. You might also find the `<` command useful. For example,

```
if a < b
    ...
end
```

Problem 3 Plotting in MATLAB

Create a new script for this problem and submit the .m file along with your PDF lab report. First create a polynomial with random coefficients. In your new script, add the lines

```
p = rand(1,21) ;
p = 2 * p - 1 ;
```

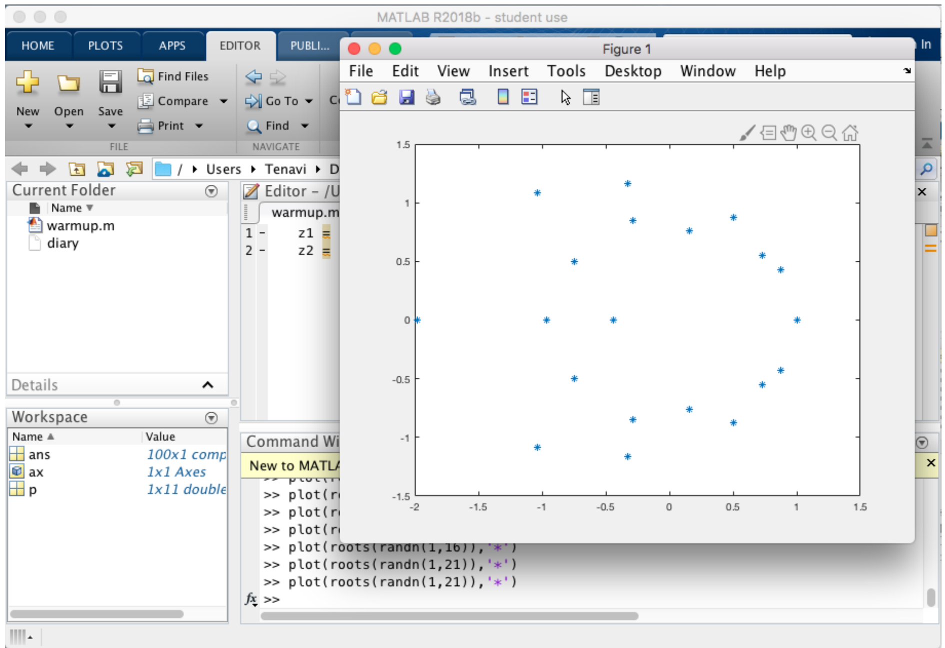

Figure 3: A figure in MATLAB.

The `rand` command returns arrays where the entries are random numbers in the range $[0, 1]$. Here we've used this to create a 1×21 array like we saw in Problem 1.3. What does the line `p = 2 * p - 1` do?

Also notice the semicolon (`;`) at the end of each line. This optional command tells MATLAB not to print out the result of the calculation, which is nice if you don't need/want to see every little detail.

Next, compute the roots of the 20th degree polynomial defined by

$$p_1 z^{20} + p_2 z^{19} + \cdots + p_{20} z + p_{21}, \tag{12}$$

where p_i is the ith entry of the random vector `p`. `plot` those roots using the command

```
plot(roots(p), '*')
```

You should see something like Figure 3. Let's play with this figure a little more. Add the following lines to your script:

```
xlabel('real'), ylabel('imaginary')
title('Roots of a random polynomial')
ax = gca ;
ax.XAxisLocation = 'origin' ;
ax.YAxisLocation = 'origin' ;
```

Re-run the script and go back to your figure. Click File → Save As. Under Format, choose Portable Document Format (*.pdf), then pick a name for your figure and save it. The saved figure can now be inserted into your lab report PDF.

Do you see any symmetry or patterns in the distribution of the polynomial roots?