MATLAB Exercise

Exercise C11

[40pts] Find all of the *n*th roots of a complex number and draw them using MATLAB, as outlined below.

Consider a complex number $\mathbf{z} = a + \boldsymbol{j}b = Ae^{j\varphi}$.

(a) Derive an expression for ALL of the *n*th roots of **z**, i.e.,

$$\mathbf{z}^{1/n} = Re^{j\beta} = X + \boldsymbol{j}Y$$

where *n* is an integer. In other words, treating parameters *A* and φ as known, derive expressions for *R*, β, *X*, and *Y* suitable to be used to draw the complex numbers. You will also need to derive an expression for $d\beta$, the phase difference between two adjacent *n*th roots of **z**.

(b) Use your expressions from part (a) to write the missing expressions in the MATLAB code on the next page. (You may instead use the "Getting Started" M file on Moodle.)

(c) Design a check to verify that <u>each</u> calculated value of *X* and *Y* satisfies $\mathbf{z} = (X + \boldsymbol{j}Y)^n$.

(d) Run your fully functioning script to determine the following:

1. $(3+j4)^{1/5}$
2. $(2-j5)^{1/4}$
3. $(-1)^{1/3}$
4. $(-1-j0.0001)^{1/3}$
5. $(-j)^{1/9}$
6. $(5+j15)^{1/7}$

(e) Explain why the principal cube root of –1 is not equal to –1. HINT: Compute it by hand.

(f) Explain why the principal cube root in d3 is different from the principal cube root in d4. HINT: Compute it by hand, treat the amplitudes as equal, and highlight the major differences.

NOTES:

- In Electrical and Computer Engineering, we generally write complex numbers as $a + jb$, but the input to MATLAB will need to be in the form a+bj. (You may also use the form a+bi, but I recommend that you start getting accustomed to "j" as the imaginary unit, as we generally use *i* as the symbol for current.) For instance, enter "3+4j", "2-5j", "-1", …; that is, no parentheses or * operators are needed.
- Add a few lines at the beginning of your script with your name, the date you started, the assignment (ECE 296C Fall 2023), and a very brief description of this assignment.
- The script is set up to work in degrees, so you should use the functions COSD and SIND to compute in degrees, not radians.
- The checks for part (c) will output automatically as a column array of very small complex numbers, once you have written suitable expressions. If the checks are not very, very small, there is a mistake.
- The answers to parts (e) and (f) should be added as comments to the end of your script. Use the exponential form of each complex number throughout your explanations. That is, you never need to convert anything to rectangular form.
- Print your final script to PDF, in color, with line numbers and a proper header. Print the Command Window also to PDF with a proper header. Suppress output for everything but the checks.
- Export each figure using "Save As…" PNG or JPG. In other words, you should not take any screen shots. You might need to adjust the size of the Figure window to fit the title. Adjust the window also to make sure the font sizes are appropriate. Combine the PDFs of your script, your hand calculation, and the Command Window with all six figures, e.g., using http://online2pdf.com, into one PDF.
- The older version of "Finishing B1" in the Lecture Links on OWL is very useful.

Exercise C11 (continued)

```
clf   % clear all figures
clear   % remove all variables from the workspace
format shortG

% needed to draw axes
ax = [-100,100]; ze = [0,0];

hold on
% plot x- and y-axes as black lines (draw the axes before using QUIVER)
plot(ax,ze, 'k', 'LineWidth', 1)
plot(ze,ax, 'k', 'LineWidth', 1)

% ----- input z = a + jb, then convert to exponential form -----
z = input('Input a complex number z as a+bj: ');
n = input('Input the power of the root, i.e., n of z^1/n: ');
a = real(z); b = imag(z);

% z = a+jb = A e^jP

A = norm(z);   % amplitude of z
P = rad2deg(angle(z));   % phase of z in degrees, -180deg < phi <= 180deg

% ----- set up parameters needed to find n roots of z -----
% z^1/n = R e^jB

R = ***expression***;   % amplitude R of z^1/n
B = ***expression***;   % phase angle in deg, associated with principal value of z^1/n
dB = ***expression***;   % difference between phase angles, in degrees

% R e^jB = X + jY

X = ***expression***; Y = ***expression***;

% ----- plot roots and compute checks -----
% plot an arrow to represent the principal value of z^1/n, in red
quiver(0, 0, X, Y, 0, 'r', 'LineWidth', 3)

check = zeros(n,1);   % initialize n checks as a column vector

check(1) = ***expression***;   % first check, output after the FOR loop

for i = 2:n   % cycle through the other values to make the rest of the arrows

    B = B + dB;   % add dB to find the next root of z
    X = ***expression**; Y = ***expression***;

    % plot an arrow for the next root, in blue
    quiver(0, 0, X, Y, 0, 'b', 'LineWidth', 3)

    check(i) = ***expression***;   % the rest of the checks

end

check   % output the n checks; each should be close to 0+j0

% ----- make the figure look nicer -----
grid on; axis equal;   % make frame square (emphasize symmetry of roots)

ac = gca;
ac.FontSize = 16; ac.GridAlpha = 0.5;   % change fonts to 16pt; make grid darker

xlabel('Re({\textbf{z}})', 'FontSize', 20, 'Interpreter', 'latex');
ylabel('Im({\textbf{z}})', 'FontSize', 20, 'Interpreter', 'latex')

% determine the sign of b, so that we can include z in the title
bSgn='+';
if b<0
    bSgn='-';
end
bMag = norm(b);   % magnitude of b

title({'ECE 296C Exercise C11', ...
    sprintf('Finding and drawing the %g values of $(%g%sj%g)^{1/%g}$', ...
    n,a,bSgn,bMag,n)}, 'FontSize', 24, 'Interpreter', 'latex')

max = ceil(R+0.1);   % round up R to the next integer
axis([-max max -max max])   % set the upper and lower limits of the axes

hold off
```