ECE-210-B HW1

Instructor: Jonathan Lam

Spring 2021

In this assignment, you will utilize the basic skills you learned in the first week of class; you will assign values to variables, use some of MATLAB's basic functions, and perform some simple arithmetic operations on scalars and matrices. You should also use this assignment as a way to get comfortable with the `help` and `doc` functions. It may be good to get into the habit of beginning your scripts with `clc`, `clear` and `close all`. Don't remember what these do? Use `help`!

For your final submission, please suppress all outputs with semicolons. Submitting either a *.m or *.mlx file is fine – recall that they are essentially equivalent. (This applies to future submissions as well.)

Remember, Good Code Style™ is important! Here are some recommendations, but feel free to do what suits you, so long as it is consistent and logical.

- Follow a convention for variable names. It can be `snake\_case`, `camelCase`, `PascalCase`, `alllowercase`, etc. Names that are too short (e.g., `x`) are quick to forget their original purpose, and variable names that are too long (e.g., `theTimerThatIIncrementEvery5Ms`) are annoying. The exception to the first rule is when the name is clear from context, e.g., `x` and `t` to denote time series data, but even then it is usually nice to subscript them (e.g., `x_1` and `t_1` if you are working with multiple time-series). Be sensible!

- Long lines tend to be hard to read, especially on smaller screens. Try to limit lines to a maximum length. MATLAB has a visual indicator at 80 characters to remind you of this. (80 chars is also great for printing!) If you need to break a long expression over multiple lines, you can do so using ellipses (`...`) at the end of a line, e.g.:

```
this_is_a_long_variable_name ...
        = some_long_expression ...
        * another_long_expression;
```

- Use comments when you feel like you need to explain your code. (Recall that comments are lines that start with `%` and are used to annotate your code.) The better and more consistently your variables are named, the less commenting you need to do in order to keep your code maintainable.

- Using sections and consistent spacing makes for easier reading/debugging.

(For this assignment, don't worry too much about variable names.)

1. Create (scalar) variables with the values:

   - $||4 + 5j||$
   - j^j
   - $(2e + 4j)^{1/2}$
   - $\sec^{-1}(5 - 2j)$
     (hint: arctan:arcsec::`atan`:`?`)

   Then make a row vector containing each of these four values in order as entries. Do this using variable names, i.e. don't just copy and paste the mathematical expressions.

2. Compute the real part, imaginary part, magnitude and phase of the two complex variables from Question 1. Store these values in a 4×2 matrix, with each column representing one of the complex numbers.

3. Create and store in a variable the 4×1 vector with the sum of each row of the matrix from question 2 using the `sum` command. Consult `help sum`.

4. From questions 1 and 2, you now have a 1×4 and a 4×2 matrix. Call the first vector A and the second one B. Then perform each of the following operations, and save the result in a (unique) variable:

   (a) AB (regular matrix multiplication)
   (b) Use `repmat` (or the abbreviated concatenation syntax; see `help vertcat`) to make A a 2×4 matrix; call this C.
   (c) $C^T - B$
   (be careful to perform the transpose, not the conjugate transpose)
   (d) Elementwise multiplication of C^T with B
   (e) Elementwise square of CB
   (f) $(CB)^2$
   (g) $\det(CB - 2I_2)$, where I_2 is the 2×2 identity matrix

5. Generate the following ranges (vectors) using the more appropriate method covered in class (i.e., using `linspace` or the colon operator):

   - A length 5000 vector of equally-spaced values from π to e, inclusive.
   - The vector of time samples, in units of seconds, if you sample at 7MHz for 2 seconds (i.e., between $t = 0$ and $t = 2$). (E.g., sampling at 2Hz for 3 seconds would give you the vector `[0 0.5 1 1.5 2 2.5 3]`.)

6. *(Optional fun)*: Create a row vector y with 101 values equally spaced between -10 and 10, inclusive. Create a column vector $x = y^T$. Use broadcasting to generate the matrix z where $z_{xy} = x^2 + y^2 - 4$.

7. *(Optional fun)* Try the exercises above in Python using the numpy library.