ECE-210-A HW1

Instructor: Jonathan Lam

Spring 2022

This assignment ensures that you have MATLAB up and running, and aims to get you comfortable with the basics covered in the first lecture, such as variable assignment, matrix operations, and the MATLAB documentation.

For your submission, 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.

- Begin your scripts with `clc`, `clear` and `close all`. (Don't remember what these do? Use `help`!)
- Suppress outputs of intermediate values by ending the line with a semicolon. Long outputs printed in the command window are hard to follow. I prefer suppressing all outputs and storing answers in descriptive variables.
- Follow a convention for variable names. It can be `snake_case`, `camelCase`, `PascalCase`, `alllowercase`, etc. Names that are too short (e.g., `x`) are not descriptive, and `variableNamesThatAreTooLongLikeThis` become tedious. 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 80 characters. (MATLAB has a visual indicator for this.) To break an expression over multiple lines, use ellipses (`...`), e.g.:

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

- Use comments to explain code. (Recall that comments start with `%`.) The better and more consistently your variables are named, the less commenting you need to keep your code maintainable.
- Using sections and consistent spacing makes for easier reading/debugging. Section separators must have two percent signs at the beginning of the line, followed by the space, followed by the section title.

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

1. Create (scalar) variables with the values:
   - $||7 + 24j||$
   - j^{j^j}
   - $\sec^{-1}(5 - j)$
   - The positive root of $5x^2 - 4x - 3$, by writing out the quadratic formula.

   Then make a row vector E containing each of these four values in order as its elements. Do this using variable names, i.e. don't repeat the mathematical expressions.

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

3. Create and store in a variable B the 4×1 vector which contains the row-wise means of A using the `mean` command.

4. What happens if you try to do $A + B$? What about $A + 1$? What about $A + E$? Describe what happens.

5. Perform the following operations, saving each result in a new variable:
   (a) $A^T B$ (use the regular transpose, not the conjugate transpose)
   (b) B with each element decreased by 2
   (c) Use `repmat` (alternatively, see `help horzcat`) to repeat A into a 4×4 matrix. Call this C.
   (d) $C - AA^T$ (again, regular transpose)
   (e) Elementwise multiplication of C with itself
   (f) Elementwise square of C
   (g) C^2
   (h) trace$(C - D)$, where D is the 4×4 identity matrix.

6. Generate the following ranges (vectors) using the more appropriate method covered in class (i.e., using `linspace` or the colon operator). (Please suppress these outputs!)
   - A length 1000 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]`.)

7. *(Optional fun)* Try the exercises above in Python using the numpy library. Note how similar the syntax is.