Math 260, Fall 2020
Last updated: 10/18/20

HOMEWORK 9

DUE FRI. OCT. 30

Reminder: Project topic selection is also due this week (see project document). That submission will be via text on Gradescope.

Exercises (no submission).

E1 (DFT testing). You may want to refer to the example code here but it may also be useful to avoid doing so unless needed.

a) Define the function

$$f(t) = \sin(2t) - 4\cos(6t) + 3\sin(10t)$$

and sample it using N points in $[0, 2\pi]$ (an even number).[1]

Take the DFT, then plot the real and imaginary parts. Check that it correctly identifies the components.

b) Take the inverse DFT, and verify that it gives back the original data. Plot both the real and imaginary parts of the inverse DFT - is the imaginary part zero?

c) Do the same as in (a), but sample with N points in $[0, \pi]$. What k-values correspond to what frequencies in the singal?

Problems (only one this time!)

Q1 (phone numbers). A touch tone phone uses distinct sounds to represent each number.[2] Each digit

$$f(t) = \sin(770 \cdot 2\pi t) + \sin(1633 \cdot 2\pi t)$$

would be the 'zero' sound. This representation allows the sounds to easily be picked out (even among noise). The data is provided in the function `tone_data`.

You are given a file `dial.wav` that contains seven numbers being dialed, along with `.wav` files for some numbers individually and a routine that loads these into python.

a) Take the DFT of the file `0.wav`. Plot the real/imaginary parts *or* plot the magnitude $|F_k|$ vs. k. Use this to get a sense of what the data looks like and identify the peaks and the corresponding k's. Check that the k/L rule gives the right frequency in Hz.

a) Write a function `identify_digit` that takes in a .wav file of a digit being dialed and returns that digit. Test your code on the individual files `0.wav` and `5.wav`. (I've also included an unreleated sound `lion.wav` for testing).

[1] Hint: you can use `np.linspace(0,L,n,endpt=false)` to generate equally spaced points up to but not including the endpoint.

[2] Of course, this example is now quite outdated, but it's a good illustration nonetheless.

You may assume here that there is at most one digit in the input; include a failure case if one is not found. Your algorithm does *not* have to be perfect; you may also assume that there is a distinct peak at the appropriate frequencies (and not at others).

b) Now write a function that takes in the sound file `dial.wav` and returns the phone number being dialed (as a string of digits). For simplicity, assume

- The signal is split into sections of length 0.7 seconds
- One digit is dialed in each section and seven digits are dialed in total

Using the Fast Fourier transform provided by numpy, identify the sequence being dialed; include this answer in a comment at the top of your code. Do the same for `dial2.wav`.

c) Test your code on `noisy_dial.wav`, which adds a bit of background noise. Your code from (b) should still work here, but may require some adjusting.[3]

Some hints:

- It's important to keep track of the frequencies corresponding to each value in the transform. You may want to avoid using `fftfreq`,`fftshift` and do these shifting calculations manually as needed.
- Use $|F_k|$ (magnitude) rather than looking at real/imaginary parts separately, so you have only one thing to search through. Also note that the $+$ and $-$ frequencies come in pairs!
- See the lecture slides for the k/L rule to get frequencies in Hz from the index.

[3]Source: stock audio of an airport; https://www.soundjay.com/ambient-sounds.html.