CSE 3313 – Homework #6

Assigned: Wednesday, October 9, 2019
Due: Wednesday, October 16, 2019 at the end of class

Note the following about the homework:

1. You must show your work to receive credit.
2. For the hand-written problems, submit a hard-copy.

Assignment:

1. (15 points) Find the output difference equation for the following system functions:

$$a)\ \ H(z) = \frac{2z^{-2} - z^{-3}}{1 + 2z^{-1} - 4z^{-2}} \qquad\qquad b)\ \ H(z) = \frac{4 - 2z^{-1} + z^{-3}}{1 - 6z^{-2}}$$

2. (15 points) For each of the following, use z-transforms to find the poles and zeros and draw them on the pole-zero plot. Be clear which are the poles and which are the zeros.

   (a) `y[n] = 3x[n] + 4x[n-1] + x[n-2]`

   (b) `y[n] = 6y[n-1] - 8y[n-2] + x[n] - x[n-2]`

   (c) `y[n] = y[n-2] + 2x[n] + 4x[n-1] + 3x[n-2]`

Programming Applications

3. (35 points) **Application Area: Noise Removal using FFT**
   Purpose: Learn how to display the spectrum of an audio file and how changing some of the spectrum components can be used to remove noise from an audio file.

   Sometimes we wish to identify frequencies, or range of frequencies, present in a signal and alter their amplitude. For this we can produce the DFT of a signal.

   You will write a program that reads an audio file that contains noise and use the DFT to remove the noise.

   (a) On the course website is a skeleton file, `fftNoiseRemoval.py`, that you will use. At the bottom of the file are the lines

```
if __name__ == "__main__":
    filename = "P_9_2.wav"
    offset = 123

    processFile(filename, offset)
```

   You will place your code in the function `processFile()`. The line

```
if __name__ == "__main__":
```

allows you to execute `fftNoiseRemoval.py` while also allowing the function `processFile()` to be imported by another program. This will allow me to call your function with my own arguments. For testing purposes, you can change the filename and the offset but I will test your code using my own version of `fftNoiseRemoval.py` that will call `processFile()`.

(b) Note that it is OK to decompose your code into functions, for example, by creating a function for plotting. However, calling `processFile()` with arguments should be how all of your code is executed. Think of `processFile()` as the "main" of your program.

(c) On the course website is a file, `P_9_2.wav`, that contains a piece of music that has been corrupted with noise. This file is from [SB96].

(d) To remove the noise, you will

i. Apply the FFT (see `np.fft.fft()`) to the signal.
ii. Find the index of the midpoint of the FFT values.
iii. Choose an offset and then set the values in the range of `midpoint` $\pm$ `offset` to 0. Make sure that the number of values on each side of the midpoint that you change are the same. Depending on how you implement this in Python, it's easy to have an off-by-one bug.
iv. You can assume that N is even, so the midpoint value won't have a matching frequency component.
v. Create a new, cleaned signal by applying the inverse FFT (see `np.fft.ifft()`) to the modified FFT values.
vi. Create two subplots, side-by-side. The left plot will be the magnitude of the FFT values from the original audio file. The right plot will be the magnitude of the FFT values after removing the noise frequencies. Note that these are magnitudes being plotted.
vii. Note that the process that we are using actually creates some issues that will result in the inverse FFT not producing real values. To overcome these issues, our saved signal will use the real part of what the ifft() function returns. **This is not the same as the magnitude of the values.**
viii. Write a new WAV file called `cleanMusic.wav`.

(e) Use the PySoundFile library `https://pysoundfile.readthedocs.io/` for reading the original audio file and for writing the cleaned version of the audio.

(f) Don't hard-code your logic to this particular data beyond what you are told above.

(g) *On your own, here are some things to investigate:*

i. Print some of the frequency component values between what you think are the actual song components and the noise. Notice they are small, but not zero.
ii. Instead of using the real parts of what ifft() returns, use the magnitude (i.e., abs()). How does this affect the sound?

4. (35 points) **Application Area: Image Processing**
Purpose:

- Learn to use correlation to find an image within another image.

- Gain more experience with image preprocessing.

We sometimes wish to find a signal within a larger signal. This is particularly challenging in the presence of noise. One approach to solving this is to find the **cross-correlation** of the smaller signal to the larger signal, where the largest correlation value represents the best fit.

An applicaton of this is in image processing when we wish to locate a small image within a larger image. This process is called **template matching**.

Note the following about your submission:

(a) On the course website is a skeleton file, `templateMatching.py`, that you will use. At the bottom of the file are the lines

```
if __name__ == "__main__":
    mainImage = "ERBwideColorSmall.jpg"
    template = "ERBwideTemplate.jpg"
    r, c = findImage(mainImage, template)

    print("coordinates of match = (%d, %d)" % (r, c))
```

You will place your code in the function `findImage()`. The line

```
if __name__ == "__main__":
```

allows you to execute `templateMatching.py` while also allowing the function `findImage()` to be imported by another program. This will allow me to call your function with my own arguments.

(b) Note that it is OK to decompose your code into functions, for example, by creating a function for plotting. However, calling `findImage()` with arguments should be how all of your code is executed. Think of `findImage()` as the "main" of your program.

(c) When submitting, only include your source code. Do not include the images described below.

(d) The images you will use are in the compressed file, `ERBwide.zip`, which is on the course website. Unzip it wherever you do your development.

(e) Your program should read these images from the same directory as your `.py` file and use the names of the files as given.

(f) Do not use OpenCV or any other library that is not part of the default Anaconda installation.

(g) Each image is in color, which will result in multiple layers when you read the file. In order to make our process simpler, you will convert each image to grayscale using the ITU-R BT.6012 luma transform. For an example, see How can I convert an RGB image into grayscale in Python?

(h) Display both original images after converting them to grayscale. Each should be in its own figure window.

(i) In order to perform the template matching, we will use the `match_template()` function in the library `skimage.feature`. The image to be searched is `ERBwideColorSmall.jpg` and the template being searched for is `ERBwideTemplate.jpg`. Note the following about this function:

- It applies **normalized** cross-correlation, which is more appropriate for this task.
- Unlike `scipy.signal.correlate2d()`, only applies cross-correlation with the template fully within the larger image. Therefore, the coordinates within the larger image where the template best fits will be the upper left-hand corner of the template. For example, if the best fit for the template was the upper left-hand corner of the larger image, the highest correlation value would be at coordinate (0, 0).

(j) Once you find the location within the larger image that the template best matches,

   i. replace that block of pixels in the larger image with zeros. That is, once you find the smaller image within the larger image, make that block of pixels black so that it's clear that you found the correct location. Do this without hard-coding the size of the template in your code.

   ii. Display the new image with the black box in its own figure window.

(k) The function `findImage()` that you write will return the coordinates within the larger image with the highest normalized cross-correlation.

(l) In order to help you turn the location of the highest normalized cross-correlation value into useful coordinates, you might want to check out
Find row or column containing maximum value in numpy array.

General requirements about the Python problems:

a) As a comment in your source code, include your name.

b) The Python program should do the work. Don't perform the calculations and then hard-code the values in the code or look at the data and hard-code to this data unless instructed to do so.

c) The program should not prompt the user for values, read from files unless instructed to do so, or print things not specified to be printed in the requirements.

To submit the Python portion, do the following:

a) Create a directory using your last name, the last 4 digits of your student ID, and the specific homework, with a hyphen between your ID and the homework number. For example, if John Smith has a student ID of 1000123456 and is submitting hw02, his directory would be named **smith3456-hw02**. Use all lowercase and zero-pad the homework number to make it two digits.

   If you have a hyphenated last name or a two-part last name (e.g., Price-Jones or Price Jones), let's discuss what you should do.

b) Place your .py files in this directory.

c) Zip the directory, not just the files within the directory. You must use the zip format and the name of the file (using the example above) will be `smith3456-hw02.zip`.

d) Upload the zip'd file to Canvas.

References

[SB96] Virginia Stonick and Kevin Bradley. *Labs for Signals and Systems Using MATLAB*. PWS Publishing Company, 1996.