BLG 354E Signals and Systems for CE
2019/2020 Spring
Final Project

- You should write all your code in Python language on Jupyter Notebook. No report files will be collected. Thus, make all your explanations on the notebook.
- You can install Jupyter Notebook by following these steps on this documentation. If you are not familiar with Jupyter Notebook, you can check this tutorial. Also, you can check this tutorial for Python and Numpy and this tutorial for Matplotlib.

- This is a Final Course Project Assignment, cheating is absolutely unethical and morally unacceptable. It will be punished by a negative grade. Also disciplinary actions will be taken.
- Ninova only stores files under 20 MB. If you could not upload your results, be sure that we can easily reobtain your results by running your code again.

1 - Image FFTs (25 pts)

DFT can also be used for multidimensional signals, like images. We can define a DFT X of a grayscale $M \times N$ image x as

$$X(k,l) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) e^{-j2\pi(\frac{km}{M}+\frac{ln}{N})}, k = 0, 1, ..., M-1, l = 0, 1, ..., N-1. \tag{0.1}$$

We can also write the inverse DFT of an image as

$$x(m,n) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X(k,l) e^{j2\pi(\frac{km}{M}+\frac{ln}{N})}, k = 0, 1, ..., M-1, l = 0, 1, ..., N-1. \tag{0.2}$$

In this part of the project you will write the Python code to create Fourier representations via 2D FFT for the grayscale images. You can use built-in libraries to calculate FFT of images (e.g. np.fft.fft2). To read and write images, you can benefit from the following skeleton code which reads an image, rotates it and writes again as a different file. You will use OpenCV library for this part of the project, but it is not necessary at

this point to delve into it deeply.

```
import numpy as np
import cv2

image = cv2.imread('lena_grayscale.jpg', cv2.IMREAD_GRAYSCALE) #Reads the image having the shape (512,512) and values between [0-255].

image = np.rot90(image)

cv2.imshow('First Window', image)
cv2.waitKey()#A window with name "First Window" will be opened and the rotated image will shown untial a key is pressed.

cv2.imwrite('lena_rotated.jpg', image) #The image is saved.
```

In this part of the project, you will work on two grayscale images *lena_grayscale.jpg* and *fabric_grayscale.jpg*.

- Read each of the images.
- Using 2D-FFT, calculate 2D Fourier representations of these images.
- Decompose these representations into its **magnitude**s and **phase**s. Remember, phase of each point is just an angle on the complex plane.
- What kind of operation is needed to recreate the original images from magnitude and phase information? Recreate the images and show them.
- Obtain new images by using one image's phase and other image's magnitude information. Using the method given in the previous point, this time you will face with two problems:
  - Very small imaginary values may appear at a level of $10^{-13}s$ due to numerical computation. You may omit imaginary parts.
  - The recreated image is not between 0-255 now. You should do a max-min normalization to remap the image between [0-255]. You can do it with $x_{new} = \frac{x-min}{max-min} * 255$.

If everything goes well, you will obtain the images given in Figure 1.
Explain the effect of using this type of operation on the images. Which effects on the images are related with phase information?

Hint: Be careful about casting since the read images are in unsigned int type (np.uint8). However, after doing the operations, you will obtain data from float type. To show the images in the correct way, it is useful to recast them to uint8.
For your questions about the question 1: Yusuf H. Sahin (sahinyu@itu.edu.tr)

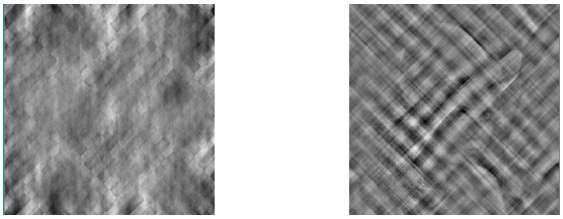

Figure 1: Example outputs for Part 1.

2 The Spectrogram (30 pts.)

With the document you will receive two different numpy files namely **spectrogram.npy** and **phases.npy**. These two files are created by analysing a small music file using a rectangular filter with size 2000. Step size is also used as 2000. While the spectrogram have the magnitude information of the audio as usual, **phases.npy** has the phase information of each windowed area. Instead of using built-in spectrogram functions, you will calculate it step by step.

- Plot the spectrogram in linear scale (**Spectrogram A**).
- Reobtain the audio using the given information.
- Use Hann windows with sizes 500 (**Spectrogram B**), 2000 (**Spectrogram C**), 4000 (**Spectrogram D**) to obtain new spectrograms. Let step size be equal to window size.
- Compare the four spectrograms. Explain the differences and the reasons for them if any.

For your questions about the question 2: Yusuf H. Sahin (sahinyu@itu.edu.tr)

3 Continuous Signal Sampling (10 pts.)

- Let $x = sin(\frac{3\pi t}{4} + \frac{\pi}{2})$ and $y = sin(\frac{\pi t}{4} + \frac{3\pi}{5})$.
- Sample these two signals using 5 different sampling frequencies. Plot these sampled signals. Among them which are more relatable to the original signals. Show undersampled and oversampled examples. Illustrates the phenomenon of aliasing.
- What are the minimum sampling rates to show these signals? Why?
- Make Discrete Time Convolution between sampled signals with different rates. (Please do not use built-in convolution functions, you should write your own convolution function.)

For your questions about the question 3: Abdullah Ekrem Okur (okurabd@itu.edu.tr)

4 Amplitude Modulation (35 pts.)

In this question, you are expected to implement a Frequency-Division Multiplexing (FDM). FDM is used to stack multiple signals on the same channel by allocating the different frequency bands to each signal. You should build a system which consist of a sinusoidal amplitude modulator and sinusoidal demodulator. Please examine the diagram given below in detail. Using FDM, we will create a virtual AM radio broadcast system which have two channels. Two audio files belonging to İlber Ortaylı and Emrah Safa Gürkan, which are given with the homework file, should be stacked in a single channel with amplitude modulator and then reconstructed through the demodulator part of the system. For details, please follow the step-by-step description below.

- Read the sound files and plot their graphics separately in time and frequency domain. You can use *np.fft.fft* method to obtain discrete Fourier Transform of signals and *scipy.io.wavfile.read* method to read sound files.
- Filter the sound signals with a lowpass filter to obtain bandlimited signals. You should decide the bandwidth by examining the signals and their frequencies. Plot the filtered signals in the frequency domain. Please briefly explain why signals should be bandlimited before the modulation process. Which problems arise if we do not use bandlimited signals in amplitude modulation? You can use *scipy.signal.butter* to design filter.
  NOTE: You don't need to go into the details of the digital Butterworth filter design for all filters which are used in this question. You can simply design your filters using this sample code and scipy documentation here.
- Produce two different carrier signals to appropriate the bandwidth you decide. Write down the frequencies of the carrier signals clearly and briefly explain why you choose these frequencies. In this step, you are expected to try different frequencies and then observe their results and choose the best combination.
- Obtain amplitude modulation of the sound signals with carrier signals and stack these signals into a single signal. Then, plot the stacked signal in time and frequency domain.
- Implement demodulator part in figure 3 like modulator part. Plot the graphics of the results at each step of the diagram. Write the filter parameters of your choice clearly for both filters and briefly describe what you have chosen. Explain why these filters are used in the demodulator.
- Compare reconstructed signals with original ones using graphs in frequency and time domain.
- Write a function that saves the reconstructed audio file of the desired historian using your implementations above. You can use *scipy.io.wavfile.write* to save audio files. Briefly interpret the reconstructed sounds and state your observations.

- Finally, run again FDM system you implemented above with overlapping carrier frequencies and listen reconstructed sound signals. Briefly explain your observations.

NOTES:

- Please use a separate Jupyter Notebook cell for each step and upload a version of notebook contains all graphics and results. Also, to make the graphics easier to analyze, draw the graphics in large size.
- Please make sure that all the plots have following attributes such as title, label of axis, legend etc.
- Please examine the relevant sections from our textbook and slides for Frequency-Division Multiplexing.

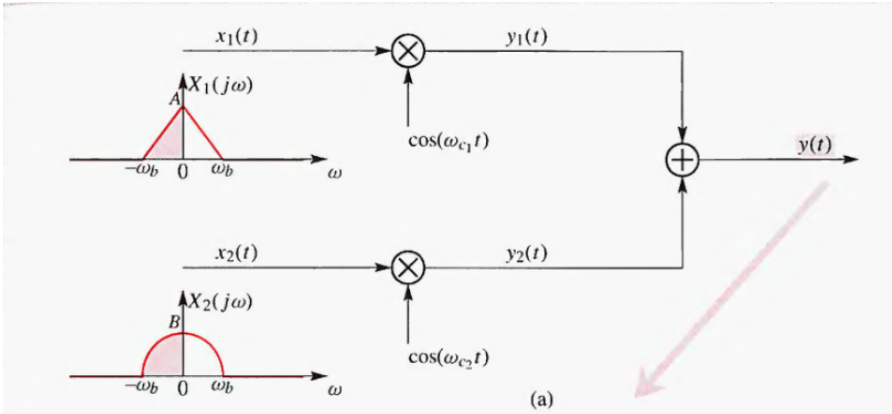

Figure 2: Block Diagram of Amplitude Modulator[1]

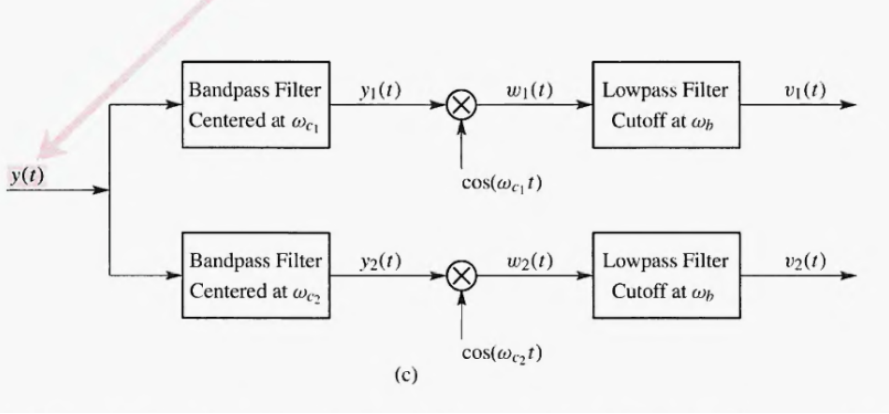

Figure 3: Block Diagram of Demodulator[1]

For your questions about the question 4: Abdullah Ekrem Okur (okurabd@itu.edu.tr)

Bibliography

[1] J. H. McClellan, R. W. Schafer, and M. A. Yoder, *Signal processing first*, 2003. 5