UVIC COMPUTER SCIENCE 475, Spring 2016
ASSIGNMENT #1

DUE January 28, 2015

There are 2 question each worth 5 points. They are marked with stars as follows: (*) basic (**) average (***) intense. The assignment is worth 10% of the final grade. There is some variance in the amount of time each question probably will require. Therefore don't expect them to be equally difficult even though they are all worth the same number of points.

Please provide your answers in a **SINGLE PDF** file through the Connex website for the course. Also please answer the questions in order and explicitly mention if you have decided to skip a question. For questions that involve programming provide a listing of the relevant code but not all the details that are not directly relevant to the question.

Question #1. (5 points)

The goal of this question is to familiarize you with sine waves and audio as well as the basic operations from which the DFT is built i.e the vector inner product and sinusoidal basis functions. You can use any programming language for your implementation but it will probably be easier to do in an environment that supports plotting such as Matlab/Octave or Python/NumPy/SciPy/Matplotlib. Feel free to use the code we examined during class as a foundation for your submission.

- **(1pts) (*)** Write a function that makes a mixture of three harmonically related sinusoids with frequencies $f, 2f, 3f$ with user provided amplitudes and phases. Show a time domain plot of adding three sinusoids with amplitude $1.0, 0.5, 0.33$ corresponding respectively to $f, 2f, 3f$ all with 0 phases and another plot with random phases. Using this function generate two seconds of audio in .wav format for a mixture with $f = 440Hz$. Listen to the generated mixture using an audio editor like *Audacity*. Provide a figure showing the spectrogram of your mixture using *Audacity*.

- **(1pts) (*)** Generate 3 seconds of audio containing a mixture of two sine waves with frequencies $500Hz$ and $502Hz$. Listen to the generated sound. What do you hear ? Try to explain what is happening.

- **(1pts) (*)** Read about the concept of signal to noise ratio (SNR). Write a function that takes as input a SNR in decibels (dB) and a frequency in Hz in order to generate a mixture of white noise and a 440 sine wave. Generate 2 seconds of the corresponding audio and view/listen to it in *Audacity*.

- **(1pts) (**)** Consider a mixture of 3 harmonically related sinusoids as the ones you created. What you would like to devise is a process to estimate the amplitudes of each sine wave assuming that you know the frequencies that the mixture is composed of. Try taking the inner product of a mixture with a unit amplitude sinusoid of frequency f, then take the inner product of the mixture with a unit amplitude sinusoid of frequency $2f$. Make sure that the phases of the "probing" sinusoid are the same as the phases of the mixture sinusoids. What do you observe about the inner products ? Describe how you could modify this procedure to estimate the amplitude of mixtures of 4 sinusoids mixed with noise.

- **(1pts) (***)** Now let's make the scenario a little bit more challenging. We still know the frequencies of the mixture but we want to estimate not only the amplitudes but also the phases. Observe what happens with the inner products when the phase is not zero. Plot the inner products of the input mixture for all possible phase shifts of the probing sinusoid. On that plot do you notice anything different about the "correct" phase ? Based on your observation describe and also implement an algorithm for estimating both amplitudes and phases. Show using one or more examples how it works.

Question #2. (5 points)

The goal of this question is to explore one of the most important tools in music information retrievial, signal processing, and engineering in general the Discrete Fourier Transform (DFT) and its computationally efficient implementation the Fast Fourier Transform. In addition we will continue looking into audio programming and processing sound in buffers.

- **(1pt) (*)** Write code to read/write data from a .wav file (you can use a library) in buffers of 2048. Verify that your code can read, apply simple processing (like a gain) and write audio files correctly.

- **(1pt) (*)** Using any library or implementation of the Fast Fourier Transform for your programming language calculate the frequency domain complex spectrum of the 3 component mixture signal from question 1. Plot the magnitude spectrum.

- **(1pt) (*)** Using a programming language of your choice write code to directly compute the Discrete Fourier Transform (DFT) of an input array. You should express everything directly at low level using array access, *for* loops, and arithmetic operations i.e do not use a complex number type if the language supports it, and do not use any matrix multiplication facilities. Provide a listing of your code and a plot showing that your algorithm produces the same magnitude response as a Fast Fourier Transform routine in your language of choice that is either built in or freely available. Verify that your output is identical with the one by the FFT implementation you used.

- **(1pt) (**)** Modify the code that reads/writes data in buffers so that each buffer is converted to a frequency domain complex spectrum. Compute magnitude and phase spectrum for each buffer and plot an example for each type. Convert the magnitude and phase spectra back to a complex spectrum and then using the inverse DFT return to the time domain and write the buffer to an audio file. If everything is working correctly you should get back the original audio. Once you have verified that's the case then you can perform some simple spectral processing. Replace the phase spectrum with appropriate random numbers uniformly distributed. How is the resulting audio affected ? Try to describe what you hear especially when processing pieces of music. Experiment with large windows corresponding to 3-5 seconds. What happens ?

- **(1pt) (*)** Read about the overlap-add of computing the Short Time Fourier Transform. Basically the idea is to window each buffer and read the data in overlapping chunks so that when they are summed the two windows sum up to 1. First ensure that you can do proper overlap-add by reading buffer that overlap by half and windowing them appropriately. Once you have that working apply the same process as the previous question of randomizing the phase spectrum. What has changed compared to before ?