UVIC COMPUTER SCIENCE 475, Spring 2015
ASSIGNMENT #1

DUE January 29, 2015

There are 4 regular questions and each question is worth 20 points. In addition there are two questions marked as A+ questions that are more difficult and open ended. Each A+ question is only worth 5 points even though it is likely that they will be more time consuming than the regular questions.

IMPORTANT Each question has some additional deliverables for the graduate students taking the course. These are marked with [**CSC575**]. Undergraduate students are welcome to do them but they are optional and will not affect their grade.

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. There is no need to copy the assignment specification. 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.

You can either hand me a paper copy of your submission in class or email me an electronic copy.

Question #1. (20 points)

The goal of this question is to familiarize you with the basic operations from which the DFT is built i.e the vector inner product and the 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.

- **(4pts)** Write a function that takes as input a period N expressed in samples and a number of sample S, an amplitude A, and phase ϕ in samples, and returns an array of size S containing the discrete samples of the sinusoid with the specified period, amplitude, and phase. Show a plot of 100 samples of a sinusoid with amplitude 1.0, ϕ 10 samples and a period of also 100 samples created by calling appropriately your function.

- **(4pts)** 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.

- **(4pts)** Consider a mixture of 3 harmonically related sinusoids as the ones created in the previous question. What you would like to devise is a process to estimate the amplitudes assuming that you know the frequencies that the mixture is composed of. Try taking the inner product of a mixture with all the phases at 0 with a unit amplitude sinusoid of frequency f, then take the inner product of the mixture with a unit amplitude sinusoid of frequency $2f$. Observe what happens when the phase is not zero. Plot the inner products of the input mixture for all possible phase shifts of the probing sinusoid and see if you can figure out the correct phase for each component.

- **(4pts)**

  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. For testing use a linear combination of 3 sinusoids that are spaced harmonically and have different amplitudes. Plot the result of multiplying the input signal with the "closest" bin to the fundamental frequency of your input as well as the result of multiplying the input signal with the "closest" bin to the first harmonic. Now plot the multiplication of your input signal with a random basis function. Each bin corresponds to two basis functions one for the real part and one for the imaginary part. Show the point-wise multiplications of your input singal with both for each bin. Write 2-3 sentences about your observations.

  [**CSC575**] Measure how long it takes to compute 100 DFT transforms of 256, 512, 1024, and 2048 using your direct DFT implementation and compare your measured times with the ones using some implementation of the Fast Fourier Transform.

Question #2. (20 points)

Using an existing FFT implementation plot the DFT magnitude response in dB (using a DFT size of 2048) of a sine wave that is exactly equal to the center frequency of DFT bin 600. On the same plot overlap the DFT magnitude response in dB of a sine wave with frequency that would correspond to bin 600.5 (in a sense falling between the crack of two neigh- boring frequency bins). Finally on the same plot overlap the DFT magnitude rsponse in dB of the second sine wave (600.5) windowed by a hanning window of size 2048. Write 2-3 brief sentences describing and explaning the three plots and what is the effect of windowing to the mangitude response for these inputs.

[**CSC575**] Read data from a soundfile that contains a single note of a musical instrument. Plot the spectrum of 2048 samples during the steady state of the note after the initial attack with and without windowing. Plot the spectrum of 256 samples starting at the same location in time with and without windowing. Make an array of 2048 samples with the 256 samples from before followed by zeroes (zero-padding). Plot the spectrum of the 2048 zero-padded array with and without windowing. Write 2-3 brief sentences describing what you observe.

Question #3. (20 points)

The goal in this question is to extract a fundamental frequency contour from an audio file. Utilize the *qbh-examples.wav* audio file available on the course webpage for testing. You are welcome to implement this in any environment/language you like but it has been designed to provide a gentle introduction to *Marsyas* an open-source audio analysis framework specifically designed for Music Information Retrieval. In *Marsyas* algorithms are expressed as networks of processing objects with data passing through them (similarly to block diagrams in signal processing). It is possible to work directly in C++, or using Python bindings in Python, or use a custom-designed scripting language. Example code and some tutorials specific to this problem will be provided on the course website and G+ community.

- A simple fundamental frequency estimation is to compute the magnitude spectrum, select the highest peak, and return the corresponding frequency as the result. Processing the sound in windows will result in a time series of F0 estimates that can be plotted over time. In *Marsyas* the following *MarSystems* would need to be connected: *SoundFileSource*, *Windowing*, *Spectrum*, *PowerSpectrum*, *Selector*, *MaxArgMax* and *Accumulator*. Check that your method works by using as input a sine. Show the F0 plot for *qbh_examples.wav* with this method when using windows of 1024 samples at 22050 sampling rate.

- Copy your network and modify it to use *Autocorrelation* for the f0-estimation. Plot the new contour and compare it with the previous one. Is one consistently better than the other ?

- Change your code so that both the F0 estimates are computed for every window. Plot the sum of the two resulting F0 contours. The computation should be done in one pass through the audio file. In *Marsyas* use a *Fanout* after the *Windowing* followed by a *Sum*.

- [**CSC575**] Create a program for additive synthesis of 4 sinusoids. Create a sound with a missing fundamental for example sines at 200Hz, 300Hz, 400Hz, 500Hz and verify that you can perceive the pitch of the tone as the missing fundamental. Analyze the resulting sound using the two pitch detectors above and discuss your findings. In *Marsyas* you can use a *Fanout*, 4 *SineSources*, and a *SoundFileSink* to do the additive synthesis.

Question #4 (20 points)

Read the article "YIN, a fundamental frequency estimator for speech and music" by Alain de Cheveigne and Hideki Kawahara (available on the MIR course webpage under Course/Readings) and answer the following questions:

- The authors propose several steps for improving the F0 estimation accuracy of a standard autocorrelation-based method. What step had the biggest effect ?
- What was your favorite figure in the paper ? If you don't have a favorite pick one at random. Explain what you learn from this figure ?
- Where is interpolation used in the algorithm and why is it used ?
- **[CSC575]** Which was the hardest subsection to read and understand ? Try to put more effort into understanding it (read it again, consult the internet, read related papers etc). Do you feel you now understand it ? Don't just yes or no but provide some context around your answer. Do you think the authors could have written that subsection in a more readable way ? If yes how ?

Question #5. (5 points, A+)

Write code that shows animations of phasors and their sum. In addition to the examples we showed (real and imaginary axes projections, sum of phasors at the same frequency, sum of clockwise and counter-clockwise phasors of same frequency and amplitude) think of other interesting examples to show. I provide some suggestions but feel free to be creative, there is no need to implement all of my suggestions and feel free to contact me for any clarifications and advice.

Some possible ideas: sampling and aliasing, beating (the sum of two phasors with frequencies that are close to each other), and harmonics (multiples of the same fundamental frequency). Connect the phasor animation to audio input and output for example perform a DFT to incoming audio, select a few peaks and visualize them as rotating phasors or enerate sine waves for each animated phasor for synthesis. In order to make the motion visible use an appropriate "slow-down" factor for example a 200Hz audio signal could be visualized as a 0.2Hz phasor. Have multiple phasors plots and create fancy visualizations.

Question #6. (5 points, A+)

Make a complete F0 contour analysis and resynthesis system. The goal is to get a resynthesized signal from the F0 contour that sounds good and similar to the original. Examples of possible extensions to the basic F0 detection schemes you implemented above: some of the ideas from the Yin algorithm, energy detection used for modulation of the resynthesis so there are no random pitches when there is a pause in the input, median filtering of the pitch contour to reduce transient errors, dynamic adaptation of pitch range as the melody unfolds based on a maximum allowable interval. For resynthesis the calculated F0 contour can be used for example to drive a sine wave oscillator. Experiment with adding multiple sine wave oscillators to get more realistic sound. Add quantization to MIDI notes and detection of onsets and offsets. Any creative extension is fine and there is no need to implement all my suggestions just some. Feel free to contact me for clarifications and advice.