

**UVIC COMPUTER SCIENCE 475, Spring 2014  
ASSIGNMENT #1**

DUE February 03, 2014 (midnight PST)

There are 6 regular questions and each question is worth 15 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. (15 points)**

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. 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. (15 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 neighboring frequency bins). Finally on the same plot overlap the DFT magnitude response in dB of the second sine wave (600.5) windowed by a hanning window of size 2048. Write 2-3 brief sentences describing and explaining the three plots and what is the effect of windowing to the magnitude 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. (15 points)

Read the article “Music Retrieval: A Tutorial and Review” by Nicola Orio (available on the MIR course webpage under Course/Readings) and answer the following questions:

- Pick a piece of music that you like and try to characterize it using the time scales and music dimensions discussed in the paper. Try to be specific about the particular music piece.
- For the same piece of music describe specifically what types of information needs the different users mentioned in the MIR overview would be interested in.
- What is the difference between query-by-humming and query-by-example (1 paragraphs) ?
- How is MusicXML different from MIDI (1 paragraph) ?
- What part of the article was the most novel/interesting from your personal perspective ? Explain it to someone who has not read the article in your own words.
- [CSC575] Select 3 of the cited papers in the bibliography and read them (if you can not find a specific paper online or through the UVic library choose another one). Write a short summary (1-2) paragraphs for each of them as well as a paragraph of what you thought was the most interesting/novel part.

#### Question #4.

This question requires reading data from an audio file and writing data to an audio file. In almost all programming languages that I know reading data from audio files is either directly supported or there are libraries. It is also relatively straightforward to write your own input/output for uncompressed audio formats. Feel free to contact me if you have questions regarding what library to use for a particular programming environment. It also assumes that you have access to a FFT function.

- Write code that reads buffers of data from the audio file, converts them to a complex spectra using the Fast Fourier Transform, converts them back to the time domain using the inverse FFT and writes the output to a file. Confirm that things are working by checking that the output file is identical to the input i.e there is no loss of information from performing the FFT and inverse FFT.
- Convert the complex spectrum from real and imaginary coefficients to polar form i.e magnitude and phase. Replace the phases with random numbers, convert back to real and imaginary coefficients and perform the inverse FFT. What is the effect on the underlying audio ? Experiment with different sizes of window (256, 512, 1024, ..., 32768). Explain what you hear based on your understanding of the DFT in 2 sentences
- Perform a similar analysis to above but select the 4 highest peaks in the magnitude spectrum and set all the other magnitudes to zero while retaining the phases. Convert back to real and imaginary coefficients and perform the inverse FFT. What is the effect on the underlying audio ? Explain what you hear based on your understanding of the DFT in 2-3 sentences
- **[CSC575]** The use of windowing can reduce spectral leakage as explored in Q2. Adding windowing to the processes above however has an audible effect. Overlap-add is a procedure in which the analyzed windows overlap in time in such a way as to not introduce any artifacts in the reconstruction. This is typically achieved by having windows that add up to 1 when overlapping. Implement overlap-add analysis and resynthesis and check that the resynthesis is identical to the original. Then apply the two transformations (phase randomization and peak selection) described above. Write 2-3 sentences contrasting the results with the non-overlapping, rectangular window approach.

### Question #5.

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 #6 (15 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 #7. (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 generate 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 #8. (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.