ECE 5630: Digital Signal and Image Processing - Fast Convolution

Due: December 9, 2021

Name (Print):

Objectives

This programming assignment has three objectives:

1. Give you practice in using a program to perform digital signal processing.
2. Help to solidify your understanding of the DFT and its properties.
3. Understand the overlap-add and overlap-save methods for implementation of convolution.

Instructions

1. Use an FFT algorithm of your choice. Many are available on the web, or you can use the fft842 algorithm on the Canvas website.
2. The program should be written entirely in C or C++, except for standard libraries and the FFT algorithm (if contained in a binary library).
3. Use the `.wav` file `galway11_mono_45sec.wav` to test your program. This file is available on Canvas and is sampled at 11.025 kHz.
4. Use the Matlab tool `fdatool` to design your FIR filter.
5. For all of the frequency response plots, the normalized frequency running along the x-axis in your plot should be scaled to units of cycles/sample and should cover the range from 0 to 1.
6. For all of the magnitude frequency response plots (except the desired magnitude response), the y-axis should be scaled to units of dB by computing $(20 \log_{10} |H(f)|)$.
7. Include your work, answers, plots, and code within a `.pdf` document report and upload to the Canvas website. Include your observations and any other appropriate comments. The report should include an introduction, a section describing your approach and findings, and a conclusion.

---

1. Design a linear-phase FIR digital filter with the following characteristics:
   - 256 coefficients
   - Lowpass filter
   - Passband frequency of 300 Hz with unit gain.
   - Stopband frequency of 400 Hz with zero gain.
   - A signal sample rate of 11.025 kHz.

   (a) Plot the impulse response $h(n)$ of the filter.

   (b) Plot the desired magnitude response (as a gain of either 1 or 0), the actual magnitude response, and phase response.

2. In C or C++, write a program that performs the FIR filtering in the time domain. Your program should be written so that you can filter an infinite length signal.

   (a) What are the number of multiplies and adds per output sample?

   (b) Verify that your filter coefficients are correct and your filter routine works as expected by running sinusoids of amplitude 1.0 through your filter and finding the amplitude of the output sinusoid. (Remember to wait long enough that the filter transients have died down.) Do this for frequencies of $f = 10$ Hz, $f = 40$ Hz, $f = 150$ Hz, $f = 350$ Hz, and $f = 500$ Hz. Use a sample rate of 11.025 kHz.

   (c) Compare the computed magnitude response with the theoretical magnitude response (obtained from Matlab). Record the output amplitude for each of the input frequencies and plot them on a plot with the theoretical output amplitude.

3. In C or C++, write a program that performs the FIR filtering in the frequency domain using fast convolution and at least a 512-point FFT. Consider the following:

   - You may do **either** overlap-add or overlap-save in your program.
   - You only need to perform the FFT on the lowpass filter once.
   - The FFT must be a power of two.
   - The number of operations/output sample will vary as a function of the length of the FFT used.
   - Your program should be written so that you can filter an infinite length signal.

   (a) What are the number of multiplies and adds per output sample for both overlap-add and overlap-save?

   (b) Repeat 2(b)-(c) with the frequency domain filter program.

4. Use the Matlab function `wavread()` to generate the samples of the file `galway11_mono_45sec.wav`. Use your programs from 2. and 3. above to filter the sound file. The results should be the same for both programs.

   Time the execution of your programs. How does the time to execute the time-domain filtering compare to the time required for the frequency-domain filtering using the FFT?

   Does the filter remove the high frequency components? Do the processed files sound as you expected? Write out the final results in a .wav file for the TA to listen to. Make sure the samples are scaled high enough so that the signal can be played back with a comfortable volume level. Upload the files with your report in a `.zip` file.