Math 320 Programming Project IV - Fall 2018

Low Pass Filter

Please submit all project parts on the Moodle page for MAT320. You should include all necessary files to recompile, and a working executable, all in a zipped folder (one file for upload). Time-stamp determines the submit time, due by midnight on the due-date.

Due: Monday, Nov 26

Programs should compile under g++, so no Windows specific code is allowed.

1. Implement the filter in Chapter 4 section 3, which is graphed on page 67, Figure 3.1 but allow for the filter coefficient to be input on the command line. Also, normalize the output to -1.5 dB. Call the final program lowpass.cpp.

   The program should run on the command line, and take arguments for the filter coefficient a_1, an integer number n of times to run the filter, and a wave file:

   ./lowpass $< a_1 > < n > < input.wav >$

   Output should be a new wave file output.wav

2. Notes:

   (a) Use fourier_envelope.cpp for the input/output of wav files. Replace the middle of that program with the filter implementation. You should replace all the code from the Han Window comment to just before the convert to WAV file comment. You should note that the calculations use type casts in this part of the code, so you can borrow those for your filter implementation as well.

   (b) Normalize output to wave file:

   For convenience and testing (and listening) purposes, you should also normalize the output data to -1.5 dB. To do this, you need to pass through the data and find the largest absolute value M_1. Then take one more pass through the data to normalize so that this max value is changed to a value which is -1.5 dB of the maximum possible value in a 16 bit audio file. This value, say M_2 is the appropriate fraction of 2^{15} to achieve the reduction of -1.5 dB. So all data values are normalized by the factor M_2/M_1 and then output as 16 bit data, same type as input data.

   (c) The option to run the filter process n times gives an easy way to hear the effect of the filter. If it is run 100 times, feeding the output back in as new input, then it is easy to hear the lowpass effect.