521281S Application Specific Signal Processors LAB 1: Number formats and signal noise

In this lab you experiment impact of different numerical formats on digital filter behavior. We have an IIR filter and a real signal that we wish to process. The basic code for reading the file and filtering is provided and you should modify the code based on the given instructions to have a 16-bit float and 32-bit fixed point implementations. Finally, you are asked to examine the resultant signals in MATLAB environment to observe the impacts. The signal for this experiment is a wave file which is contaminated with a bandlimited noise that shows itself as peak in frequency domain as you can observe in the following figure. You can transfer the original and the processed signals to a device with a speaker like your phone and try to hear before and after the filtering versions.

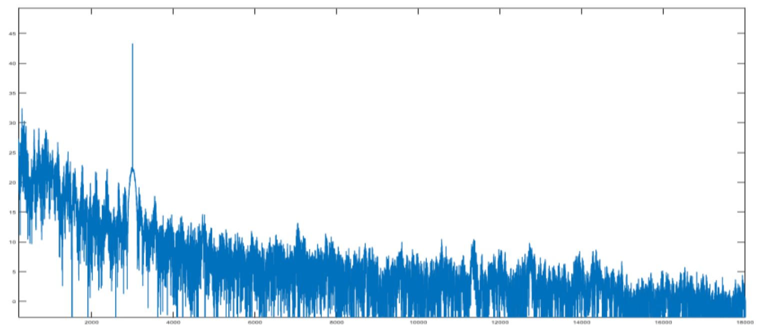

Fig. 1. Fourier transform of the signal, the large peak in the middle is the additive noise

You can save the file on a device with a speaker to be able to listen to the outputs. There is zip file in attachemnts that has three implementations of the same Butterworth IIR bandstop filter. The difference function of the filter is:

$$\boldsymbol{y}(n) = 1.0\boldsymbol{x}(n) + (-1.983)\boldsymbol{x}(n-1) + 1.0\boldsymbol{x}(n-2) - (-1.970)\boldsymbol{y}(n-1) - 0.987\boldsymbol{y}(n-2)$$

The three implementations of the filter implement the IIR computations in **32-bit floating point**, **16-bit floating point**, and **32-bit fixed point** number formats. However, only the 32-bit floating point filter implementation is complete. Read all three codes and considering the 32-bit floating point version as the reference, pinpoint the difference in the codes and briefly describe the difference.

1) Start the ASSP2015 virtual machine (VM) and extract the .zip file to a folder named lab1
2) In the VM, open up the *terminal* application and go to the folder lab1/float. Compile the filter by
3) `gcc -o filter *.c -lm`   //Note : switch -lm tells the compiler to link to math library
4) and run the filter with
   `./filter ../original.wav ../float.wav 144000 44`

The file flaot.wav is 32-bit float filtered version of original.wav

5) Navigate to the folder lab1/int32 and open up the file *filter.c*. Perform the necessary modifications to the code to complete the 32-bit fixed point implementation
   a. Compute the integer-valued filter coefficients; use scale 32768 (2^{15})
   b. Add the necessary shifting operations to the *filter* function
   c. Compile the filter as given above, but
   d. run it with `./filter ../original.wav ../int32.wav 144000 44`
6) Navigate to the folder lab1/half and open up the file *filter.c*. Perform the necessary modifications to the code to complete the 16-bit floating point implementation. Remember we have to convert the filter coefficients first to 16bit float version.

a. Use the tool lab1/half_conv to convert 32-bit floating point values to 16-bit floating point values (run by, e.g., ./half_conv 1.0)

b. Compile the filter as given above in 2), and

c. run it with ./filter ../original.wav ../half.wav 144000 44

7) Copy all the .wav files to Windows and start MATLAB

a. In MATLAB, open the file original.wav by xfo = wavread('original.wav'); (Newer versions of MATLAB does not support waveread, in this case, just drag&drop the file to MATLAB and rename the data to xfo)

b. Inspect the amplitude spectrum of xfo by plot(log10(abs(fft(xfo))));

c. Locate the amplitude value at 1000 Hz and write it to your notes

d. Do the same to the files float.wav, int32.wav and half.wav

e. Calculate the attenuation provided by each filter by computing the ratio of amplitudes at 1000 Hz (Bin number 3000): float32/original, int32/original, half/original

8) Still in MATLAB, repeat steps 6-a, 6-b, 6-c for each implementation (float, int32, …)

a. Visualize the filter amplitude response in MATLAB by the command fvtool([B0, B1, B2], [A0, A1, A2]); pick the values from the filter C code

b. What is the maximum attenuation provided by the filter?

c. What is the filter attenuation in dB at 1000 Hz (0.04167 on the x-axis)?

Note: Real frequency = **Index** * F_sampling / Length. So **1000 = 3000 * 48000 / 144000**

So 3000th bin is your 1000Hz frequency

Note: For half precision one, you need to convert back the coefficients to standard floating point format so that matlab understands the values. Simply use HALFToFloat(HALF y) function in "half_float.h" header file and printf the results something like this:

prinft("\n %f \n",HALFToFloat(B0))

d. Is there a difference between the maximum attenuation and the one at 1000 Hz? Why?

9) As a summary, how does the 32-bit fixed point format and the 16-bit floating point format compare to the 32-bit floating point format in this case?