Signals and Systems

(10920EECS202002)

Computer Homework #4: Discrete-time processing of continuous-time audio signals

(Make your own Karaoke and sound effect)

Due: 24:00, 06/16/2021

In this homework, you will practice DT audio signal processing, DT filter design, re-sampling of DT signals, reconstruction of CT audio signals, and experience audio watermarking. The provided sample codes – ComputerHW4_SampleCodes.m uses the following MATLAB built in functions:

audioread(), audiowrite(): read and write audio file.

fir1(): MATLAB's FIR filter design tool which provides the impulse response of the filter with the given specification. Note that different from our lectures, the frequency normalization in fir1() is done by normalization with Fs/2 (i.e., half sampling rate).

freqz(): Plot the frequency response of a filter, which can also be done by the approximated CTFT codes in the computer HW3 or by fft() (see the equivalent codes in the provided sample codes if interested). Note that different from our lectures, the frequency normalization in freqz() is done by normalization with Fs/2 (i.e., half sampling rate).

conv(): perform the convolution.

resample(): Re-sample the DT signals with which the sampling rate can be changed by a non-integer factor.

For the following problems, please refer to the corresponding MATLAB sample codes.

(CAUTION: MAKE SURE the volume is moderately low to avoid hearing damage. 請注意測試輸出音樂時，小心音量控制，避免傷到聽力。此音樂訊號最大的振幅(magnitude)為 1，人耳的靈敏度應該可以聽到 0.01-0.001 的聲音(端看你喇叭音量開多大)，請小心測試)

1. (10%) Read the music (sisiter_8sec.wav) using MATLAB **audioread()**. Plot the signal in time domain and its frequency-domain representation (i.e., magnitude spectrum) using MATLAB **fft()**. Remember to plot with the correct time and frequency axes. Can you find the signals representing drum beats in time or frequency domain? If you can, explain why the signals of drum beats present in such a form.

2. Filter the music. Apply the designed low-pass filter, high-pass filter or band-stop filter to the music and listen to find the differences.

(a) (20%) Use the MATLAB function **fir1()** to obtain the impulse response **h** of a low-pass (or a high-pass) filter. Change the filter order to 8, 16, 32, 64, 128, and 256, and then briefly describe the difference between their frequency responses (by MATLAB function **freqz()**) and impulse responses (by MATLAB function **stem()**). Plot the frequency responses and impulse responses to find the differences. From your observation, please also tell the idea how the **fir1()** performs the low-pass filter design (Hint: compare the ideal low pass filter and the filter designed by **fir1()** or see problem 5.55 in Handwriting HW4). Remember to check whether the designed filter works or not by comparing the signal spectra before and after filtering.

If you'd like to, you also can try band-stop filtering of the music to see if you can get rid of the vocal part and get a karaoke version of the music.

(b) (10%) Listen to your results carefully and answer the following questions. How differently do the low-pass and high-pass versions sound? Does the filter length affect the sound quality of the output? How and why? Note that the FIR filter length is the support length of the impulse response and is equal to the filter order + 1. Note that you had better vary the cut-off frequency of your filter in order to get better feeling of the processed music.

(c) (20%) Use fir1() to design a band-stop filter with which 0.5-kHz to 8-kHz sounds are suppressed. Convolve your filter with the provided audio clip 'sister_8sec.wav'. Can you hear the vocal part? Adjust the cut-off frequencies of the band-stop filter so that the vocal part you hear, is maximally suppressed while keeping the sound quality of the background music. Comment on how such a processing can provide you a karaoke version of the music.

3. Re-sample the given ECG signal and make it audible.

   (a) (20%) Given a MATLAB data file – ECG.mat where a raw ECG signal (i.e., ECG) and the sampling rate (in Hz, i.e., ECG_Fs) used to acquire the ECG signal are stored, resample the given ECG signal so that the resampled ECG signal has the same sampling rate as that of the music "sister_8sec.wav" in problem 2. Note that after re-sampling, the resampled ECG signal should has the same data length as the music "sister_8sec.wav" and the re-sampling should result in no aliasing in frequency domain. Detail your signal processing procedure for the re-sampling along with your codes. If in the procedure, you apply filtering, tell how you design the filter. Note that you can verify your results by using MATLAB resample().

(b) (20%) If you simply play the resampled ECG by your computer sound card and speaker (i.e., MATLAB codes – soundsc(ECG_resampled, Fs), you will find out you CANNOT hear the ECG. Figure out why you cannot hear the ECG sound and devise a signal processing procedure (i.e., a DT system) to make the processed ECG audible and justify your processing (see the in-class demo of the audible ECG). Remember to output your audible ECG to "AudibleECG.wav".

Notice:

1. Please hand in your solution files to the LMS elearning system, including your word or pdf file of the detailed solutions, the associated Matlab codes, and all the related materials. It would be nice that you can put your "KEY" code segment with comments side by side along with your answer in the word or pdf file whenever necessary.
2. Name your solution files "EECS2020_HW4_StudentID.doc" (or "EECS2020_HW4_StudentID.pdf") and "EECS2020_HW4_StudentID.m", and archive them as a single zip file: EECS2020_HW4_StudentID.zip.
3. The first line of your word/pdf or Matlab file should contain your name and some brief description, e.g., % EECS2020 Calvin Li u9612345 HW4 MM/DD/2021