Lab 4: Static & Switched Audio Equalizer

Professor Deepa Kundur

Objectives of this Lab

The goals of this lab are:

- To introduce audio processing with the ultimate goal of real-time implementation.
- To develop intuition on frequency bands of audio signals and auditory perception, through a case study of audio equalization
- To provide exposure to the application of filter design techniques for audio processing.

Prelab

Prior to beginning the lab, you must:

- Carefully read over this lab in order to plan how to implement the tasks assigned. Please highlight all the parts you need to show or answer for the TA, so that you do not miss any graded points during the in-lab component or for the report.
- Design your equalizer frequency bands as discussed in the Prelab section.
- Select an audio source you wish to bring for the lab to test out your equalizer. MP3 players with your favorite music will work well.

Deliverables

- During the lab, you must show your TA the Simulink results discussed in the lab instructions; and
- After the lab, each group must submit a separate report answering the lab questions.

Grading and Due Date

The lab will be graded on correctness, comprehensiveness and the insight you are able to provide when answering the questions. For full points, lab reports should be written in complete sentences with correct grammar.

Please note STRICT DEADLINE for report on the course web site.

Lab 4: Static & Switched Audio Equalizer

Introduction and Background

You just got a set of new wheels (well pre-owned vehicle), and you're cruising, when you come to a stop at the lights. The car next to you is blasting out nothing but bass, shaking everything around it, including your ride. You try to turn up your own stereo, but the car next to you has BOSE^{TM} while your audio system just blows. Fortunately you have taken this DSP course before, and you quickly (and embarrassingly) drive back home, and pull out your DSP kit, which has been gathering dust since last year.

You've already implemented digital filters in the past labs. From your experience, it should not be difficult to imagine how you might boost the bass of your audio system. Figure 1 presents an ideal design for boosting the bass of continuous-time audio signals. Like in the past labs, you have to create a practical *digital* filter that mimics Figure 1. You can do this in many ways. For example, you can design the practical filters for continuous-time signals first, and then use the bilinear transform to obtain the discrete-time filter.

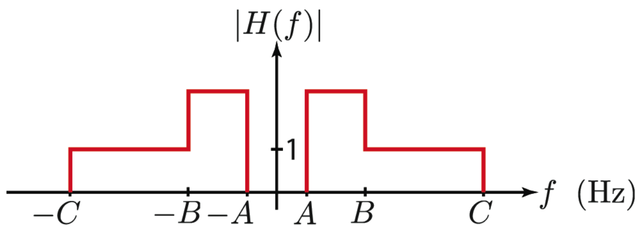


Figure 1: Magnitude Response of Ideal Filter (for Continuous-time Signals) for Bass Boost (2-band Adjustable)

In Figure 1, cutoff frequency A (Hz) is used because the human auditory system has limited dynamic range. That is, the human ear cannot pick out frequencies lower than A (Hz), unless you have snake ears. Similarly cutoff frequency C (Hz) characterizes the highest pitch perceivable to humans – again unless you have the ears of a dog or bat, you don't need to pass higher frequencies. The lower frequency band from A to B (Hz) characterizes the bass part of the audio signal. Note that the gain in this region is greater than 1 (or greater than 0-dB), which means you are boosting this part of the band, i.e. the bass. On the other hand, the band from B to C (Hz) has a gain of 1 (or 0-dB), which means you're passing the audio signal in this band as is. If you wish to turn down the non-bass part of the audio stream (like a lot of people do when they are cruising), the gain would be less than 1 (or negative-dB) in this region.

Figure 1 can be interpreted as consisting of *two bandpass* filters. The first bandpass filter is the A to B (Hz) band, while the second bandpass filter is the B to C (Hz) band. In practice, non-ideal bandpass filters used in tandem may *bleed* into one another, which is all right for audio signals, as the human ear may not perceive this. Since you are dealing with linear systems, you can use the principle of superposition to process each of the regions separately, and then recombine them as show in Figure 2.

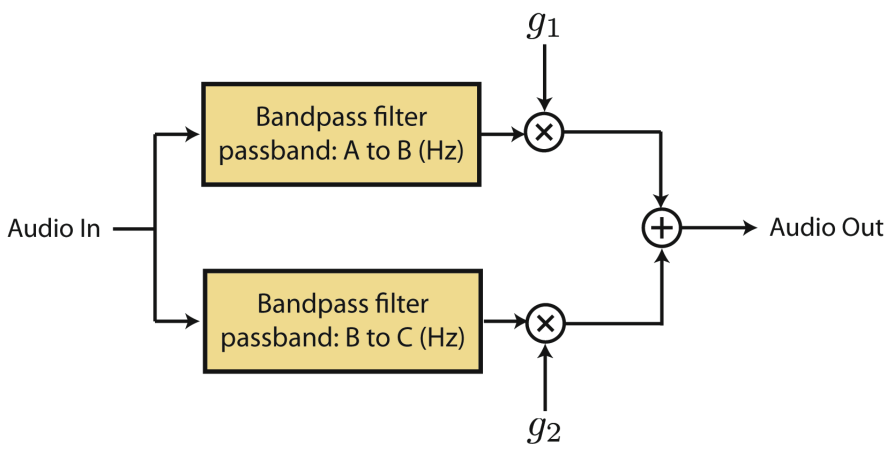

g_1
Bandpass filter passband: A to B (Hz)
Audio In
Audio Out
Bandpass filter passband: B to C (Hz)
g_2

Figure 2: Implementation of Equalizer of Figure 1 using Superposition.

In Figure 2, the audio signal is filtered using two bandpass filters, and the gain on each filter is then adjusted using g_1 and g_2. For example, in our previous bass boost example, we might have $g_1 = 3$, and $g_2 = 1$, giving the bass 3 times more power, while keeping the rest of the frequencies the same. A more advanced audio equalizer would have a number of bandpass filters focusing on disjoint frequency bands, for example partitioning the band more finely (i.e., with higher "granularity") to allow more control to the user. For this lab, you will be controlling 5 bands.

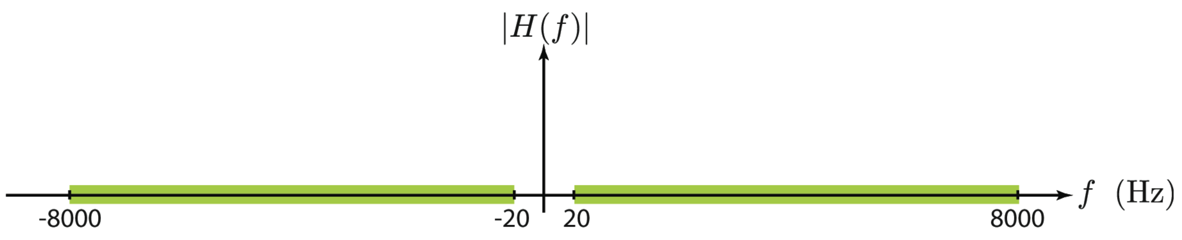
So, when you are playing with the equalizer on your stereo, twiddling the slider up and down for different bands, you are simply controlling the gain on each of the bands, which can be filtered out and identified using bandpass filters. When you program your DSP in this lab, you will hardcode the gains first to make sure your filters are working. In the next lab, you will use the switches on the DM6437 board to control your equalizer.

Prelab: Equalizer Band Design

The following should be completed prior to the lab, but it should be presented with the lab report (submitted after the lab by the deadline). The first phase in the equalizer design process is to decide how to split up the audio frequency range into bands. This can be done in two simple steps:

Step 1: Determine the *processing band* of your audio signal.

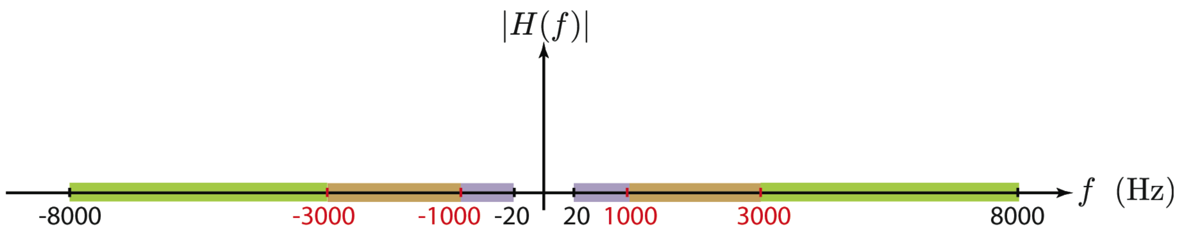
The perceptible audio range is roughly 20 Hz to 20 kHz. Assuming the DSP employed works at a sampling frequency of F_s Hz, an anti-aliasing filter with cutoff frequency $0.5F_s$ Hz would be applied to the analog audio signal prior to sampling. Thus, no frequencies above $0.5F_s$ Hz would be present. Therefore, the processing band ranges from 20 Hz to $\min(0.5F_s, 20{,}000)$ Hz. For example, for $F_s = 16{,}000$ Hz, the processing band is shown in the following figure:



The sampling frequency F_s to be used with the DM6437 for audio processing is 8 kHz.

Step 2: Determine the equalizer bands (i.e., the non-overlapping frequency bands to independently control via the equalizer).

There are an endless number of ways that you can divide the processing band. Assuming we are dividing the processing band into L bands, one approach is to equally partition them into L non-overlaping bands of width $(\min(0.5F_s, 20{,}000) - 20)/L$ Hz each. Another more popular approach suited to the human auditory system is to have bands that increase in width by approximately a factor of two. For example, following the example of Step 1, if we choose to partition into $L = 3$ bands, one possible division is shown below:



In this lab, we let $L = 5$. **Please specify 5 non-overlapping equalizer frequency ranges to be used in this lab.**

Now, we are ready to use MATLAB's Filter Design & Analysis Tool during the lab.

Design and Implementation

Static Equalizer

In this section you will design an equalizer model in Simulink that has non-adjustable gains for each audio band. You will run this model on the DM6437 board to make sure your basic design is sound.

1. The first step in the design process is to decide how to split up the audio frequency range into bands. Use your prelab results in which you divided the processing band into five reasonable passband frequency ranges for the audio equalizer.
2. Type fdatool into the MATLAB command window to bring up MATLAB's Filter Design & Analysis Tool. Design five passband filters for each one of your chosen frequency bands. You may choose any filter style you desire. Note that you can store multiple filters in fdatool. To store a filter, click on the Store Filter button and type in a name for the filter. To view previously stored filters, click on the Filter Manager button and choose the desired filter. Here are some hints on how to proceed with your filters:
    a. Using a minimum order filter (rather than specifying the order of the filter) will give you more control over the frequency characteristics of the filter.
    b. You probably want to make the transition between passband and stopband sharp in order to more closely mimic the rectangular bands of an ideal equalizer.
    c. The total filter order should be no more than 120 to prevent latency (delay) in the audio output. Playing around with filter types, stop band attenuation, and pass/stop band sharpness will allow you to build an equalizer with a total order less than 120.
    d. Making the Fpass frequencies of adjacent bands equal will ensure that all frequencies are contained within one of the five passbands (i.e. no frequencies near the juncture of two bands will be left in an attenuated transition region).
3. Open a new Simulink model and export your filters from fdatool to the model.
4. As usual, add the input and output 6437 blocks (ADC/DAC and data conversion blocks) and the Target block. In the block parameters for the ADC block, change the ADC source to Line In and the sample rate to 8 kHz.
5. Using the basic superposition structure shown in Figure 2, implement your equalizer in the Simulink model. Nominally, you should use either gain blocks from Simulink → Math Operations or dB Gain blocks from Signal Processing Blockset → Math Functions → Math Operations as the multiplication factor for each band. For now, set the gain of your block so that it is equivalent to a unity gain.
6. Modify the Configuration Parameters if necessary to implement your model on the board (See Lab 3 if you don't remember these settings).
7. Set up CCStudio and build, load and run your model as usual.
8. Use any audio source you wish and input it to the board. Connect the output to speakers and make sure you can hear the audio source. The output should sound "normal" in that all frequencies are passed equally.

9. Halt your program and go back to your Simulink model. Adjust some of the gains to cut out or amplify a couple of the frequency bands. Rebuild the model, reset the board, and load and run the new program. Verify that the audio source sounds different with the new set of gains. Depending on your audio source and how well you chose your gains, you may have to go back and change the gains several times to hear any significant difference. Note that you'll have to rebuild, reset, load, and run each time.

Variable Equalizer

In this section you will use multiple gains attached to switches to vary the filtered output of the DM6437 DAC without having to rebuild and load each time. This will be done using the four grey switches located on the edge of the 6437.

1. Insert into your static equalizer block diagram four DIP switches located under Target Support Package → Supported Processors → Texas Instruments C6000 → Board Support → DM6437.
2. Set the sample time for each DIP block to -1 and change the switch that each block controls to whatever you would like for it to be (the switch numbers are designated on the board).
3. Place four multi-port switches into the block diagram and change the input of each to 2.
4. The top pin of each multi-port switch is the control pin. Connect the output of each DIP switch to the control pin on each multi-port switch.
5. Connect a gain block to the input of each multi-port switch and connect the output of each filter to the gain blocks feeding into the switches. This is an example of what each switched filter will look like:

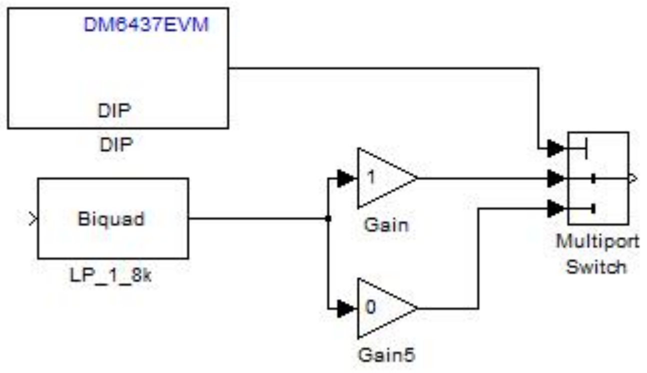


Figure 5: Switched filter example

6. There are five filters but only four switches. Attaching two filters to a switch or leaving a filter un-switched will fix this problem. Play around with this and find an audio equalizer you prefer.

Questions

1. Earlier, we mentioned that due to the non-ideal nature of the filters (i.e. they are not perfectly rectangular in the frequency domain), the side-lobes of the filter in the frequency domain would cause frequencies in the stop-band to be passed, albeit attenuated. We mentioned that for audio applications, this is ok, since the human audio system, to a certain degree, cannot perceive that the non-ideal filters let some frequencies through, when they shouldn't be. In this question we will investigate how much of the frequencies in the stop-band are passed when window-type filters are used. We shall look at two very similar window filters: Ha<u>nn</u>ing and Ha<u>mm</u>ing. The Hanning window is given as $w_n = 0.5 - 0.5\cos(2\pi n/M)$ for $n = 0, \ldots, M$. The Hamming window is given by $w_n = 0.54 - 0.46\cos(2\pi n/M)$ for $n = 0, \ldots, M$. Note that these filters are all discrete-time filters. Thus when we speak of cutoff frequencies, we shall give them as normalized frequencies between 0 and 2π. Notice that the only design parameter is M. For our purposes, we will assume that the passband is encompassed by the main-lobe, while side-lobes represent the stop-band.
   a) Design a Hanning window whose cut-off (normalized) frequency is $\pi/4$.
   b) What fraction of energy is passed in the stop-band for the Hanning window design in part a)?
   c) Design a Hamming window whose cut-off (normalized) frequency is $\pi/4$.
   d) What fraction of energy is passed in the stop-band for the Hamming window designed in c)?
   e) What is the ratio of the peaks of the main side-lobes (first side-lobes) of the magnitude responses for the two windows above?
   f) Which of the above two windows would you prefer to use?

   Please provide your MATLAB code (or any other means) used to generate the results as well as the reasoning for your results. Good luck!