Lab: High-pass and low-pass filters on audio signals

1 Introduction

In Homework 8 (Circuits, Laplace Transforms, Frequency Response, and Sensors), we look at the behavior of RC, RL, and LC circuits as filters. This lab investigates the effect of high-pass, low-pass, band-pass, and notch filters on audio.

When filters are applied to audio signals, certain frequencies of sound are attenuated. Music professionals use more complicated filters and equalizers to change the balance of instruments or to eliminate an unwanted sound. We will apply simpler filters to music using MATLAB.

2 Prelab

2.1 Motivation

Open your preferred music player on your computer (e.g. iTunes, Winamp, MediaMonkey, Windows Media Player, etc.) and learn how to use the equalizer. Depending on your application, you probably have some sliders, and a list of presets such as "pop," "rock," "jazz," "classical," etc. Try different presets and listen. Look at how the sliders move when you select different presets. Then try changing those as well. What happens when you slide a group of them all the way to zero?

2.2 Preparation

If you have a laptop with MATLAB, bring it to lab. Also bring headphones if you have them.

The following steps apply to those who are bringing their laptops. Download the following files from the course website: `me161filter.m`, `timeDomainPlot.m`, `lab_annoying.wav`, and `lab_bandpass.wav` from the course website, `me161.stanford.edu`. If you want to experiment with filters on your own music, convert the song(s) to WAV format and clip them to 30 seconds or less. (Longer songs can work, up to a limit, but will take too long for the purposes of this lab.)

3 Experimental

Launch MATLAB and change your current directory to the folder containing `me161filter.m` and the music files. Type the following lines into the command prompt:

```
global player
global stopped
timeDomainPlot('lab_bandpass.wav')
```

You should hear music playing and see a time-domain plot of the sound's amplitude.

Now view the spectrogram (frequency-domain plot) for the same music clip:

`frequencyDomainPlot('lab_bandpass.wav')`

Note: The lowest frequencies might be inaudible if you are not using good-quality headphones.

3.1 Syntax

`timeDomainPlot(filename)` shows a time-domain plot of the sound's amplitude.

`timeDomainPlot(filename, filter_type, w_cutoff)` shows a time-domain plot of the sound's amplitude, with a filter applied to the sound.

`frequencyDomainPlot(filename)` shows a frequency-domain plot of the sound's amplitude.

`frequencyDomainPlot(filename, filter_type, w_cutoff)` shows a frequency-domain plot of the sound's amplitude, with a filter applied to the sound.

Possible filter types are `'low'`, `'high,'` `'bandpass,'` and `'stop.'` If you're using a low-pass or high-pass filter, `w_cutoff` is a scalar; if you're using a bandpass or stop-band filter, `w_cutoff` is a two-element array in the form `[w1 w2]`.

Note: To stop a song early, hit the 'Stop' button on the lower-left corner of the plot. If you close the plot in any other way, you must kill the script from the command prompt with CTRL+C, and pause the audio with `pause(player)`. Also, if you do not have the global variables `player` and `stopped` declared, the 'Stop' button won't work.

3.2 Low-pass filters

Run the `frequencyDomainPlot` script for the music clip `'lab_annoying.wav')`. Note the sound and the frequency range. Use a low-pass filter to completely eliminate the annoying high-pitched sound. (This will take a few iterations of guess-and-check.) What value of `w_cutoff` did you use?

3.3 High-pass filters

Run the `frequencyDomainPlot` script for the music clip `'lab_annoying.wav')`, with a high-pass filter using the same `w_cutoff`. What happens?

3.4 Band-pass filters

Run the `frequencyDomainPlot` script for the music clip `'lab_bandpass.wav')`, Then use a band-pass filter to isolate the midrange instrument. (Remember, you need a two-element array for `w_cutoff` now.) What values work well?

Warning: Do not go below `w_1 = 150`.

3.5 Notch/band-stop filters

Run the `frequencyDomainPlot` script for the music clip `'lab_bandpass.wav')`, using a band-stop filter to remove the midrange instrument. What values for `w_cutoff` work well?

3.6 Time domain

Now apply the same band-pass and band-stop filters to `'lab_bandpass.wav')`, but use the time domain plot instead. Is this helpful? Why did we design our filters with the frequency domain visualization instead?

4 Analysis

In this lab, we actually used a fifth-order Butterworth filter, which has a very flat frequency response in the passband, and drops off very rapidly to the stopband. If you were to implement a low-pass filter using a simple RC circuit, the relationship between gain and frequency might look like this:

$$G = \frac{1}{\sqrt{1 + (\omega RC)^2}}$$

The cutoff frequency ω_c is defined as the frequency at which the gain is reduced to $\frac{1}{\sqrt{2}}$.

Imagine you are implementing a simple RC circuit to get rid of the annoying high-pitched sound in the first part of this lab. Given the above equation relating G, ω, R, and C, find values for R and C that would give your low-pass filter the desired cutoff frequency.

Plot G as a function of ω, using the values of R and C you just found. The horizontal and vertical axes should both be log_{10}-scaled. How is this filter different from the Butterworth filter we used in the lab?