Individual Final Project: Audio Filters

ECE3101 - Signals and Systems

Sam Duong

May 2nd, 2024

Professor Johanna Garcia

Introduction

Wavelets are mathematical functions that divide an input signal into different scale components. These are small functions. From that, wavelet transforms are the representation of a function by the wavelets. These are scaled and translated from a finite-length signal known as the mother wavelet. In DSP, wavelet transforms are advantageous as audio filters over Fourier transforms because of the nature of audio signals containing discontinuities and sharp peaks. The wavelet transform results in analyzing a signal into different frequencies at different resolutions. The width and central frequency of the wavelet can be adjusted, a process called scaling. An expanded wavelet will resolve low frequency components of the input signal with bad time resolution, and a shrunken wavelet will resolve high frequency components of the signal with good time resolution. The discrete wavelet transform is a much more compact version of the wavelet transform as it only samples a certain sequence of the signal.

Objectives

In this project, I will run a wavelet filter over an audio signal and observe how this affects the input signal in the output. With the MATLAB simulation, I will be able to input any input signal and run it through a wavelet filter. Note that in order for the code to work, the input signal used must be present in the computer. I will email you a sample audio file, or you may choose to run an audio file of your own choice.

Mathematical Explanation

The Wavelet transform looks like this:

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-inf}^{inf} f(t)\psi * (\frac{t-\tau}{s})dt$$

Note that the mother wavelet, the function that all wavelets are derived from, is Ψ(t).

The signal f(t) is multiplied by the conjugate of the wavelet. The parameters τ and s represent time localization and scale parameter (1/frequency). As mentioned earlier, the s is used to scale and compress the waveform.

The Discrete Wavelet transform looks like this:

$$D[a,b] = \frac{1}{\sqrt{b}} \sum_{m=0}^{p-1} f[t_m] \psi[\frac{t_m - a}{b}]$$ where $a = k2^{-j}, b = 2^{-j}$. Note τ and s from the Wavelet transform become a and b.

MATLAB Simulation

```
[sample_data, sample_rate] = audioread('Cal_Poly_Pomona_SDECE3101.wav'); % load the audio sample


%plot signal in time and the amplitude of its frequency components using
%the FFT


sample_period = 1/sample_rate;
t= (0:sample_period:(length(sample_data)-1)/sample_rate);
subplot(2,2,1)
plot(t,sample_data)
title('Time Domain - Unfiltered Signal')
xlabel('Time (seconds)')
ylabel('Amplitude')
xlim([0 t(end)])


m = length(sample_data);
n= pow2(nextpow2(m));
y=fft(sample_data, n);
```

```
f=(0:n-1)*(sample_rate/n);
amplitude = abs(y)/n;
subplot(2,2,2)
plot(f(1:floor(n/2)),amplitude(1:floor(n/2)))
title('Frequency Domain - Unfiltered Signal')
xlabel('Frequency')
ylabel('Amplitude')

Fs = sample_rate; %sampling frequency (hz)
Fn = Fs/2; %nyquist frequency
Wp = 1000/Fn; %passband frequency (normalised)
Ws = 1010/Fn; %stopband frequency (normalised)
Rp = 1; %passband ripple (dB)
Rs = 150; %Stopband ripple (dB)
[n,Ws] = cheb2ord(Wp,Ws,Rp,Rs); %Filter order
[z,p,k] = cheby2(n,Rs,Ws,'low'); %Filter Design
[soslp,glp] = zp2sos(z,p,k); %convert to second-order-section for St

figure(3) %filter bode plot
freqz(soslp,2^16,Fs)

filtered_sound = filtfilt(soslp, glp, sample_data);
sound(filtered_sound, sample_rate)

t1 = (0:sample_period:(length(filtered_sound)-1)/sample_rate);
subplot(2,2,3)
plot(t1,filtered_sound)
```

```
title('Time Domain - Filtered Signal')
xlabel('Time (seconds)')
ylabel('Amplitude')
xlim([0 t1(end)])

m1 = length(sample_data);
n1 = pow2(nextpow2(m1));
y1=fft(filtered_sound, n1);

f = (0:n1-1)*(sample_rate/n1);
amplitude = abs(y1/n1);
subplot(2,2,4)
plot(f(1:floor(n1/2)),amplitude(1:floor(n1/2)))
title('Frequency Domain - Filtered Sound')
xlabel('Frequency')
ylabel('Amplitude')
```

NOTE THAT 'Cal_Poly_Pomona_SDECE3101.wav' MAY BE ANY OTHER INPUT SIGNAL. IF YOU WOULD LIKE TO TEST THE FILTER WITH ANOTHER AUDIO FILE, REPLACE THAT TEXT WITH THE NAME OF THE DESIRED AUDIO FILE.

Click on this link for the audio file. Cal_Poly_Pomona_SDECE3101

Conclusion

After running the code, the audio output was noticeably different when compared to the audio input. The output seemed 'muffled' compared to the input signal. From this, it was easy to conclude that the filter worked as intended in removing some noise from the signal.