Chen Ji
SUNetID:jjjjjc

Homework 5a

Music424
May 20, 2019

Problem 1

(a) A linear phase filter delays all frequency components the same amount τ,

$$h_{linear}(n) = F^{-1}\{e^{-j\tau\omega}|H(\omega)|\}$$

In order to return the impulse response of the associated linear-phase filter, the MATLAB is shown below:

```
function LinPhaseIR= tf2linearIR(tf, L)
    % tf is a magnitude response |H(z)|
    % L is the length of its impulse response h[n]
    w = [ 0 : L-1 ]*2*pi/L;
    tau = L/16;
    LinPhaseIR = real(ifft(exp(-1i*tau*w).*(abs(tf)')));
    LinPhaseIR = LinPhaseIR( 1 : L/2-1 );
end
```

(b) Before converting the impulse responses to linear phase, the original impulse response of "Jen57On.wav" is shown below:

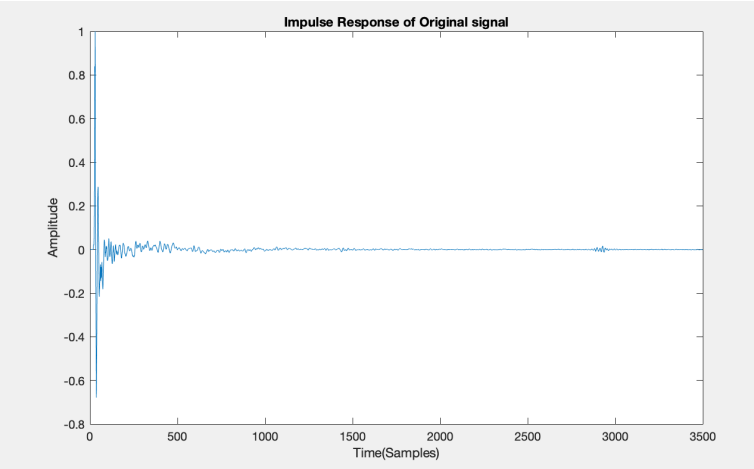

Figure 1: Original Impulse Response of Jen57On.wav

Then I wrote the MATLAB script as below to convert it into linear phase:

```
% Read file, get length:
[h, fs] = audioread('Jen57On.wav');
L = length(h);
% Convert to frequency domain:
```

```
5 H = fft(h,L);
6 % Convert to linear phase:
7 linPhase = tf2linearIR(H,L);
8 % plot the impulse response
9 plot(linPhase)
```

Then the impulse response in linear phase version can be plot as below: It can be seen

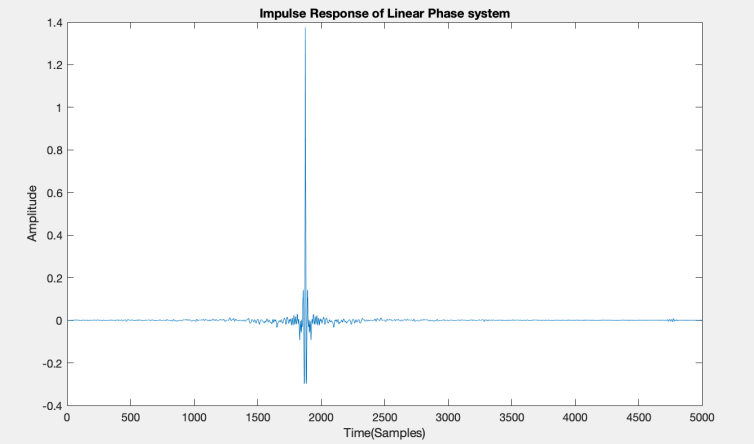

Figure 2: Linear phase version of Impulse Response of Jen57On.wav

from two plots, the linear version of the impulse response is symmetric while it is not for the original one. Also, wen can find the frequency response of two cases of impulse response after doing FFT as below:

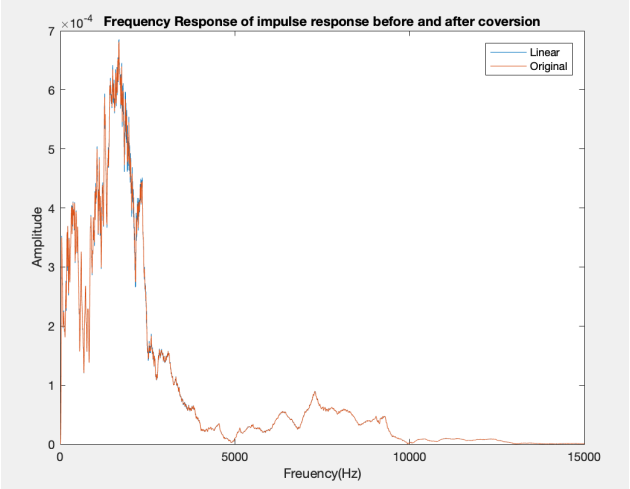

Figure 3: Linear phase version of Impulse Response of Jen57On.wav

Basically the magnitude frequency response of two versions are the same, so the conclu-

sion that converting impulse response to phase response will not influence the magnitude frequency response.

(c) The linear phase version sounds fuzzier in the midrange frequencies. There is more roundness to the tone, whereas the original filter sounds comparably crisp and direct. These differences must be due to phase, as the frequency responses match.

Problem 2

(a) To convert h(n) to minimum phase, the equation below can be used:

$$h_{\min} = \mathcal{F}^{-1}\left\{\exp\left[\log|H(\omega)| - j\mathcal{H}\{\log|H(\omega)|\}\right]\right\}$$

Then the MATLAB implementation is shown below:

```
function MinPhaseIR = tf2MinPhaseIR(tf, L)
% tf is a magnitude response |H(z)|
% L is the length of its impulse response h[n]
MinPhaseIR = real(ifft(conj(exp(hilbert(log(abs(tf')))))));
MinPhaseIR = MinPhaseIR(1:L);
end
```

(b) The following MATLAB script uses the function above to convert the measured impulse response to minimum phase:

```
% Read file:
[h, fs] = audioread('Jen57On.wav');
% Get length:
L = length(h);
% Convert to mag:
h = h/max(abs(h));
H = fft(h,L);
% Convert to LP:
minPhase = tf2MinPhaseIR(H,L);
% Get new mag:
minPhase = minPhase /max(abs(minPhase));
H1 = fft(minPhase);
```

It seems really similar to the original one, so I zoom in the region near zero, then the impulse response can be shown in Figure 5 that the minimum phase impulse response is more centered to the zero, which means the energy will be concentrated at the beginning.

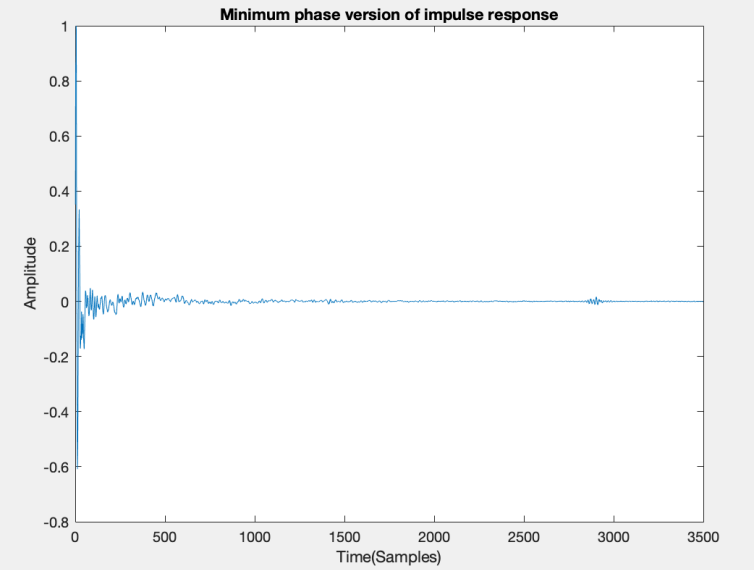

Figure 4: Minimum phase version of impulse response

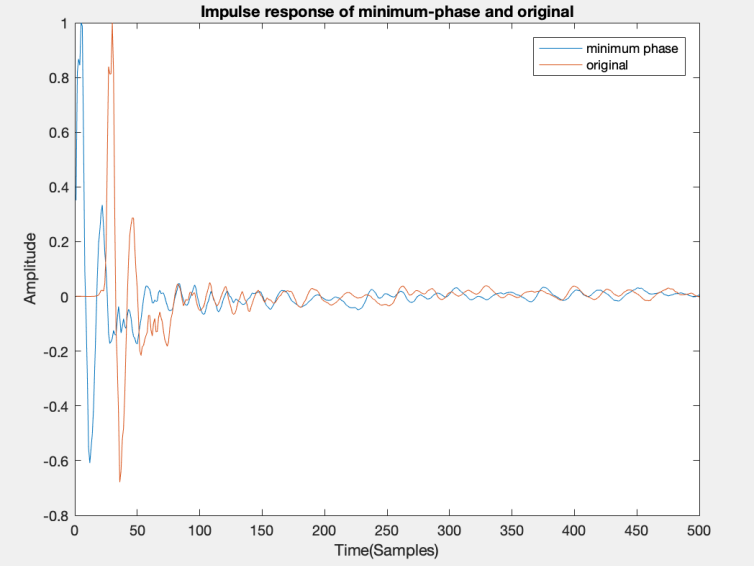

Figure 5: Comparison of minimum phase and the original

By comparing the magnitude frequency response, two versions' are the same, which means that the conversion from original impulse response to minimum phase will not change the magnitude response.

(c) Although they are very similar, the phase-altered filter exhibits a bit less clarity in the mid-range frequencies. It has a somewhat fuzzier, thicker, less crisply defined presence.

Homework 5a

Problem 2

(a) By smoothing the filter power over bandwidths proportional to critical bandwidth, filter complexity is reduced while retaining psychoacoustically relevant cues. Critical-band smoothed filter power may be computed via a running mean over frequency:

$$P(\omega;\beta) = \sum_{\varphi=f(b(\omega)-\beta/2)}^{f(b(\omega)+\beta/2)} |H(\varphi)|^2$$

```
function smoothed_out = cbsmooth(tf,beta,N)
% Constants:
fs = 44100;
halfFs = fs/2;
HsQ = abs(tf(1:(N/2) + 1)).^2;
% Pre-allocate:
Out = zeros(size(HsQ)+1);
% Main averaging loop:
for i = 1 : (N/2) + 1
% Center of the process block:
centerHz = i * halfFs / (N/2) / 1000;
% Convert to Erb:
centerBark = khz2erb(centerHz);
% Define lower bound:
lowBound = erb2khz(centerBark - beta/2);
% Round lower bound:
floor = max(1, round(lowBound*1000*(N/2)/halfFs));
% Define upper bound:
highBound = erb2khz(centerBark + beta/2);
% Round upper bound:
ceiling = min(N/2, round(highBound*1000*(N/2)/halfFs));
% Average between two bounds:
Out(i) = mean(HsQ(floor:ceiling));
end
% Return output:
smoothed_out = Out;
end
```

(b) The following MATLAB script was written to smooth the impulse response with values of beta = [0.5, 1, 2, 5] using the function cbsmooth().

```
% Constants:
beta = [0.5 1.0 2.0 5.0];
N = 8192;
```

```
% Read file:
[h, fs] = audioread('Jen57On.wav');
% Get length:
L = length(h);
% Convert to mag:
h = h/max(abs(h));
H = fft(h,N);
figure(1);
subplot(5,1,1);
plot(h);
figure(2);
subplot(5,1,1);
plot(20*log10(abs(H)));
for i = 1 : length(beta)
% Convert to LP:
smoothedH = cbsmooth(H,beta(i),N);
% Get new mag:
smoothedH = smoothedH /max(abs(smoothedH));
H1 = fft(smoothedH);
% Plot:
figure(1);
subplot(5,1,i+1);
plot(smoothedH);
xlabel(['Beta = ' num2str(beta(i))]);
figure(2);
subplot(5,1,i+1);
plot(20*log10(abs(H1)));
xlabel(['Beta = ' num2str(beta(i))]);
hold all
end
```

The smoothing plot can be shown as below: The above plots are in dB unit for y axis.

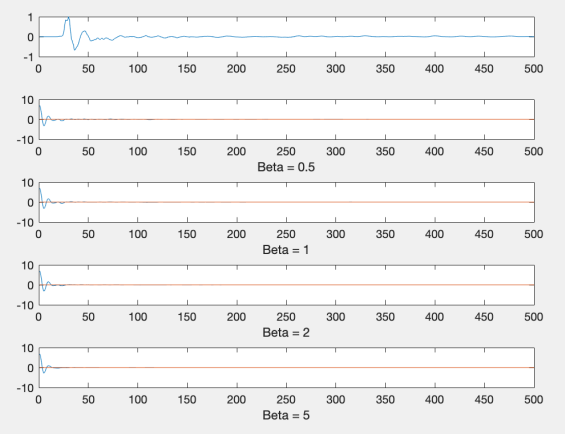

Figure 6: Plot of critical band smoothing

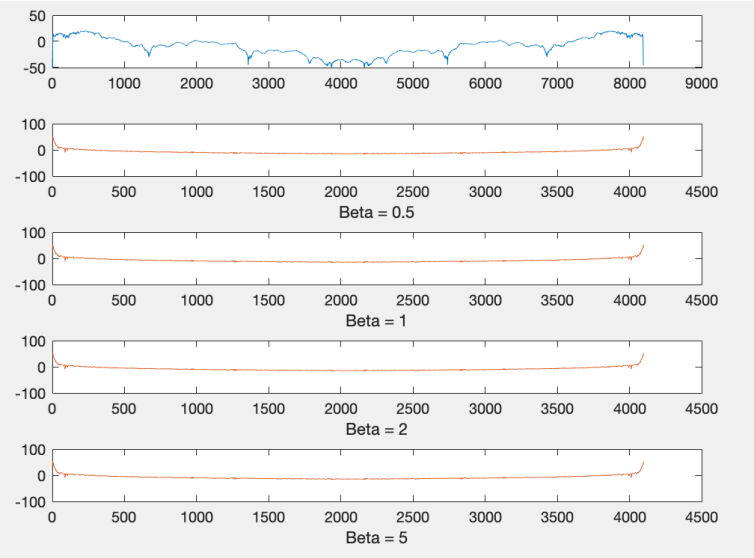

Figure 7: Plot of critical band smoothing as spectrum

(c) By applying the minimum-phase transfer function, the plot can be shown as below:

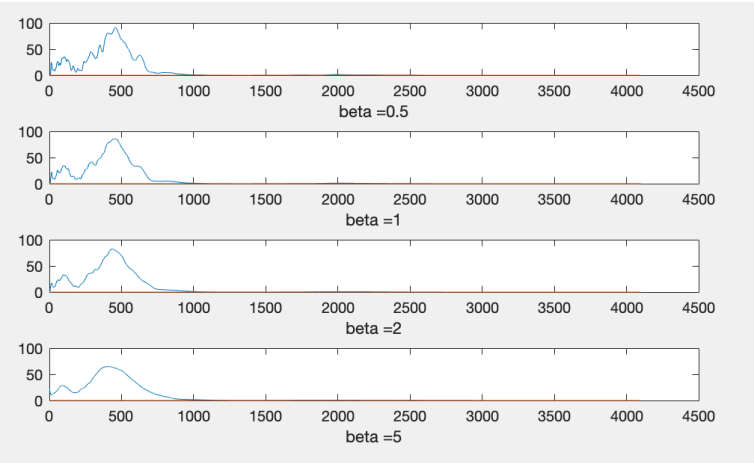

Figure 8: Plot of critical band smoothing after minimum-phase transform

Compared to the Figure 6, they are almost the same. After applying this smoothing to the guitar sound, the distorted guitar sound a bit clear.