Problem Set 4 Solutions

Problem 1. [30 Points]

1(a). [20 Points] The matlab scripts hw4q1.m and linphase.m are shown below.

```
% hw4q1.m
% Music424 Homework #4
% Spring 2008
% SongHui Chon

% Problem 1
% linear phase impulse response
% based on page 281 of the lecture notes

% Part (b)

clear all;
load ps04.mat;

ir = ir / max(abs(ir));
Nfft = 8192*4; % fft size in samples
tau = Nfft/16; % time constant in samples, to make it a linear phase filter

h_lin = linphase(ir, Nfft, tau);

IR = fft(ir, Nfft); % transfer function
H_LIN = fft(h_lin, Nfft);

figure;
subplot(211);
plot(linspace(0, (length(ir)-1)*1000/fs, length(ir)), ir);
grid on; xlabel('time (ms)'); ylabel('amplitude');
title('HW4 Q1(b): Original Impulse Response');
subplot(212);
%plot([0:2*tau-1]*1000/fs, h_lin(1:2*tau));
plot([0:2*tau-1]*500/fs, h_lin(1:2*tau));
grid on; xlabel('time (ms)'); ylabel('amplitude');
title('HW4 Q1(b): Linear Phase Impulse Response');

figure;
subplot(211);
```

```
plot(20*log10(abs(IR))); axis([xlim -150 50]); grid on;
xlabel('frequency (Hz)'); ylabel('magnitude (dB)');
title('HW4 Q1(b): Magnitude of Original Phase Impulse Response');
subplot(212);
plot(20*log10(abs(H_LIN))); axis([xlim -150 50]); grid on;
xlabel('frequency (Hz)');
ylabel('magnitude (dB)'); title('HW4 Q1(b): Magnitude of Linear Phase Impulse Response

% Part (c)
% Both filt_noise and filt_noise_lin sound the same.

noise = randn(1, fs/2);
filt_noise = fftfilt(ir, noise);
filt_noise_lin = fftfilt(h_lin, noise);
% sound(filt_noise);
% sound(filt_noise_lin);

% function h_lin = linphase(ir, Nfft, tau)
% Music424 Homework #4
% Spring 2008
% SongHui Chon

% Problem 1
% Part (a): linear phase impulse response
% based on page 281 of the lecture notes

function h_lin = linphase(ir, Nfft, tau)

L = length(ir); % length of impulse response in samples

omega = [0:Nfft-1]*2*pi/Nfft; % frequency axis from [0, 2*pi)
IR = fft(ir, Nfft); % transfer function
h_lin = real(ifft(exp(-j*tau*omega).*abs(IR)')); % linear phase filter
h_lin = h_lin(1:Nfft/2); % linear phase impulse response for positive frequency range
```

1(b). [**7 Points**] Figure 1 shows the original and converted impulse responses and Figure 2 the original and converted magnitude frequency responses respectively. As can be seen from both figures, the magnitudes are pretty much the same.

1(c). [**3 Points**] The convolution outputs sound the same. This makes sense, since both original and converted impulse responses have the same magnitude frequency responses.

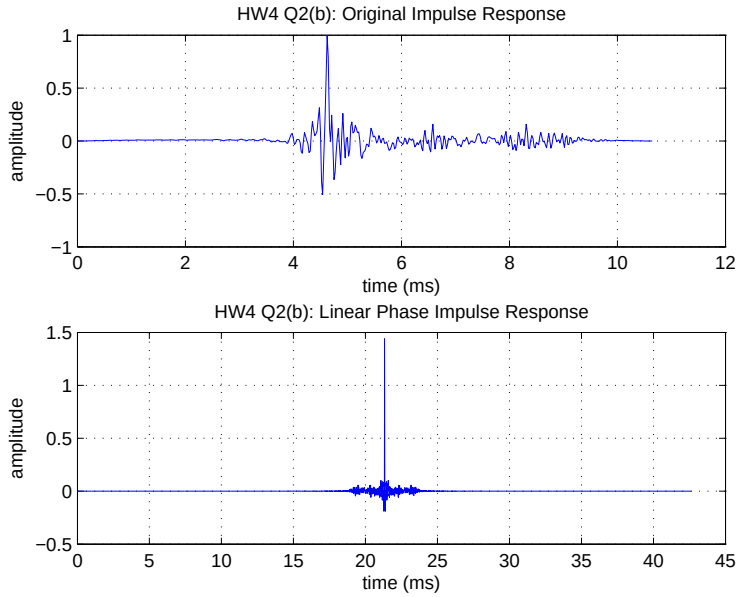

Figure 1: Problem 1(b): impulse response

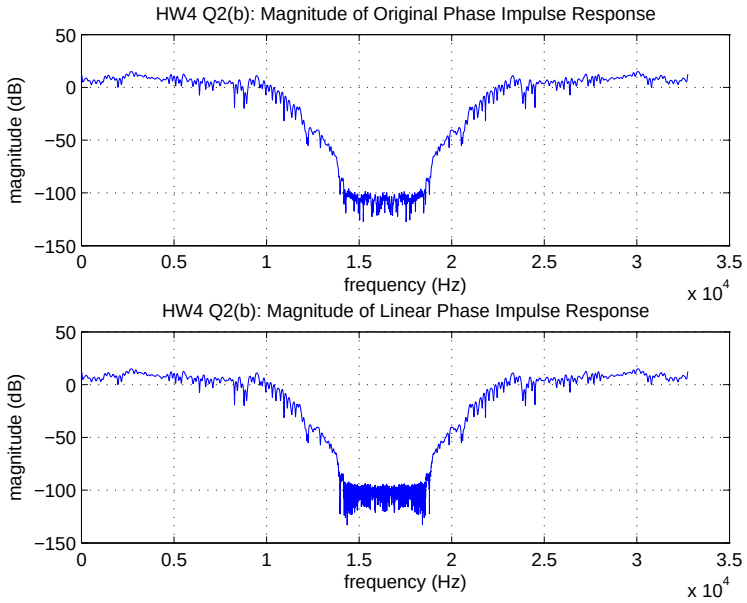

Figure 2: Problem 1(b): magnitude frequency response

Problem 2. [**30 Points**]

2(a). [**20 Points**] The matlab scripts hw4q2.m and minphase.m are shown below.

```
% hw4q2.m
% Music424 Homework #4
% Spring 2008
% SongHui Chon

% Problem 2
% minimum phase impulse response
% based on pages 286-293 of the lecture notes

% Part (b)

clear all;
load ps04.mat;

ir = ir / max(abs(ir));
Nfft = 8192; % fft size in samples

h_min = minphase(ir, Nfft);

IR = fft(ir, Nfft); % transfer function
H_MIN = fft(h_min, Nfft);

figure;
subplot(211);
plot(linspace(0, (length(ir)-1)*1000/fs, length(ir)), ir);
grid on; xlabel('time (ms)'); ylabel('amplitude');
title('HW4 Q2(b): Original Impulse Response');
subplot(212);
plot(linspace(0, (length(ir)-1)*1000/fs, length(ir)), h_min(1:length(ir)));
grid on; xlabel('time (ms)'); ylabel('amplitude');
title('HW4 Q2(b): Minimum Phase Impulse Response');

figure;
subplot(211);
plot(20*log10(abs(IR))); axis([xlim -150 50]); grid on;
xlabel('frequency (Hz)'); ylabel('magnitude (dB)');
title('HW4 Q2(b): Magnitude of Original Phase Impulse Response');
subplot(212);
plot(20*log10(abs(H_MIN))); axis([xlim -150 50]); grid on;
```

```
xlabel('frequency (Hz)'); ylabel('magnitude (dB)');
title('HW4 Q2(b): Magnitude of Minimum Phase Impulse Response');

% Part (c)
% Both filt_noise and filt_noise_min sound the same.

noise = randn(1, fs/2);
filt_noise = fftfilt(ir, noise);
filt_noise_min = fftfilt(h_min, noise);
% sound(filt_noise);
% sound(filt_noise_min);

% function h_min = minphase(ir, Nfft)
% Music424 Homework #4
% Spring 2008
% SongHui Chon

% Problem 2
% Part (a): minimum phase impulse response
% based on pages 286-293 of the lecture notes

function h_min = minphase(ir, Nfft)

L = length(ir); % length of impulse response in samples

IR = fft(ir, Nfft); % transfer function
h_min = real(ifft(conj(exp(hilbert(log(abs(IR')+eps))))); % minimum phase filter
h_min = h_min(1:Nfft/2); % minimum phase impulse response for positive frequency range
```

2(b). [**7 Points**] Figure 3 shows the original and converted impulse responses and Figure 4 the original and converted magnitude frequency responses respectively. As can be seen from both figures, the magnitudes are pretty much the same.

2(c). [**3 Points**] Again, the convolution outputs sound the same, since both original and converted impulse responses have the same magnitude frequency responses.

Problem 3. [**40 Points**]

3(a). [**25 Points**] The matlab scripts hw4q3.m and cbsmooth.m are shown below.

```
% hw4q3.m
% Music424 Homework #4
% Spring 2008
% SongHui Chon
```

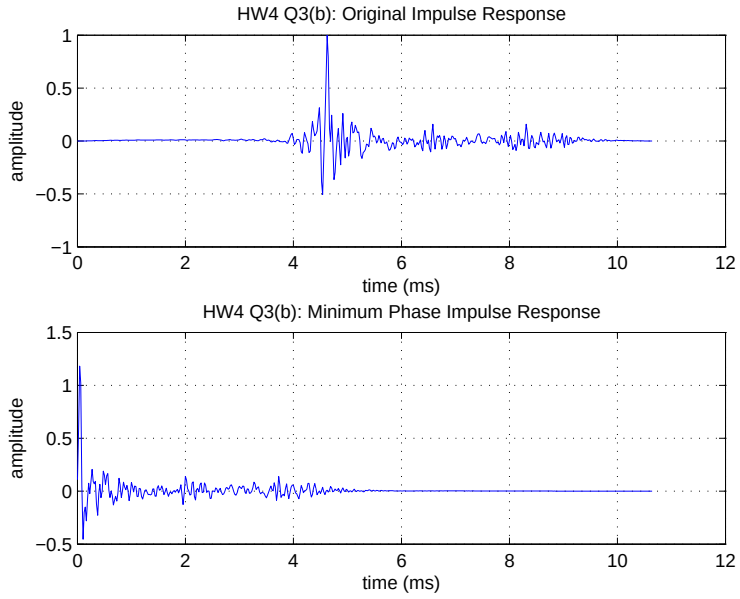

Figure 3: Problem 2(b): impulse response

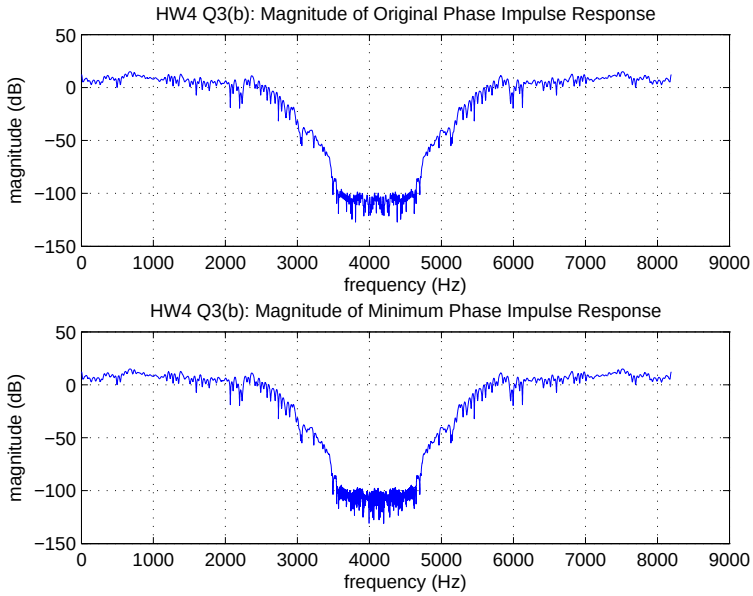

Figure 4: Problem 2(b): magnitude frequency response

```
% Problem 3
% Part (b): perform critical-band smoothing on an input transfer function
%           with beta = 0.5, 1.0 and 2.0

clear all;
load ps04.mat;

ir = ir / max(abs(ir));
Nfft = 8192;
beta = [0.5, 1, 2];
H05 = cbsmooth(ir, Nfft, fs, beta(1));
H10 = cbsmooth(ir, Nfft, fs, beta(2));
H20 = cbsmooth(ir, Nfft, fs, beta(3));
H = fft(ir, Nfft); % transfer function

figure;
subplot(221);
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H(1:Nfft/2+1))), '-.');
hold on; grid on;
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H05(1:Nfft/2+1))), 'r');
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H10(1:Nfft/2+1))), 'g');
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H20(1:Nfft/2+1))), 'm');
xlabel('time (ms)'); ylabel('frequency (kHz)');
legend('original', 'beta=0.5', 'beta=1.0', 'beta=2.0', 3);
title('HW4 Q3(b): original & beta = 0.5, 1.0, 2.0');

subplot(222);
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H(1:Nfft/2+1))), '-.');
hold on; grid on;
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H05(1:Nfft/2+1))), 'r');
xlabel('time (ms)'); ylabel('frequency (kHz)');
legend('original', 'beta=0.5', 3); title('HW4 Q4(b): original & beta = 0.5');

subplot(223);
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H(1:Nfft/2+1))), '-.');
hold on; grid on;
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H10(1:Nfft/2+1))), 'g');
xlabel('time (ms)'); ylabel('frequency (kHz)');
legend('original', 'beta=1.0', 3); title('HW4 Q4(b): original & beta = 1.0');

subplot(224);
```

```
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H(1:Nfft/2+1))), '-.');
hold on; grid on;
semilogx(((fs/2)*[0:Nfft/2]/(Nfft/2)/1000), 20*log10(abs(H20(1:Nfft/2+1))), 'm');
xlabel('time (ms)'); ylabel('frequency (kHz)');
legend('original', 'beta=2.0', 3); title('HW4 Q3(b): original & beta = 2.0');

% Part (c)

noise = randn(1, fs/2);
filt_noise = fftfilt(ir, noise);
filt_noise05 = fftfilt(real(ifft(H05(1:Nfft/2))), noise);
filt_noise10 = fftfilt(real(ifft(H10(1:Nfft/2))), noise);
filt_noise20 = fftfilt(real(ifft(H20(1:Nfft/2))), noise);

% function H_smooth = cbsmooth(ir, Nfft, fs, beta)
% Music424 Homework #4
% Spring 2007
% SongHui Chon

% Problem 3
% Part (a): perform critical-band smoothing on an input transfer function
%           and returns frequency response of smoothed transfer function
% based on pages 273-280 of lecture notes

function H_smooth = cbsmooth(ir, Nfft, fs, beta)

L = length(ir); % length of impulse response in samples
H = fft(ir, Nfft); % transfer function
H_sq = abs(H(1:Nfft/2+1)) .^ 2;

P = zeros(size(H_sq));
for i=1:Nfft/2+1 % Nfft/2+1 -number of frequency lines from 0 to fs/2
    f_center = i * (fs/2) / (Nfft/2) / 1000; % center freq in kHz
    cb = khz2erb(f_center); % critical band in erb
    f_low = erb2khz(cb - beta/2); % low frequency limit
    f_high = erb2khz(cb + beta/2); % high frequency limit

    index_low = max(1, round(f_low * 1000 * (Nfft/2) / (fs/2) ));
    index_high = min(Nfft/2+1, round(f_high * 1000 * (Nfft/2) / (fs/2)));

    P(i) = mean(H_sq(index_low:index_high));
end
```

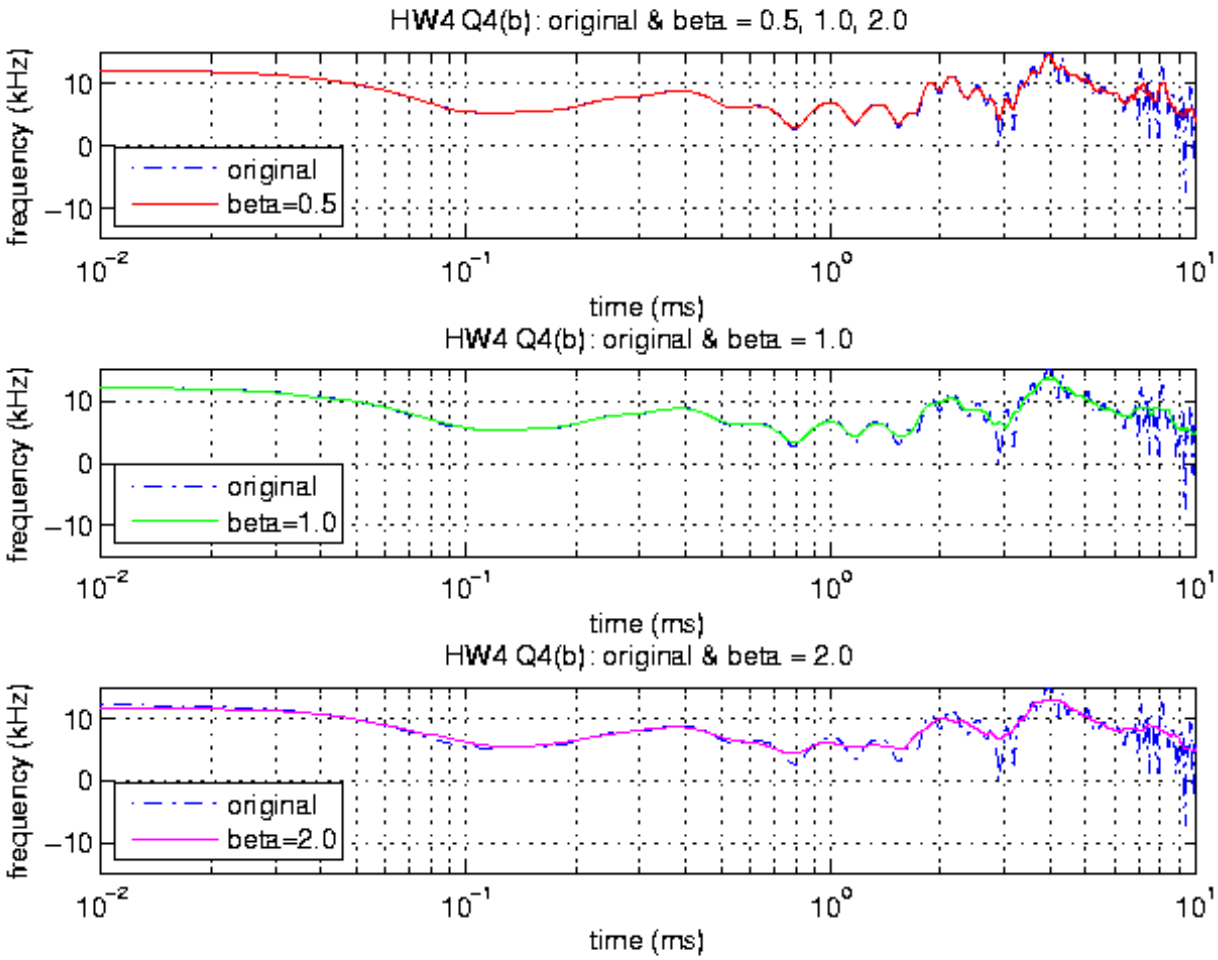

Figure 5: Problem 3(b): critical band smoothed transfer functions

```
H_smooth = sqrt(P);
```

3(b). [**7 Points**] Figure 5 shows smoothed transfer functions in comparison with the original with $\beta = 0.5, 1, 2$.

3(c). [**8 Points**] The filtered white noises for $\beta = 0.5, 1, 2$ all seem to sound the same. However, when $\beta = 10$, the filtered noise sounded a bit different.