EE6133: Assignment 2
2 Channel DFT Filter Banks
November 28, 2021

Contents

1 Block Diagram of the FB

Given below is the block diagram of both the analysis and the synthesis side of this experiment.

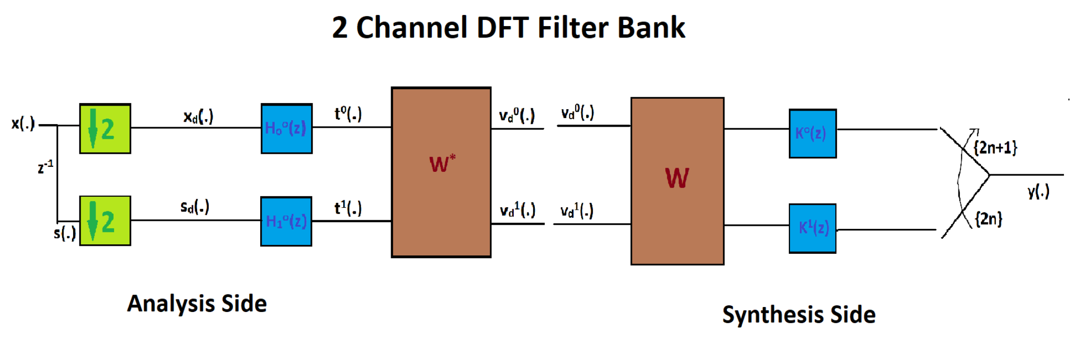

Figure 1: Block Diagram of the 2 channel DFT Analysis and Synthesis FBs

The missing links in the question are filled above in the given block diagram.

2 Choice of Prototype

Firstly, since we want a very good reproduction of the input audio, we go for a very high order Type-2 filter as it'll then have a very smooth and flat magnitude response in its passband.
I chose an order of 63 for my prototype filter H_0(z) and the filter coefficients are given below.

```
  Columns 1 through 10

    0.0005   -0.0014    0.0002    0.0014   -0.0005   -0.0021    0.0008    0.0030   -0.0014   -0.0041

  Columns 11 through 20

    0.0021    0.0055   -0.0031   -0.0072    0.0045    0.0093   -0.0063   -0.0119    0.0088    0.0152

  Columns 21 through 30

   -0.0120   -0.0195    0.0167    0.0255   -0.0236   -0.0346    0.0352    0.0510   -0.0594   -0.0921

  Columns 31 through 40

    0.1461    0.4532    0.4532    0.1461   -0.0921   -0.0594    0.0510    0.0352   -0.0346   -0.0236

  Columns 41 through 50

    0.0255    0.0167   -0.0195   -0.0120    0.0152    0.0088   -0.0119   -0.0063    0.0093    0.0045

  Columns 51 through 60

   -0.0072   -0.0031    0.0055    0.0021   -0.0041   -0.0014    0.0030    0.0008   -0.0021   -0.0005

  Columns 61 through 64

    0.0014    0.0002   -0.0014    0.0005
```

The columns start from 1 since MATLAB indexes array from 1 instead of 0.
The support of this filter is [0,63] which MATLAB has indexed as [1,64].
As we can see above, the filter is symmetric about it's 32nd or the 33rd index as it is linear phase.
Shown below is the DTFT of this prototype filter.

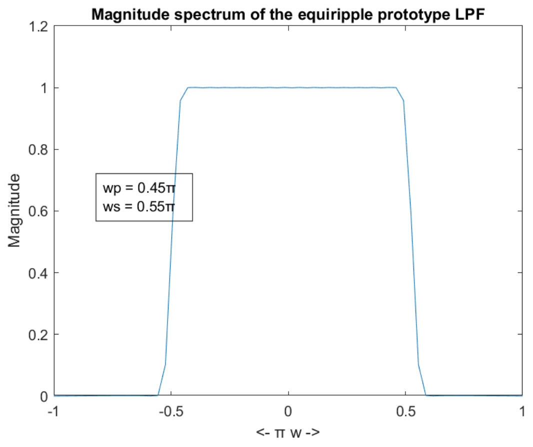

Figure 2: Prototype LPF

Shown below is the time domain plot of this prototype.

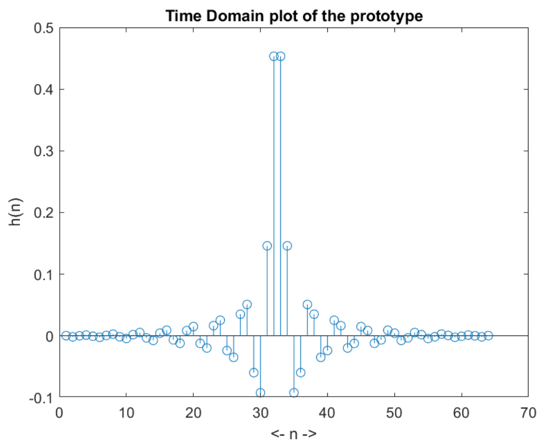

Figure 3: Prototype LPF time domain

2.1 The polyphase components

The 2-polyphase components are defined as $h_k(n) = h(2n + k)$.
Therefore, the zeroeth polyphase component of the prototype will be $h_0^0(n) = h^0(2n)$ and the first polyphase component will be $h_1^0(n) = h^0(2n + 1)$.
Thus, $h_0^0(n)$ will comprise of the even indices of the protype filter (Odd as per MATLAB indexing) and $h_1^0(n)$ comprises of the odd indices of the prototype (Even as per MATLAB indexing).
Given below are the determined coefficients for the two polyphase components.

```
 Zeroeth polyphase component:

 Columns 1 through 10

   0.0005    0.0002   -0.0005    0.0008   -0.0014    0.0021   -0.0031    0.0045   -0.0063    0.0088

 Columns 11 through 20

  -0.0120    0.0167   -0.0236    0.0352   -0.0594    0.1461    0.4532   -0.0921    0.0510   -0.0346

 Columns 21 through 30

   0.0255   -0.0195    0.0152   -0.0119    0.0093   -0.0072    0.0055   -0.0041    0.0030   -0.0021

 Columns 31 through 32

   0.0014   -0.0014

First polyphase component:

 Columns 1 through 10

  -0.0014    0.0014   -0.0021    0.0030   -0.0041    0.0055   -0.0072    0.0093   -0.0119    0.0152

 Columns 11 through 20

  -0.0195    0.0255   -0.0346    0.0510   -0.0921    0.4532    0.1461   -0.0594    0.0352   -0.0236
 Columns 21 through 30

   0.0167   -0.0120    0.0088   -0.0063    0.0045   -0.0031    0.0021   -0.0014    0.0008   -0.0005

 Columns 31 through 32

   0.0002    0.0005
```

As discussed above, if we intersperse the two polyphase components we end up with the above prototype filter i.e. ,$H^0(z) = H^0_0(z^2) + z^{-1}H^0_1(z^2)$
This is demonstrated using a graph.

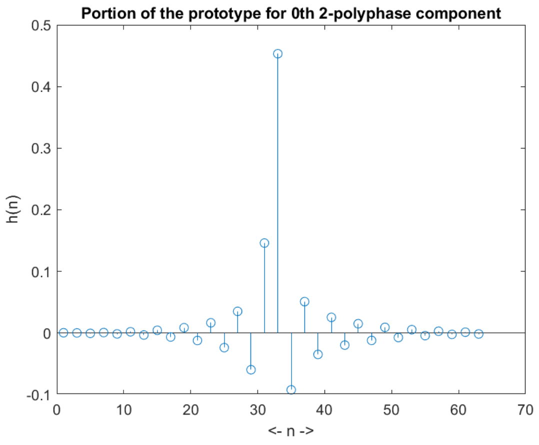

Figure 4: Upsampled zeroeth 2-polyphase component time domain

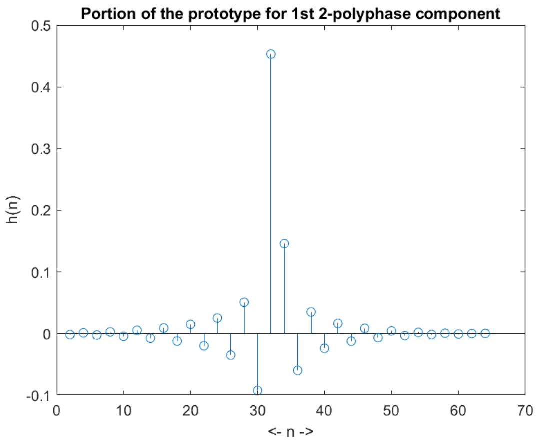

Figure 5: Upsampled and delayed first 2-polyphase component time domain

2.2 Filters Derived from the prototype

For no aliasing, we'd prefer $G^0(z) = H^1(-z)$ and $G^1(z) = -H^0(-z)$.
As this is a DFT FB, we have $H^1(z) = H^0(-z)$.
The DTFTs of this configuration is shown below.

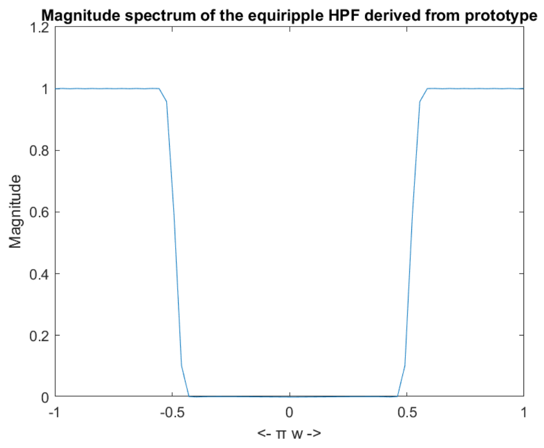

Figure 6: $\underline{\overline{H^1}}(w)$

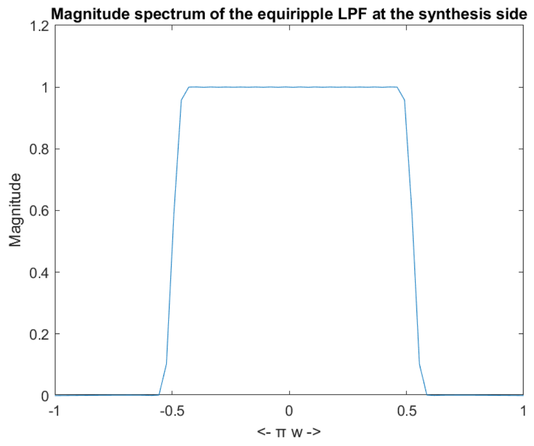

Figure 7: $\underline{\overline{G^0}}(w)$

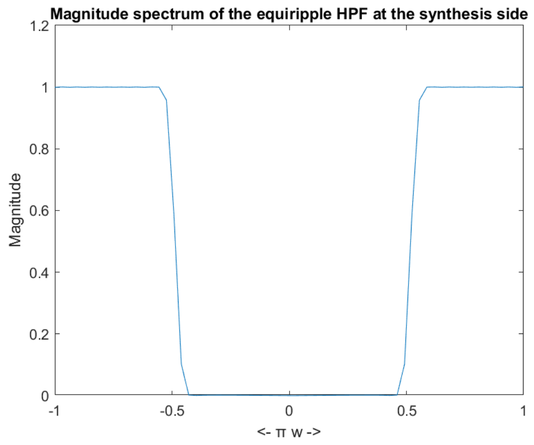

Figure 8: $\underline{\overline{G^1}}(w)$

3 Plots for the Input Audio

3.0.1 Speech Signal

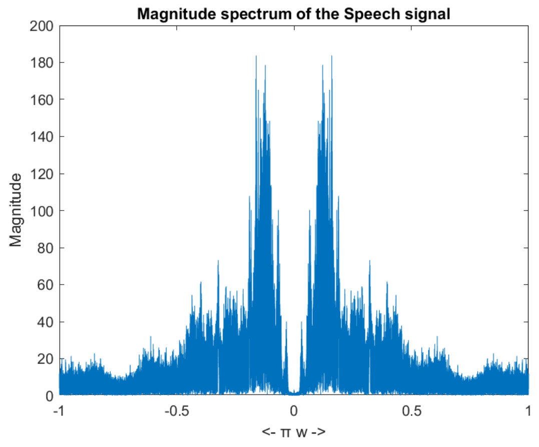

Figure 9: Input Speech Signal

3.0.2 Music Signal

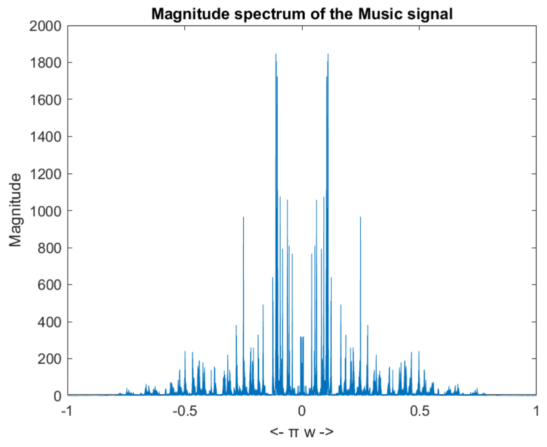

Figure 10: Input Music Signal

4 Output for Question 1

Our configuration for this question is as the above case which prevents aliasing.
$K^0(z) = H^1(z)$
$K^1(z) = H^0(z)$

4.1 Plot for Tzp

As derived in class we know that $T_{zp}(w) = \frac{1}{2}\left(\left|H^0(w)\right|^2 + \left|H^1(w)\right|^2\right)$.
From the plots for the analysis filters, we'd expect $T_{zp}(w)$ to be consistently about 0.5 .

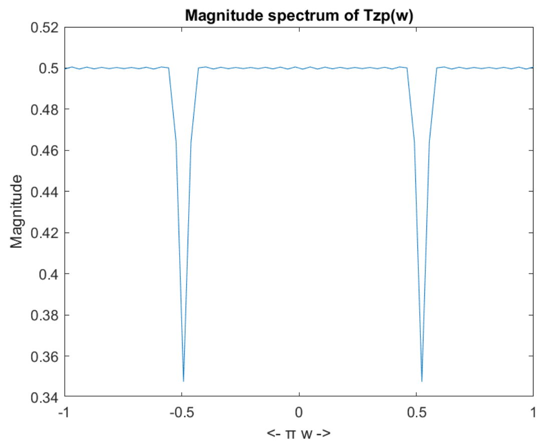

Figure 11: $\overline{\underline{T_{zp}}}(w)$

4.2 Plot for the output audio

4.2.1 Speech Signal

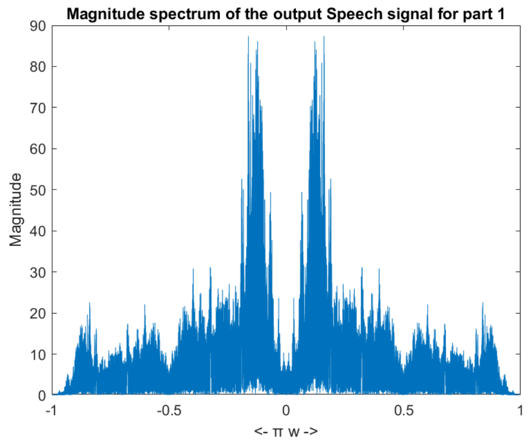

Figure 12: Output Speech Signal

4.2.2 Music Signal

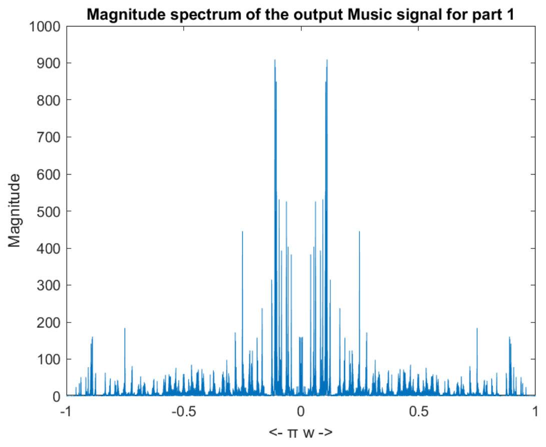

Figure 13: Output Music Signal

5 Output for Question 2

Our configuration for this question does not eliminate aliasing.
$K^0(z) = H^0(z)$
$K^1(z) = H^1(z)$

5.1 Plot for the output audio

5.1.1 Speech Signal

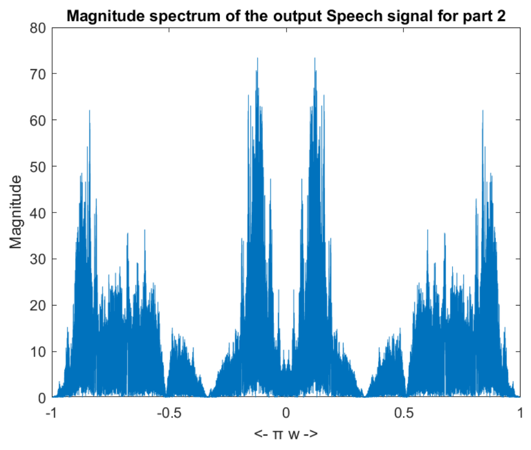

Figure 14: Output Speech Signal

5.1.2 Music Signal

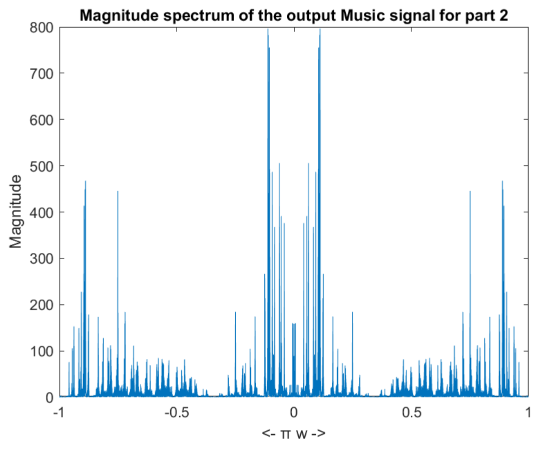

Figure 15: Output Music Signal

6 Observations

- Firstly, as we'd expect, we see that $T_{zp}(w)$ has a near constant magnitude response of 0.5 which signifies an almost perfect reconstruction with some decrease in the volume.
- From the speech and music plots for question 1, we see that the spectrum plots are almost the same except scaled by 0.5 and some frequency spilling about the boundaries.
- From the speech and music plots for the second question, we see that the scaling is 0.5 as above, however there are some aliasing components which result in peaks at the edges of the spectrum as we'd expect.

- On comparing the audios, we see that both the audios are very well perceptible but the audio from the second question has comparatively more artefacts and a shrill high frequency sound in the background which indicates aliasing.
- We see that the difference is more between the audios for the speech signal when compared to the music signal. This is however expected as the music spectrum has very little frequency content at its edges when compared to the speech spectrum and thus the aliasing affects the perception of the speech signal more.

7 Matlab Code

The MATLAB code written for the experiment is shown below,

```
%Multirate Digital Signal Processing Programming Assignment - 2

%Done By - Abhishek Sekar (EE18B067)

clear; %clearing everything out
clc;

%Getting the audio
path = fullfile('Audio_Files','speech8khz.wav') %path where the audio files are hosted

[x fs]=audioread(path); %speech data or music data depending on file

%Original magnitude spectrum
n_dtft=2^(ceil(log2(length(x)))); %number of point in DTFT
X = fftshift(fft(x,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
figure();
spectrum = plot(f_dtft,abs(X)); %plotting magnitude spectrum
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the Speech signal');
x1 = [0.8 0.6];
y1 = [0.5 0.8];
saveas(spectrum,'speech_spectrum.png','png');

%Designing the prototype filter H_o(z)

rp = 0.01;          % Passband ripple in dB
rs = 50;            % Stopband ripple in dB
w_p = 0.45*fs/2;      %passband frequency
w_s = 0.55*fs/2;      %stopband frequency
f = [w_p,w_s]; % freq for firpm
```

```
a = [1 0];        % Desired amplitudes

dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[n,fo,ao,w] = firpmord(f,a,dev,fs); %firpmord produces n = 62. As we want type-2, we go with or
h_o = firpm(63,fo,ao,w); %chosen 63 as it gives a very good filter


%plot for the filter magnitude
n_dtft=2^(ceil(log2(length(h_o)))); %number of point in DTFT
H_o = fftshift(fft(h_o,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
figure();
prototype = plot(f_dtft,abs(H_o));  %filter magnitude response
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the equiripple prototype LPF');
dim = [.2 .3 .3 .3];
str = {'wp = 0.45 pi', 'ws = 0.55 pi'};
annotation('textbox',dim,'String',str,'FitBoxToText','on');
saveas(prototype,'prototype_Ho.png','png');

%performing 2-polyphase decomposition

h_o_0 = h_o(1:2:end); % h_o_0(n) = h_o(2n)
h_o_1 = h_o(2:2:end); % h_o_1(n) = h_o(2n+1)

%plot for the polyphase components
figure();
h_o_stem = stem(h_o);
xlabel('<- n ->');
ylabel('h(n)');
title('Time Domain plot of the prototype');
saveas(h_o_stem,'prototype_stem.png','png');

indices = linspace(1,length(h_o),length(h_o));

%plotting the 0th 2-polyphase component
figure();
h_o_0_stem = stem(indices(1:2:end),h_o_0);
xlabel('<- n ->');
ylabel('h(n)');
```

```
title('Portion of the prototype for 0th 2-polyphase component');
saveas(h_o_0_stem,'polyphase_0_stem.png','png');

%plotting the 1st 2-polyphase component
figure();
h_o_1_stem = stem(indices(2:2:end),h_o_1);
xlabel('<- n ->');
ylabel('h(n)');
title('Portion of the prototype for 1st 2-polyphase component');
saveas(h_o_1_stem,'polyphase_1_stem.png','png');

%Filterbank processing begins !!

%Analysis Filterbank operations

%creating a delay element
z_inv = dfilt.delay;

s = filter(z_inv,x); %x delayed by one unit
s = s(2:end);        %s now starts with index 0
s = [s;x(length(x))]; %as filter cuts off the last element

%downsampling both branches

x_d = downsample(x,2); %downsampling by 2
s_d = downsample(s,2); %downsampling by 2

%output through the analysis filters

t_0 = filter(h_o_0,1,x_d); %first branch
t_1 = filter(h_o_1,1,s_d); %second branch

%output through the inverse DFT matrix

v_d_0 = t_0 + t_1; %first branch
v_d_1 = t_0 - t_1; %second branch


% Synthesis Filterbank operations

%output through the DFT matrix
```

```
v_d_0_out = v_d_0 + v_d_1;
v_d_1_out = v_d_0 - v_d_1;

%choosing the synthesis filters

%For the first part
k0 = h_o_1;
k1 = h_o_0;

%For the second part
k_0 = h_o_0;
k_1 = h_o_1;

%output through the synthesis filters

%for the first part
y1 = filter(k0,1,v_d_0_out);
y0 = filter(k1,1,v_d_1_out);

%for the second part
y_1 = filter(k_0,1,v_d_0_out);
y_0 = filter(k_1,1,v_d_1_out);

%implementing the final commutator

y_q1 = zeros(size(x));
y_q2 = zeros(size(x));

y_q1(1:2:end) = y0; %when the commutator is at the bottom {2n}
y_q1(2:2:end) = y1; %when the commutator is at the top    {2n+1}


y_q2(1:2:end) = y_0; %when the commutator is at the bottom {2n}
y_q2(2:2:end) = y_1; %when the commutator is at the top    {2n+1}


%generating the output audio file
audiowrite('speech_q1.wav',y_q1,fs)
audiowrite('speech_q2.wav',y_q2,fs)
```

```
%plotting Tzp(w) for the first question

%first computing the DTFTs of the two analysis filters

%filter H_0

n_dtft=2^(ceil(log2(length(h_o)))); %number of point in DTFT
H_0 = fftshift(fft(h_o,n_dtft));



%filter H_1
h_i = h_o;
h_i(2:2:end) = -h_i(2:2:end); % h_1 = (-1)^n*h_0 pi shifted
n_dtft=2^(ceil(log2(length(h_i)))); %number of point in DTFT
H_1 = fftshift(fft(h_i,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
prototype_1 = plot(f_dtft,abs(H_1));  %filter magnitude response
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the equiripple HPF derived from prototype');
saveas(prototype_1,'prototype_H1.png','png');

%filter G_0
g_o = h_o;
n_dtft=2^(ceil(log2(length(g_o)))); %number of point in DTFT
G_0 = fftshift(fft(g_o,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
prototype_2 = plot(f_dtft,abs(G_0));  %filter magnitude response
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the equiripple LPF at the synthesis side');
saveas(prototype_2,'prototype_G0.png','png');



%filter G_1
g_i = -h_i;
n_dtft=2^(ceil(log2(length(g_i)))); %number of point in DTFT
G_1 = fftshift(fft(g_i,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
prototype_3 = plot(f_dtft,abs(G_1));  %filter magnitude response
xlabel('<- pi w ->');
```

```
ylabel('Magnitude');
title('Magnitude spectrum of the equiripple HPF at the synthesis side');
saveas(prototype_3,'prototype_G1.png','png');



%resulting Tzp(w)

T_zp = 0.5*(abs(H_0 .^2) + abs(H_1 .^2));
figure();
Tzp = plot(f_dtft,abs(T_zp));  %filter magnitude response
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of Tzp(w)');
saveas(Tzp,'Tzp.png','png');

%plots of output audio spectrum
n_dtft=2^(ceil(log2(length(y_q1)))); %number of point in DTFT
Y_q1 = fftshift(fft(y_q1,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
figure();
spectrum_q1 = plot(f_dtft,abs(Y_q1)); %plotting magnitude spectrum
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the output Speech signal for part 1');
x1 = [0.8 0.6];
y1 = [0.5 0.8];
saveas(spectrum_q1,'speech_spectrum_q1.png','png');

n_dtft=2^(ceil(log2(length(y_q2)))); %number of point in DTFT
Y_q2 = fftshift(fft(y_q2,n_dtft));
f_dtft = linspace(-1,1,n_dtft); % freq
figure();
spectrum_q2 = plot(f_dtft,abs(Y_q2)); %plotting magnitude spectrum
xlabel('<- pi w ->');
ylabel('Magnitude');
title('Magnitude spectrum of the output Speech signal for part 2');
x1 = [0.8 0.6];
y1 = [0.5 0.8];
saveas(spectrum_q2,'speech_spectrum_q2.png','png');
```

8 Drive Link for Audio Files

This is the link to the output audio files of the experiment.