1. Solve:

The polynomial *P(z)* can be written as

$$P(z)=(1+z^{-1})^4(1+z)^4(az^3+bz^2+cz+d+cz^{-1}+bz^{-2}+az^{-3}).$$

Due to

$$(1+z^{-1})^4=1+4z^{-1}+6z^{-2}+4z^{-3}+z^{-4}$$

$$(1+z)^4=1+4z+6z^2+4z^3+z^4,$$

I can write the following Matlab code to know the form of *P(z)*

```
Z=[1 4 6 4 1];
Z=conv(Z,Z);
syms z a b c d;   % Know the form of P(z)
f0=sym('(z^(-4)+8*z^(-3)+28*z^(-2)+56*z^(-
1)+70+56*z^1+28*z^2+8*z^3+z^4)*(a*z^(-3)+b*z^(-2)+c*z^(-
1)+d+c*z^1+b*z^2+a*z^3)');
collect(f0,z)
```

The answer is shown as follows (It's too long to shown it all):

```
ans =

(a*z^14 + (8*a + b)*z^13 + (28*a + 8*b + c)*z^12 + (56*a + 28*b + 8*c + d)*z^11 + (70*a + 56*b
```

So I can know

$$\begin{aligned}P(z)=&az^7+(8a+b)z^6+(28a+8b+c)z^5+(56a+28b+8c+d)z^4\\&+(70a+56b+29c+8d)z^3+(56a+71b+64c+28d)z^2\\&+(29a+64b+98c+56d)z+(16a+56b+112c+70d)\\&+(29a+64b+98c+56d)z^{-1}+(56a+71b+64c+28d)z^{-2}\\&+(70a+56b+29c+8d)z^{-3}+(56a+28b+8c+d)z^{-4}\\&+(28a+8b+c)z^{-5}+(8a+b)z^{-6}+az^{-7}\end{aligned}$$

Since the even coefficients of *P(z)* must equal zero, and p_0=1, I can write

$$\begin{cases}8a+b=0\\56a+28b+8c+d=0\\56a+71b+64c+28d=0\\16a+56b+112c+70d=0\end{cases}$$

I can write the following Matlab code to get coefficients a, b, c, and d

```
syms a b c d;     % Get the coefficients a,b,c,and d
f1=sym('8*a+b=0');
f2=sym('56*a+28*b+8*c+d=0');
f3=sym('56*a+71*b+64*c+28*d=0');
f4=sym('16*a+56*b+112*c+70*d=1');
[a,b,c,d]=solve(f1,f2,f3,f4);
```

```
a=double(a);
b=double(b);
c=double(c);
d=double(d);
```

The result is

```
a =
-5/2048

a =
   -0.0024

b =
5/256

b =
    0.0195

c =
-131/2048

c =
   -0.0640

d =
13/128

d =
    0.1016
```

In order to receive zeros of *P(z)*, I write the following code

```
P=[a 8*a+b 28*a+8*b+c 56*a+28*b+8*c+d 70*a+56*b+29*c+8*d
56*a+71*b+64*c+28*d 29*a+64*b+98*c+56*d 16*a+56*b+112*c+70*d
29*a+64*b+98*c+56*d 56*a+71*b+64*c+28*d 70*a+56*b+29*c+8*d
56*a+28*b+8*c+d 28*a+8*b+c 8*a+b a];
zeros=roots(P);
zeros=zeros';
```

The result is

```
zeros =

   3.0407 + 0.0000i
   2.0311 + 1.7390i
   2.0311 - 1.7390i
  -1.0097 + 0.0041i
  -1.0097 - 0.0041i
  -1.0038 + 0.0097i
  -1.0038 - 0.0097i
  -0.9959 + 0.0094i
  -0.9959 - 0.0094i
  -0.9906 + 0.0038i
  -0.9906 - 0.0038i
   0.2841 + 0.2432i
   0.2841 - 0.2432i
   0.3289 + 0.0000i
```

I can notice several things

(i) The polynomial has eight zeros on the unit circle approximately.

(ii) The first three zeros are the reciprocals of the last three zeros.

After knowing zeros, with the purpose of making each filter linear, I can write the following Matlab code to get coefficients of $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$

```
% Get the coefficients of H0(z) before dividing a factor
t1=[1 -Zeros(1,1)];
t2=[1 -Zeros(1,2)];
t3=[1 -Zeros(1,3)];t4=[1 -Zeros(1,4)];
t5=[1 -Zeros(1,5)];t6=[1 -Zeros(1,6)];
t7=[1 -Zeros(1,7)];
h0_t1=conv(t1,t2);
h0_t2=conv(t3,t4);
h0_t3=conv(t5,t6);
h0_b=conv(h0_t1,h0_t2);
h0_b=conv(h0_b,h0_t3);
h0_b=conv(h0_b,t7);
% Get the coefficients of G0(z) before dividing a factor
t8=[1 -Zeros(1,8)];t9=[1 -Zeros(1,9)];
t10=[1 -Zeros(1,10)];t11=[1 -Zeros(1,11)];
t12=[1 -Zeros(1,12)];t13=[1 -Zeros(1,13)];
t14=[1 -Zeros(1,14)];
g0_t1=conv(t8,t9);
g0_t2=conv(t10,t11);
g0_t3=conv(t12,t13);
g0_b=conv(g0_t1,g0_t2);
g0_b=conv(g0_b,g0_t3);
g0_b=conv(g0_b,t14);
h1_b=zeros(1,8);       % To get the coefficients of H1(z)
                        before dividing a factor
g1_b=zeros(1,8);       % To get the coefficients of G1(z)
                        before dividing a factor
for i=1:1:8
    if mod(i,2)==0
        h1_b(i)=-g0_b(i);
        g1_b(i)=h0_b(i);
    else
        h1_b(i)=g0_b(i);
        g1_b(i)=-h0_b(i);
    end
end
% Get the coefficients of H0(z),H1(z),G0(z),G1(z) after
dividing factors
h0_b=h0_b/256;h1_b=h1_b/8;
```

```
g0_b=g0_b/8;g1_b=g1_b/256;
```

The Matlab code to show the magnitude response of the linear phase filters $H_0(z)$ and $H_1(z)$ is

```
w=pi*(0:0.005:1);
h0=freqz(h0_b,1,w);
h1=freqz(h1_b,1,w);
hmag0=abs(h0);
hmag1=abs(h1);
figure(1)
plot(w/pi,hmag0,'r-');hold on;
plot(w/pi,hmag1,'g--');hold on;
title('Magnitude Spectrum Plot')
xlabel('Normalized Frequency (radians)')
ylabel('Magnitude')
```

The figure of the magnitude of the two filters is shown in Fig. 1. It's apparent that these are not mirrored filters.

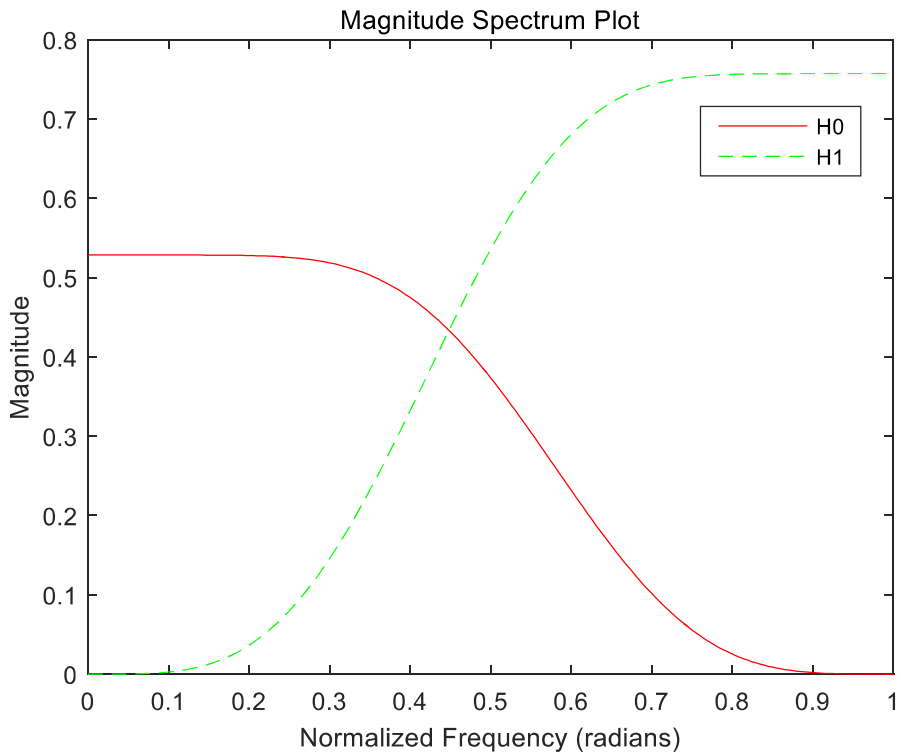


Fig. 1. The magnitude response of two resulting filters

Finally, because of $H_1(z) = G_0(z)$ and $G_1(z) = -H_0(-z)$, it's easy for us to know

$$A(z) = \{G_0(z)H_0(-z) + G_1(z)H_1(-z)\}/2 = 0 .$$

Moreover, due to

$$T(z) = \{H_0(z)H_1(-z) - H_0(-z)H_1(z)\}/2 ,$$

I can write the following Matlab code to get $T(z)$

```
M_T=conv(g0_b,h0_b);
N_T=conv(g1_b,h1_b);
T_z=(M_T+N_T)/2;
```

The result is

```
T_z =

  1 至 4 列

   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i

  5 至 8 列

  -0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.2000 + 0.0000i

  9 至 12 列

  -0.0000 - 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  13 至 15 列

   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i
```

Therefore, these filters are perfect reconstruction filters by ensuring $A(z) = 0$ and $T(z)$ is constant magnitude with arbitrary delay.

2. Solve:

From the problem1, I can know the zeros of $P(z)$ result is

```
zeros =

   3.0407 + 0.0000i
   2.0311 + 1.7390i
   2.0311 - 1.7390i
  -1.0097 + 0.0041i
  -1.0097 - 0.0041i
  -1.0038 + 0.0097i
  -1.0038 - 0.0097i
  -0.9959 + 0.0094i
  -0.9959 - 0.0094i
  -0.9906 + 0.0038i
  -0.9906 - 0.0038i
   0.2841 + 0.2432i
   0.2841 - 0.2432i
   0.3289 + 0.0000i
```

After knowing zeros, with the purpose of making filters mirrored, I can write the following Matlab code to get coefficients of $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$

```
% Get the coefficients of H0(z)
t21=[1 -Zeros(1,8)];
t22=[1 -Zeros(1,9)];
```

```
t23=[1 -Zeros(1,10)];t24=[1 -Zeros(1,11)];
t25=[1 -Zeros(1,12)];t26=[1 -Zeros(1,13)];
t27=[1 -Zeros(1,14)];
h0_t21=conv(t21,t22);
h0_t22=conv(t23,t24);
h0_t23=conv(t25,t26);
h0_b2=conv(h0_t21,h0_t22);
h0_b2=conv(h0_b2,h0_t23);
h0_b2=conv(h0_b2,t27);
h0_b2=h0_b2*sqrt(2)/6;  % To ensure a gain of sqrt(2)
% Get the coefficients of G0(z)
g0_b2=zeros(1,8);
for i=1:1:8
  g0_b2(i)=h0_b2(9-i);
end
h1_b2=zeros(1,8);       % To get the coefficients of H1(z)
g1_b2=zeros(1,8);       % To get the coefficients of G1(z)
for i=1:1:8
    if mod(i,2)==0
        h1_b2(i)=-g0_b2(i);
        g1_b2(i)=h0_b2(i);
    else
        h1_b2(i)=g0_b2(i);
        g1_b2(i)=-h0_b2(i);
    end
end
```

The Matlab code to show the magnitude response of the linear phase filters $H_0(z)$ and $H_1(z)$ is

```
w=pi*(0:0.005:1);
h20=freqz(h0_b2,1,w);
h21=freqz(h1_b2,1,w);
hmag20=abs(h20);
hmag21=abs(h21);
figure(2)
plot(w/pi,hmag20,'r-');hold on;
plot(w/pi,hmag21,'g--');hold on;
title('Magnitude Spectrum Plot')
xlabel('Normalized Frequency (radians)')
ylabel('Magnitude')
```

The figure of the magnitude of the two filters is shown in Fig. 2. It's apparent that these are mirrored filters with a gain $\sqrt{2}$ approximately.

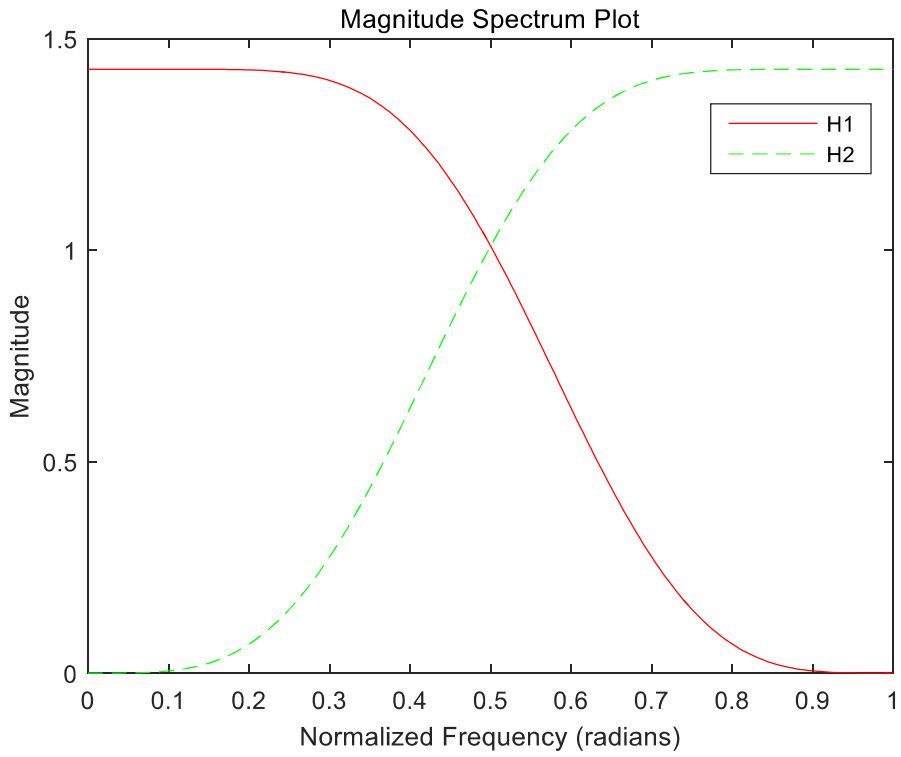


Fig. 2. The magnitude response of two resulting filters

Finally, because of $H_1(z) = G_0(z)$ and $G_1(z) = -H_0(-z)$, it's easy for us to know

$$A(z) = \{G_0(z)H_0(-z) + G_1(z)H_1(-z)\}/2 = 0 .$$

Moreover, due to

$$T(z) = \{H_0(z)H_1(-z) - H_0(-z)H_1(z)\}/2 ,$$

I can write the following Matlab code to get $T(z)$

```
M_T2=conv(g0_b2,h0_b2);
N_T2=conv(g1_b2,h1_b2);
T_z2=(M_T2+N_T2)/2;
```

The result is

```
T_z2 =

  1 至 9 列

         0   -0.0000   -0.0000    0.0000    0.0000   -0.0000    0.0000    1.0189    0.0000

  10 至 15 列

   -0.0000    0.0000    0.0000   -0.0000   -0.0000         0
```

Therefore, these filters are perfect reconstruction filters by ensuring $A(z) = 0$ and $T(z)$ is constant magnitude with arbitrary delay.

3. Solve:

(a)

The Matlab code to get the required coefficients and show the magnitude and phase response of the filters $H_0(z)$ and $H_1(z)$ is

```
ord=41-1;
wc=1/4;
b0= fir1(ord,wc,'low');
b1= fir1(ord,wc,'high');
w=pi*(0:0.005:1.0);
% Compute and plot the spectrum of the lowpass filter
h0=freqz(b0,1,w);
hmag0=abs(h0);
hphase0=angle(h0);
tol=0.95*pi;
figure(1)
subplot(2,1,1)      % Plot the magnitude response of the
                      resulting lowpass filter
hmag0=unwrap(hmag0,tol);
plot(w/pi,hmag0)
ylim([0 1.5])
title('Magnitude Response Plot (lowpass filter)')
ylabel('Magnitude ')
xlabel('Normalized Frequency (\times\pi rad/sample)')
subplot(2,1,2)      % Plot the phase response of the resulting
                      lowpass filter
hphase0=unwrap(hphase0,tol);
plot(w/pi,hphase0)
title('Phase Response Plot (lowpass filter)')
ylabel('Phase (radians)')
xlabel('Normalized Frequency (\times\pi rad/sample)')
% Compute and plot the spectrum of the highpass filter
h1=freqz(b1,1,w);
hmag1=abs(h1);
hphase1=angle(h1);
figure(2)
subplot(2,1,1)      % Plot the magnitude response of the
                      resulting highpass filter
hmag1=unwrap(hmag1,tol);
plot(w/pi,hmag1)
title('Magnitude Response Plot (highpass filter)')
ylabel('Magnitude ')
xlabel('Normalized Frequency (\times\pi rad/sample)')
subplot(2,1,2)      % Plot the phase response of the resulting
                      highpass filter
```

```
hphase1=unwrap(hphase1,tol);
plot(w/pi,hphase1)
title('Phase Response Plot (highpass filter)')
ylabel('Phase (radians)')
xlabel('Normalized Frequency (\times\pi rad/sample)')
```

The figures of the frequency response of the two filters are shown in Fig. 3 and Fig. 4 respectively.

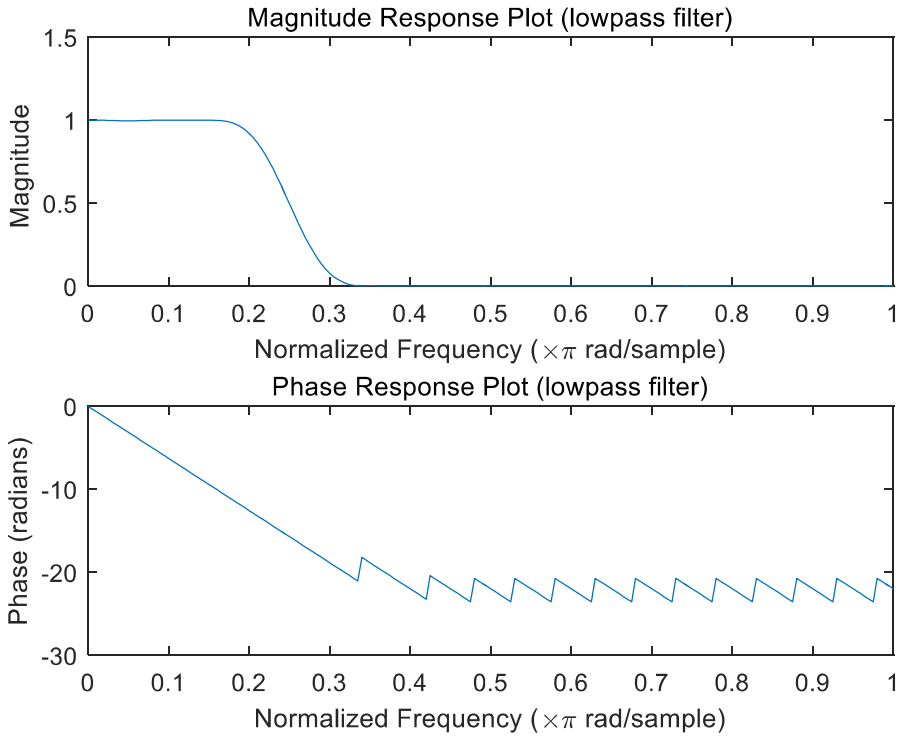


Fig. 3. The frequency response of the lowpass filter $H_0(z)$

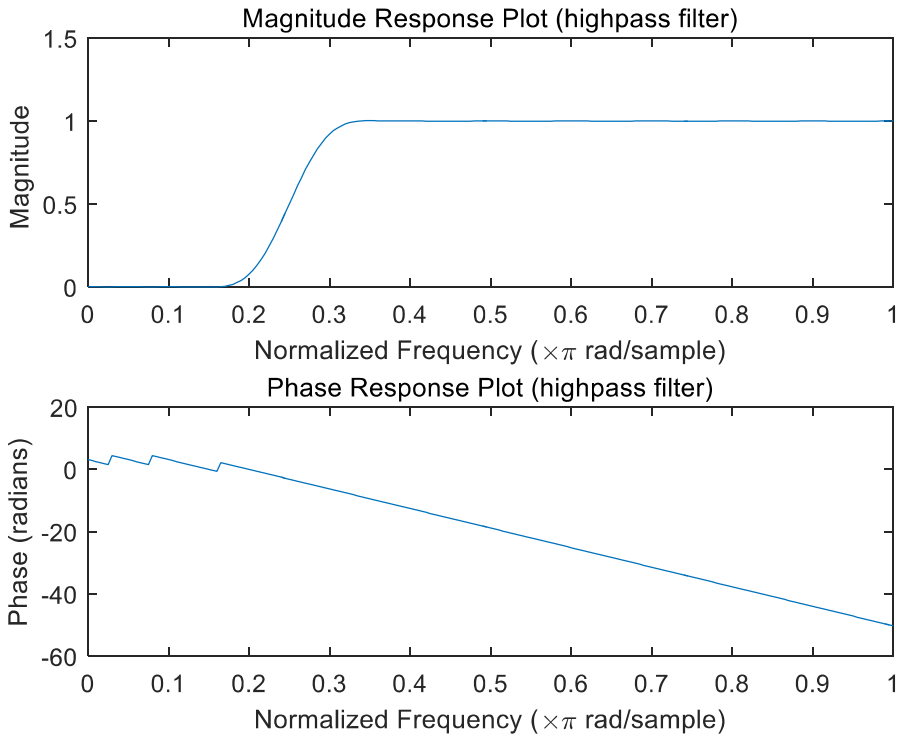
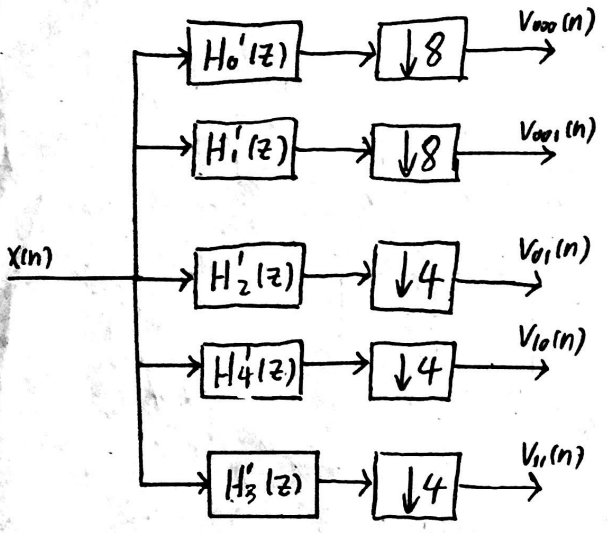
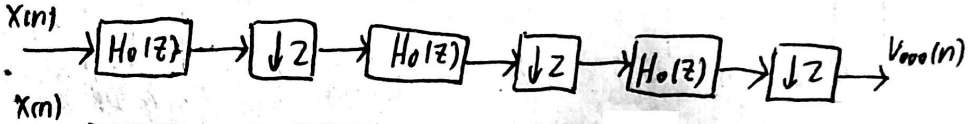


Fig. 4. The frequency response of the highpass filter $H_1(z)$

(b) A block diagram equivalent representation of the analysis section can be redrawn as



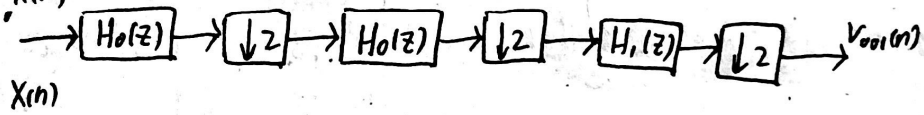
The 1st branch: X(n)



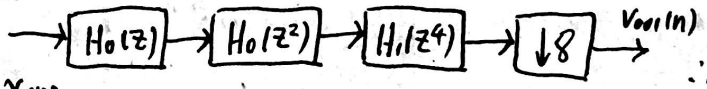
X(n)

$\therefore H_0'(z) = H_0(z) \cdot H_0(z^2) \cdot H_0(z^4)$

The 2nd branch: X(n)

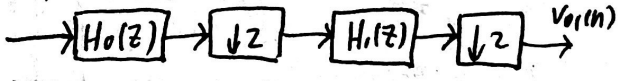


X(n)

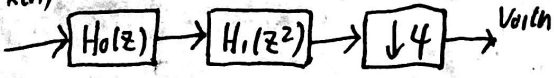


$\therefore H_1'(z) = H_0(z) \cdot H_0(z^2) \cdot H_1(z^4)$

The 3rd branch: X(n)

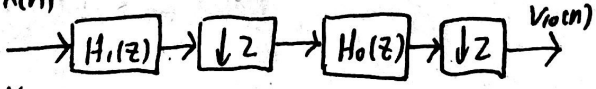


X(n)

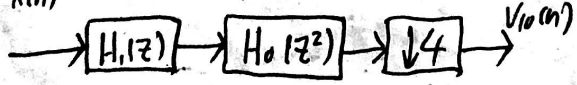


$\therefore H_2'(z) = H_0(z) \cdot H_1(z^2)$

The 4th branch: X(n)

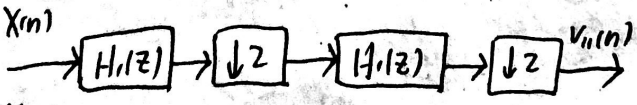


X(n)

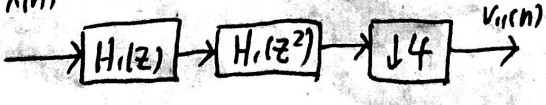


$\therefore H_4'(z) = H_1(z) \cdot H_1(z^2)$

The 5th branch: X(n)

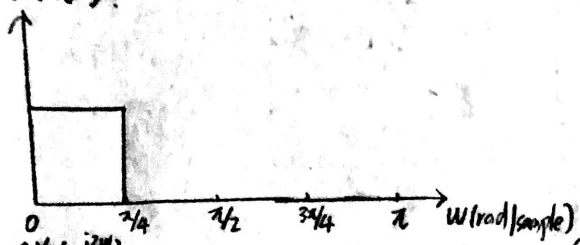


X(n)

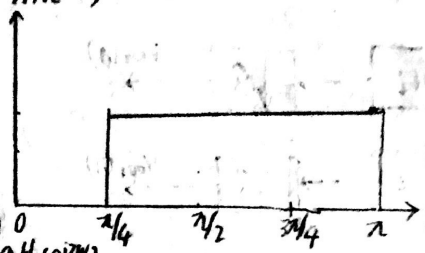


$\therefore H_3'(z) = H_1(z) \cdot H_0(z^2)$

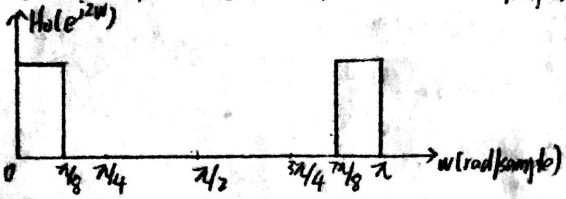
(c) $H_0(e^{jw})$



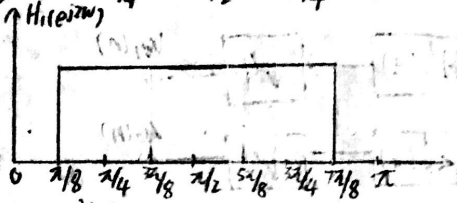
$H_1(e^{jw})$



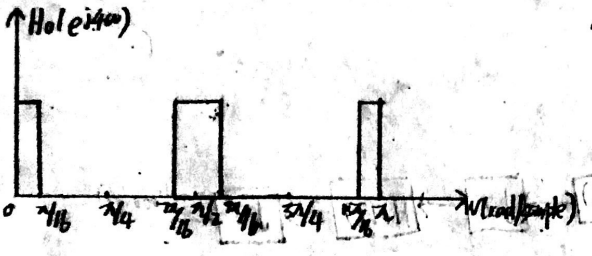
$H_0(e^{j2w})$



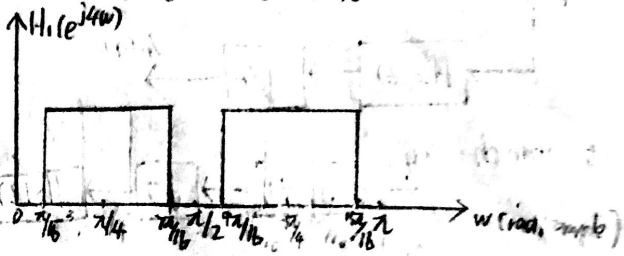
$H_1(e^{j2w})$



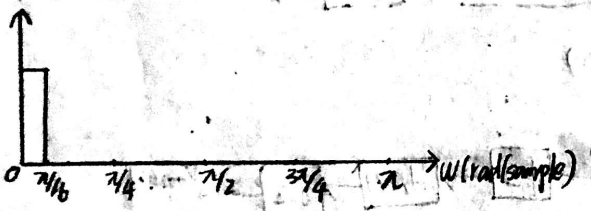
$H_0(e^{j4w})$



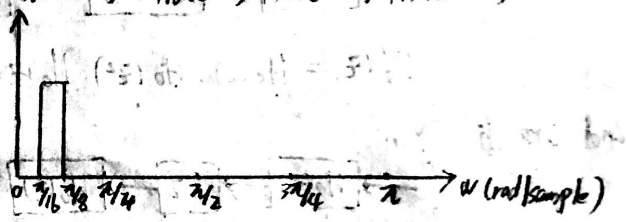
$H_1(e^{j4w})$



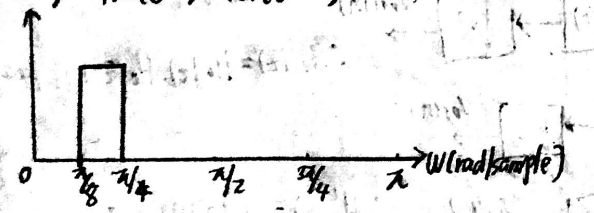
$H_0'(e^{jw}) = H_0(e^{jw}) \cdot H_0(e^{j2w}) \cdot H_0(e^{j4w})$



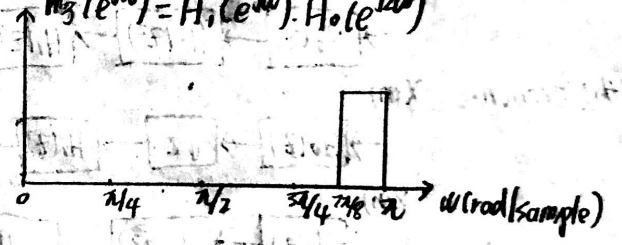
$H_1'(e^{jw}) = H_0(e^{jw}) \cdot H_0(e^{j2w}) \cdot H_1(e^{j4w})$



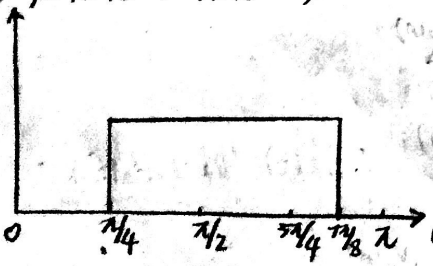
$H_2'(e^{jw}) = H_0(e^{jw}) \cdot H_1(e^{j2w})$



$H_3'(e^{jw}) = H_1(e^{jw}) \cdot H_0(e^{j2w})$



$H_4'(e^{jw}) = H_1(e^{jw}) \cdot H_1(e^{j2w})$



(d)

The Matlab code to get the required coefficients and show the magnitude response of the filters $H_0'(z), H_1'(z), H_2'(z), H_3'(z)$, and $H_4'(z)$ is

```
b0;b1;
n0=length(b0);
n1=length(b1);
% Get H0(z^2)
n0_2=2*n0;
b0_2=zeros(1,n0_2);
b0_2(1,1:2:end)=b0;
% Get H0(z^4)
n0_4=4*n0;
b0_4=zeros(1,n0_4);
b0_4(1,1:4:end)=b0;
% Get H1(z^2)
n1_2=2*n1;
b1_2=zeros(1,n1_2);
b1_2(1,1:2:end)=b1;
% Get H1(z^4)
n1_4=4*n1;
b1_4=zeros(1,n1_4);
b1_4(1,1:4:end)=b1;
%Get H_0(z)
b_0=conv(b0,b0_2);
b_0=conv(b_0,b0_4);
% Compute and plot the spectrum of the filter H_0(z)
h_0=freqz(b_0,1,w);
hmag_0=abs(h_0);
%Get H_1(z)
b_1=conv(b0,b0_2);
b_1=conv(b_1,b1_4);
% Compute and plot the spectrum of the filter H_1(z)
h_1=freqz(b_1,1,w);
hmag_1=abs(h_1);
%Get H_2(z)
b_2=conv(b0,b1_2);
% Compute and plot the spectrum of the filter H_2(z)
h_2=freqz(b_2,1,w);
hmag_2=abs(h_2);
%Get H_3(z)
b_3=conv(b1,b0_2);
% Compute and plot the spectrum of the filter H_3(z)
h_3=freqz(b_3,1,w);
```

```
hmag_3=abs(h_3);
%Get H_4(z)
b_4=conv(b1,b1_2);
% Compute and plot the spectrum of the filter H_4(z)
h_4=freqz(b_4,1,w);
hmag_4=abs(h_4);
figure(1)   % Plot the magnitude response of the resulting
filter H_0(z)
hmag_0=unwrap(hmag_0,tol);hmag_1=unwrap(hmag_1,tol);
hmag_2=unwrap(hmag_2,tol);hmag_3=unwrap(hmag_3,tol);
hmag_4=unwrap(hmag_4,tol);
plot(w/pi,hmag_0,'b');hold on;
plot(w/pi,hmag_1,'b');hold on;
plot(w/pi,hmag_2,'b');hold on;
plot(w/pi,hmag_3,'b');hold on;
plot(w/pi,hmag_4,'b');hold on;
ylim([0 1.5])
title('Magnitude Response Plot (filter H^\prime_i(z))')
ylabel('Magnitude ')
xlabel('Normalized Frequency (\times\pi rad/sample)')
```

The figure of the magnitude response of the four filters are shown in Fig. 5.

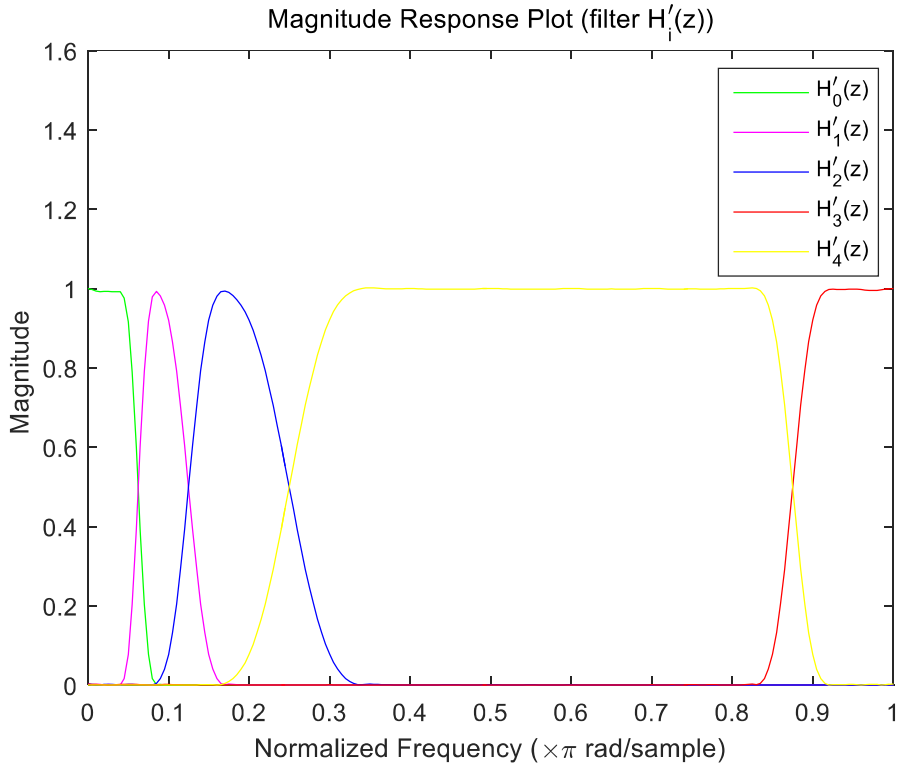


Fig. 5. The magnitude response of the five resulting filters

4. Solve:

(a)

The Matlab code to implement the required analysis and synthesis parts of a QMF is

```
% To get v_n
function v_n=down_sample(x_n,h_b)
v_temp=conv(x_n,h_b);
v_n=v_temp(1,1:2:end);
end
% To get y_n
function y_n=up_sample(v_n,g_b)
n=length(v_n);
n=2*n;
v_temp=zeros(1,n);
v_temp(1,1:2:end)=v_n;
y_n=conv(v_temp,g_b);
end
```

(b)

(b.i)

Using the implementation of the analysis part of the QMF, written in part (a) as a primitive, write a routine to implement the subband decomposition. The Matlab code is

```
% Implement the subband decomposition
% Get v000_n
v000_n=down_sample(x_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
% Get v001_n
v001_n=down_sample(x_n,h0_b2);
v001_n=down_sample(v001_n,h0_b2);
v001_n=down_sample(v001_n,h1_b2);
% Get v01_n
v01_n=down_sample(x_n,h0_b2);
v01_n=down_sample(v01_n,h1_b2);
% Get v10_n
v10_n=down_sample(x_n,h1_b2);
v10_n=down_sample(v10_n,h0_b2);
% Get v11_n
v11_n=down_sample(x_n,h1_b2);
v11_n=down_sample(v11_n,h1_b2);
```

(b.ii)

Using the implementation of the synthesis part of the QMF, written in part (a) as a primitive, write a routine to implement the subband decomposition. The Matlab code is

```
% Get y00_n
y000_n=up_sample(v000_n,g0_b2);
y001_n=up_sample(v001_n,g1_b2);
y00_temp=y000_n+y001_n;
y00_n=up_sample(y00_temp,g0_b2);
% Get y01_n (I have to delay v01_n to make it have the same
length as y00_temp)
delay=length(y00_temp)-length(v01_n);
num=length(y00_temp);
v01_new=zeros(1,num);
v01_new(1,delay+1:1:end)=v01_n;
y01_n=up_sample(v01_new,g1_b2);
% Get y10_n (I have to delay v10_n to make it have the same
length as y00_temp)
v10_new=zeros(1,num);
v10_new(1,delay+1:1:end)=v10_n;
y10_n=up_sample(v10_new,g0_b2);
% Get y11_n (I have to delay v11_n to make it have the same
length as y00_temp)
v11_new=zeros(1,num);
v11_new(1,delay1+1:1:end)=v11_n;
y11_n=up_sample(v11_new,g1_b2);
% Get y_n
y0_temp=y00_n+y01_n;
y0_n=up_sample(y0_temp,g0_b2);
y1_temp=y10_n+y11_n;
y1_n=up_sample(y1_temp,g1_b2);
y_n=y0_n+y1_n;
```

(b.iii)

The Matlab code draw figures of the input sequence x(n) and the output sequence y(n) is

```
x_n=sampdata;
figure(1)
stem(0:length(x_n)-1,x_n)
title('x(n) The input sequence')
xlabel('Sample number')
ylabel('Amplitude')
figure(2)
stem(0:length(y_n)-1,y_n)
title('y(n) The output sequence')
xlabel('Sample number')
ylabel('Amplitude')
```

The figures of the input sequence x(n) and the output sequence y(n) are shown in Fig. 6 and Fig. 7 respectively, from which we can see that the output sequence is just a

delayed version of the input approximately.

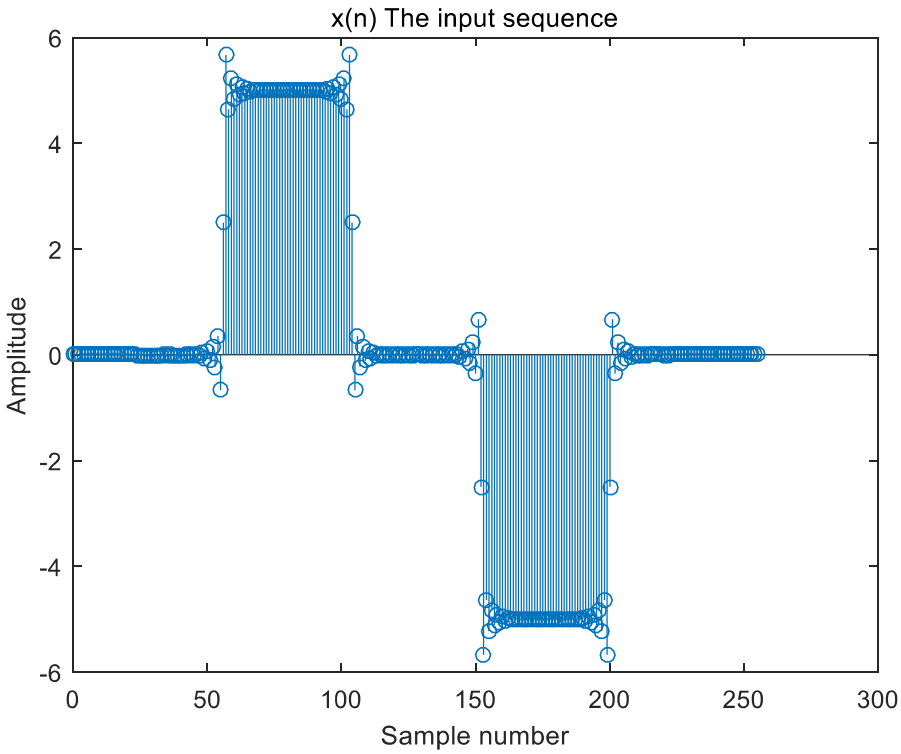


Fig. 6. The input sequence x(n)

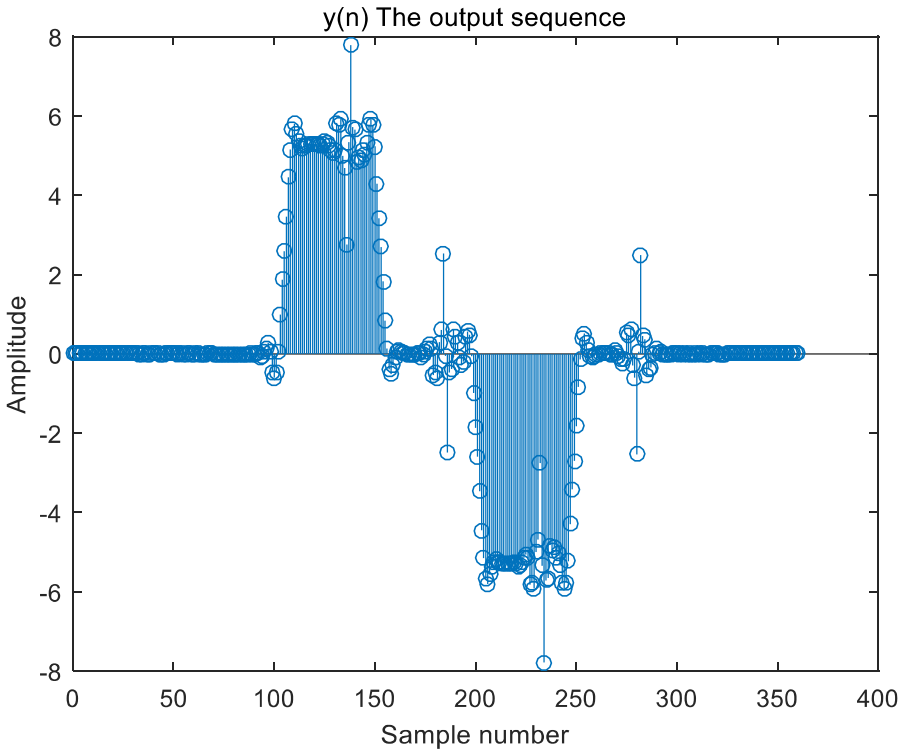


Fig. 7. The output sequence y(n)

(c)

The Matlab code to perform hard thresholding the resulting subband signals is

```
function v_n=hard_threshold(v_n,value)
num=length(v_n);
for n=1:1:num
    if abs(v_n(1,n))<value
        v_n(1,n)=0;
    end
end
end
```

(d)

The Matlab code to listen to the two sequences is

```
%%
soundsc(xn_prb2)
%%
xn_prb2_noise=xn_prb2+0.1*randn(size(xn_prb2));
soundsc(xn_prb2_noise)
```

(e)

The Matlab code to do the operation of denoising is

```
x_n=xn_prb2_noise;
figure(1)
stem(0:length(x_n)-1,x_n)
title('The noised signal')
xlabel('Sample number')
ylabel('Amplitude')
% Get v000_n
v000_n=down_sample(x_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
% Get v001_n
v001_n=down_sample(x_n,h0_b2);
v001_n=down_sample(v001_n,h0_b2);
v001_n=down_sample(v001_n,h1_b2);
% Get v01_n
v01_n=down_sample(x_n,h0_b2);
v01_n=down_sample(v01_n,h1_b2);
% Get v10_n
v10_n=down_sample(x_n,h1_b2);
v10_n=down_sample(v10_n,h0_b2);
% Get v11_n
v11_n=down_sample(x_n,h1_b2);
v11_n=down_sample(v11_n,h1_b2);
% Threshold the highpass components
v001_n=hard_threshold(v001_n,0.390);
```

```
v01_n=hard_threshold(v01_n,0.390);
v10_n=hard_threshold(v10_n,0.390);
v11_n=hard_threshold(v11_n,0.353);
% Implement the subband reconstruction
% Get y00_n
y000_n1=up_sample(v000_n,g0_b2);
y001_n1=up_sample(v001_n1,g1_b2);
y00_temp1=y000_n1+y001_n1;
y00_n1=up_sample(y00_temp1,g0_b2);
% Get y01_n (I have to delay v01_n to make it have the same
length as y00_temp)
delay1=length(y00_temp1)-length(v01_n1);
num1=length(y00_temp1);
v01_new1=zeros(1,num1);
v01_new1(1,delay1+1:1:end)=v01_n1;
y01_n1=up_sample(v01_new1,g1_b2);
% Get y10_n (I have to delay v10_n to make it have the same
length as y00_temp)
v10_new1=zeros(1,num1);
v10_new1(1,delay1+1:1:end)=v10_n1;
y10_n1=up_sample(v10_new1,g0_b2);
% Get y11_n (I have to delay v11_n to make it have the same
length as y00_temp)
v11_new1=zeros(1,num1);
v11_new1(1,delay1+1:1:end)=v11_n1;
y11_n1=up_sample(v11_new1,g1_b2);
% Get y_n
y0_temp1=y00_n1+y01_n1;
y0_n1=up_sample(y0_temp1,g0_b2);
y1_temp1=y10_n1+y11_n1;
y1_n1=up_sample(y1_temp1,g1_b2);
y_n1=y0_n1+y1_n1;
figure(2)
stem(0:length(y_n1)-1,y_n1)
title('The denoised signal')
xlabel('Sample number')
ylabel('Amplitude')
figure(3)
stem(0:length(xn_prb2)-1,xn_prb2)
title('The original signal')
xlabel('Sample number')
ylabel('Amplitude')
```

(f)

The resulting denoised signal is shown in Fig. 8, which is better than the noisy

signal because I won't listen to any sharp voice when listening to the denoised signal, meaning that just like listening to the original signal.

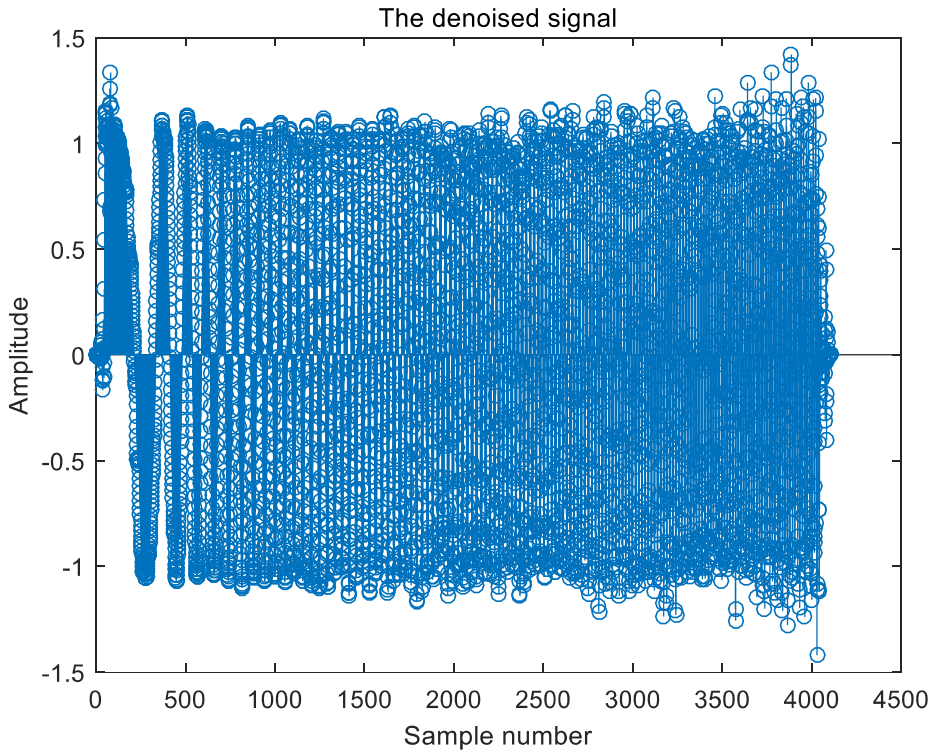


Fig. 8. The denoised signal y(n)

(g)

The Matlab code to do the operation of denoising by changing threshold values is

```
x_n=xn_prb2_noise;
% Get v000_n
v000_n=down_sample(x_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
v000_n=down_sample(v000_n,h0_b2);
% Get v001_n
v001_n=down_sample(x_n,h0_b2);
v001_n=down_sample(v001_n,h0_b2);
v001_n=down_sample(v001_n,h1_b2);
% Get v01_n
v01_n=down_sample(x_n,h0_b2);
v01_n=down_sample(v01_n,h1_b2);
% Get v10_n
v10_n=down_sample(x_n,h1_b2);
v10_n=down_sample(v10_n,h0_b2);
% Get v11_n
v11_n=down_sample(x_n,h1_b2);
v11_n=down_sample(v11_n,h1_b2);
% Threshold the highpass components
```

```
v001_n2=hard_threshold(v001_n,0.200);
v001_n3=hard_threshold(v001_n,1.900);
v01_n2=hard_threshold(v01_n,0.130);
v01_n3=hard_threshold(v01_n,0.960);
v10_n2=hard_threshold(v10_n,0.120);
v10_n3=hard_threshold(v10_n,0.950);
v11_n2=hard_threshold(v11_n,0.120);
v11_n3=hard_threshold(v11_n,0.930);
% Implement the subband reconstruction
% Get y00_n
y000_n2=up_sample(v000_n,g0_b2);
y001_n2=up_sample(v001_n2,g1_b2);
y00_temp2=y000_n2+y001_n2;
y00_n2=up_sample(y00_temp2,g0_b2);

y000_n3=up_sample(v000_n,g0_b2);
y001_n3=up_sample(v001_n3,g1_b2);
y00_temp3=y000_n3+y001_n3;
y00_n3=up_sample(y00_temp3,g0_b2);
% Get y01_n (I have to delay v01_n to make it have the same
length as y00_temp)
delay2=length(y00_temp2)-length(v01_n2);
num2=length(y00_temp2);
v01_new2=zeros(1,num2);
v01_new2(1,delay2+1:1:end)=v01_n2;
y01_n2=up_sample(v01_new2,g1_b2);

delay3=length(y00_temp3)-length(v01_n3);
num3=length(y00_temp3);
v01_new3=zeros(1,num3);
v01_new3(1,delay3+1:1:end)=v01_n3;
y01_n3=up_sample(v01_new3,g1_b2);
% Get y10_n (I have to delay v10_n to make it have the same
length as y00_temp)
v10_new2=zeros(1,num2);
v10_new2(1,delay2+1:1:end)=v10_n2;
y10_n2=up_sample(v10_new2,g0_b2);

v10_new3=zeros(1,num3);
v10_new3(1,delay3+1:1:end)=v10_n3;
y10_n3=up_sample(v10_new3,g0_b2);
% Get y11_n (I have to delay v11_n to make it have the same
length as y00_temp)
v11_new2=zeros(1,num2);
```

```
v11_new2(1,delay2+1:1:end)=v11_n2;
y11_n2=up_sample(v11_new2,g1_b2);

v11_new3=zeros(1,num3);
v11_new3(1,delay3+1:1:end)=v11_n3;
y11_n3=up_sample(v11_new3,g1_b2);
% Get y_n
y0_temp2=y00_n2+y01_n2;
y0_n2=up_sample(y0_temp2,g0_b2);
y1_temp2=y10_n2+y11_n2;
y1_n2=up_sample(y1_temp2,g1_b2);
y_n2=y0_n2+y1_n2;

y0_temp3=y00_n3+y01_n3;
y0_n3=up_sample(y0_temp3,g0_b2);
y1_temp3=y10_n3+y11_n3;
y1_n3=up_sample(y1_temp3,g1_b2);
y_n3=y0_n3+y1_n3;
figure(4)
stem(0:length(y_n2)-1,y_n2)
title('The denoised signal after decreasing the threshold
value')
xlabel('Sample number')
ylabel('Amplitude')
figure(5)
stem(0:length(y_n3)-1,y_n3)
title('The denoised signal after increasing the threshold
value')
xlabel('Sample number')
ylabel('Amplitude')
```

The resulting denoised signals are shown in Fig. 9 and Fig. 10 respectively. When increasing the value of threshold, I won't listen to any sharp voice. However, when decreasing it, I will still listen to some sharp voice. I can draw a conclusion that decreasing the value of threshold will make the filtered result worse.

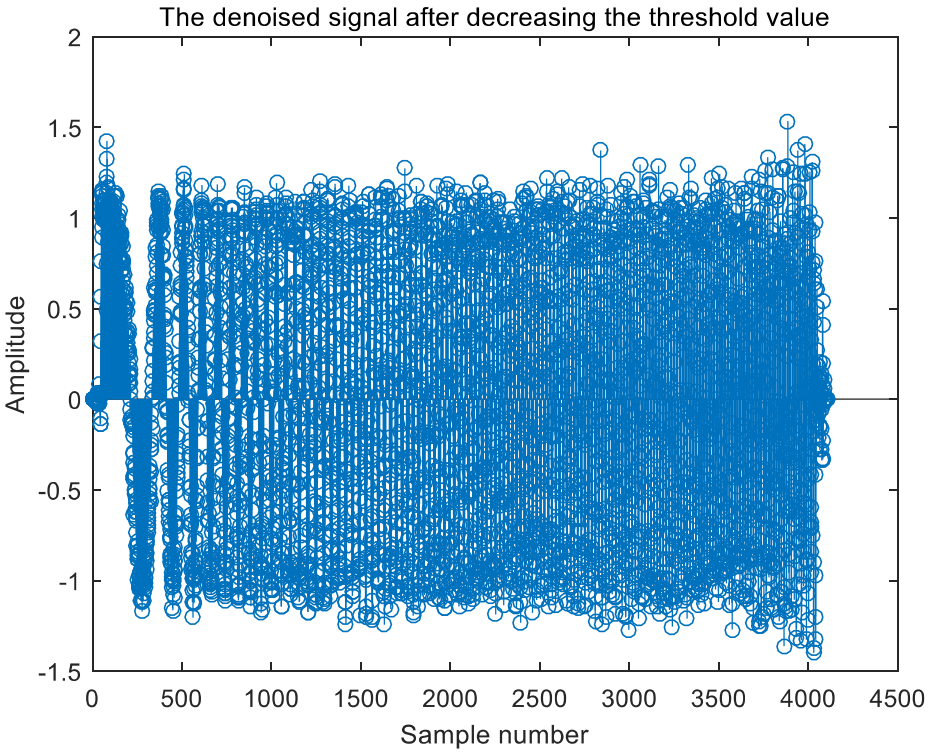


Fig. 9. The denoised signal y1(n) after decreasing the threshold value

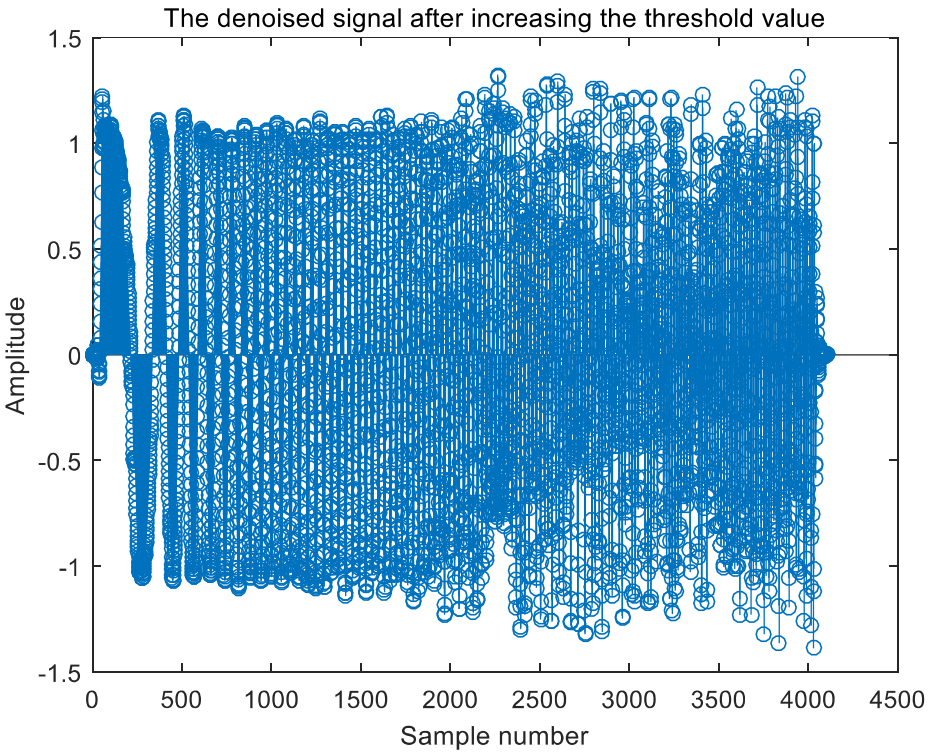


Fig. 10. The denoised signal y2(n) after increasing the threshold value