Multi-rate Digital Signal Processing

Experiment: 2

Haricharan B

Introduction

Filter banks are essential for transmitting signals over a transmission line. In our 2-channel DFT filter bank, we utilize a low pass filter $H^0(z)$, and set $H^1(z) = H^0(ze^{-j\pi}) = H^0(-z)$.

As usual, in the “good” case, to achieve alias cancellation, we set $G^0(z) = H^1(-z)$ and $G^1(z) = -H^0(-z)$. This, however, simplifies to $G^0(z) = H^0(z)$ and $G^1(z) = -H^1(z)$.

This gives T(z) a nice expression, $T(z) = \frac{1}{2}(H^0(z)G^0(z) + H^1(z)G^1(z)) = \frac{1}{2}(H^0(z)^2 - (H^1(z)^2)$

In general, the above will achieve alias cancellation, but not Perfect Reconstruction. If we did want PR, we would have to set $H^0(z) = a + bz^{-n_0}$ for some odd n_0, which ends up not being a very good filter.

Designing the filter

To design the filter in our experiment, we use the following parameters:

- N = 199
- Fpass = 0.45 (Normalized units)
- Fstop = 0.55 (Normalized units)
- Wpass = 1
- Wstop = 1e-3

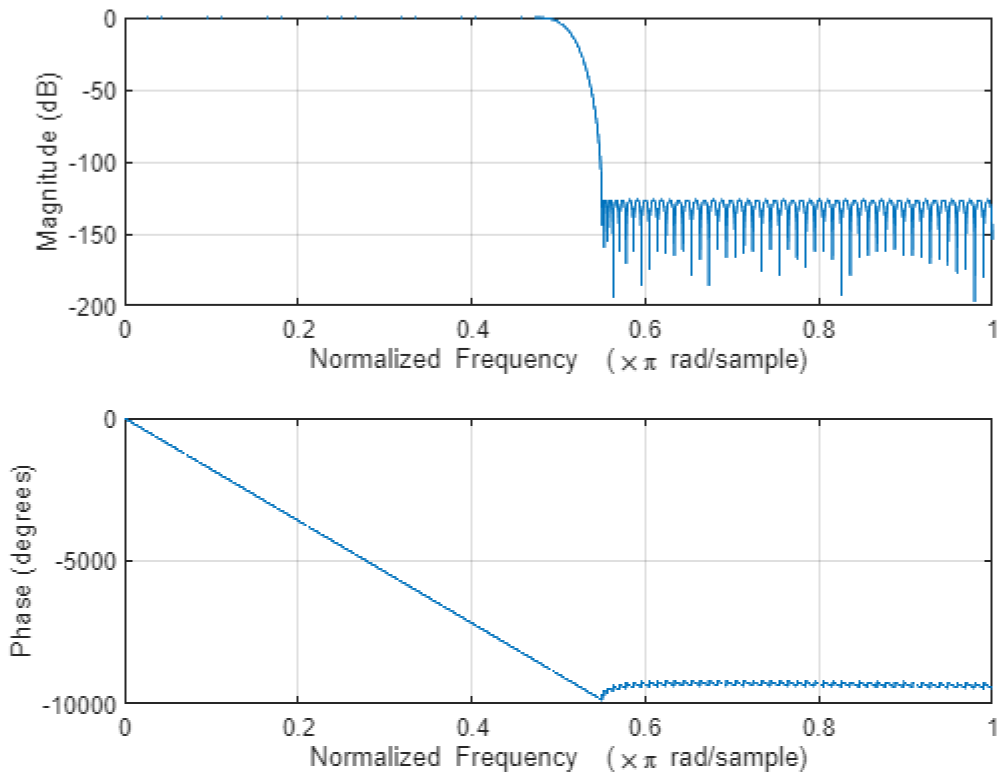
$$\frac{-10\log_{10}(\delta_1\delta_2) - 13}{2.324\Delta\omega}$$

Note that the value of N chosen is in line with the above equation.

We use the `firpm` function in MATLAB to design a Parks-McClellan filter, with these input parameters. Plotting this gives us results as below:

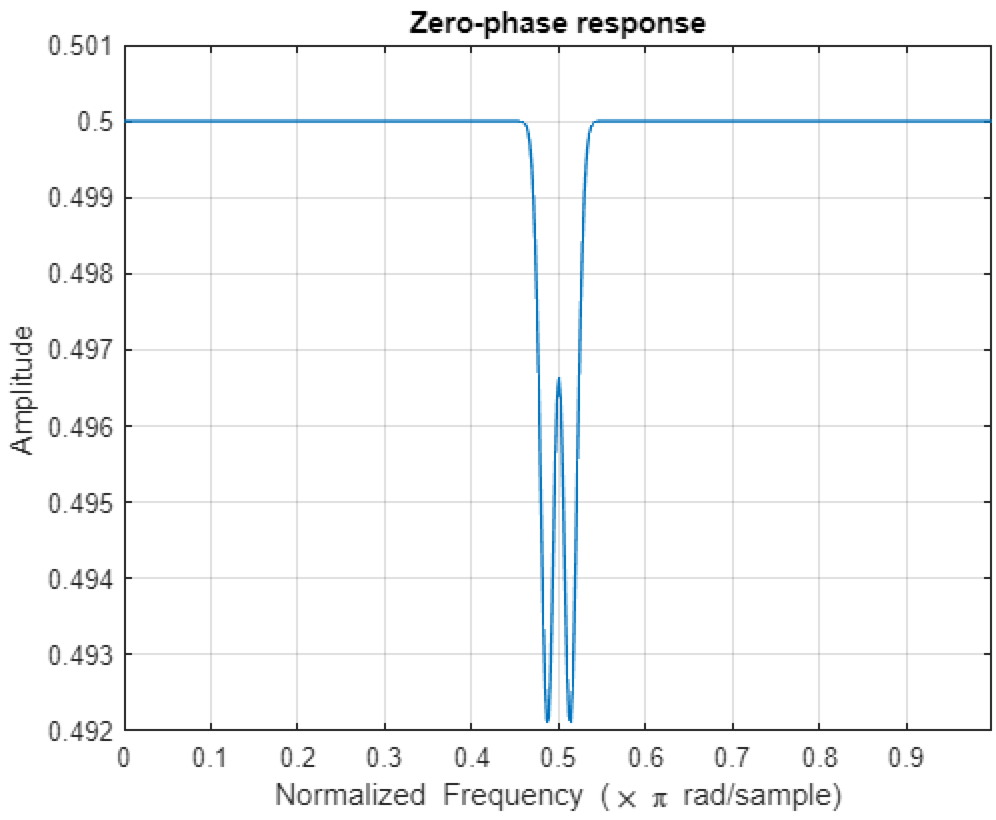
```
speech8khz.wav
    "the value of N is"    "199"

Magnitude response of H0(z)
```



This leads to a $T(z)$, with a zerophase response (for the “good” case with alias cancellation) as below:

We observe that the filter is more or less flat and that the ripple is very small.



Code

The MATLAB code is attached here:

```
f("speech8khz.wav");
f("music16khz.wav");

function sound(audio, Fs)
% Sound function, that plays all sound files one-by-one instead of simultaneously.
y = audioplayer(audio,Fs);
playblocking(y);
end

function f(filename)
% displaying file name
disp(filename)

% Reading the audio file
[audio, SamplingRate] = audioread(filename);

% Parameters
```

```
N = 199; % for this code to work, N should be odd
disp(["the value of N is", num2str(N)])
Fpass = 0.45;
Fstop = 0.55;
Wpass = 1;
Wstop = 1e-3;

disp("Magnitude response of H0(z)")

% Designing the filter
array = firpm (N, [0 Fpass Fstop 1], [1 1 0 0], [Wpass Wstop]);
Hsuper0 = dfilt.dffir(array);


% Plotting the filter, commented out now for clarity
%{
figure;
freqz(Hsuper0.Numerator, 1, SamplingRate)
%}

% Finding R, defined to be H0(z) * G0(z) = H^0(z)^2
R = conv(Hsuper0.Numerator, Hsuper0.Numerator);

% Setting T (z) = (R (z) - R(-z))/2 and plotting it
for i=1:2:length(R)
    R(i) = 0;
end
T = R;

% Plotting the zp response, commented out now for clarity
%{
figure;
zerophase(T, 1)
%}

% Finding the polyphase components
Hsuper0_0 = Hsuper0.Numerator(1:2:end);
Hsuper0_1 = Hsuper0.Numerator(2:2:end);

% Getting the two outputs of analysis FB
[vsuper0_d, vsuper1_d] = analysis(audio, Hsuper0_0, Hsuper0_1);

% output of synthesis FB for the good case
disp("Good case")
y_good = synthesis(vsuper0_d, vsuper1_d, Hsuper0_1, Hsuper0_0);
figure;
freqz(y_good, 1, SamplingRate);

% output of synthesis FB for the bad case
disp("Bad case")
y_bad = synthesis(vsuper0_d, vsuper1_d, Hsuper0_0, Hsuper0_1);
figure;
```

```
freqz(y_bad, 1, SamplingRate);

% Playing the audio files
% sound(audio, SamplingRate);
% sound(y_good, SamplingRate);
% sound(y_bad, SamplingRate);

end

function [vsuper0_d, vsuper1_d] = analysis(x, Hsuper0_0, Hsuper0_1)

% Finding polyphase components of x
x_0 = x (2:2: end);
x_1 = x (1:2: end) ;

tsuper0 = conv(x_0, Hsuper0_0);
tsuper1 = conv(x_1, Hsuper0_1);

% Carrying out the matrix multiplication by DFT matrix
vsuper0_d = tsuper0 + tsuper1;
vsuper1_d = tsuper0 - tsuper1;

end

function [y] = synthesis(vsuper0_d, vsuper1_d, K_0, K_1)

% Carrying out the matrix multiplication by DFT matrix
k_0_input = vsuper0_d + vsuper1_d;
k_1_input = vsuper0_d - vsuper1_d;

y_0 = filter(K_0, 1, k_0_input);
y_1 = filter(K_1, 1, k_1_input);

% Creating the commutator
y = zeros (2 * max(length (y_0), length(y_1)), 1);
for i = 1:length (y_0)
    y (2*i) = y_0 (i) ;
end
for i = 1:length (y_1)
    y (2*i - 1) = y_1(i) ;
end
end
```

Deliverables

First two deliverables attached above, Other plots at the end.

Conclusions

Speech File

We observe that in the “bad” choice of filter:

1. The audio is of poor quality and sounds muffled with some ringing sound. This is because there is **no alias cancellation**, so the quality is very bad.
2. In the magnitude spectrum also, aliasing is observed.

We observe that in the “good” choice of filter:

1. The audio is of similar quality to original. This is because there is **alias cancellation**, so the quality is very good.
2. In the magnitude spectrum also, no aliasing is observed.

Music File

We observe that in the “bad” choice of filter:

1. The audio is of poor quality and the tones are not very clear. This is because there is **no alias cancellation**, so the quality is very bad.
2. In the magnitude spectrum also, aliasing is observed.

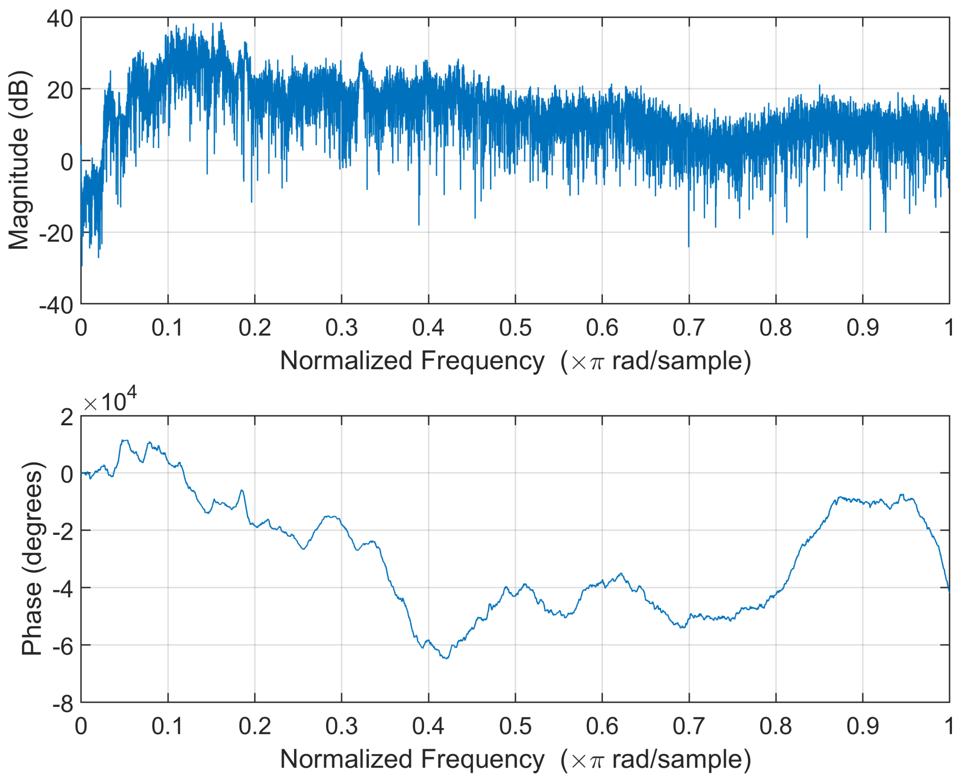
We observe that in the “good” choice of filter:

1. The audio is of similar quality to original.
2. This is because there is **alias cancellation**, so the quality is very good. In the magnitude spectrum also, no aliasing is observed.

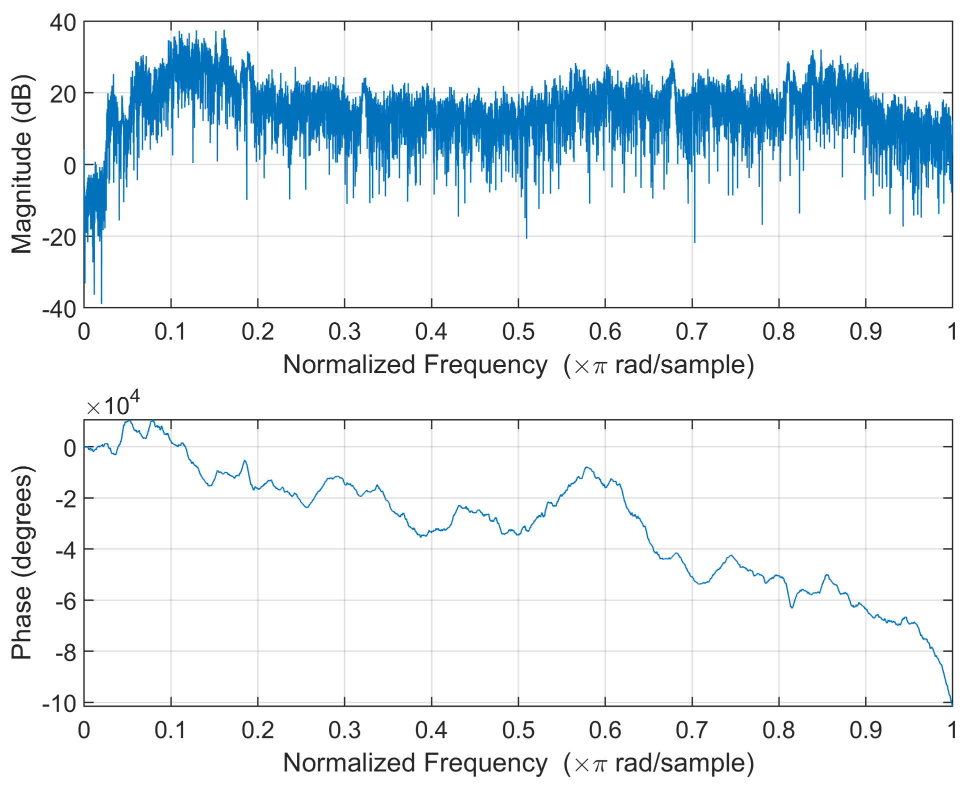
$$M = \frac{-10\log_{10}(\delta_1\delta_2) - 13}{2.324\Delta\omega}$$

```
speech8khz.wav
    "the value of N is"    "199"

Magnitude response of H0(z)
Good case
```

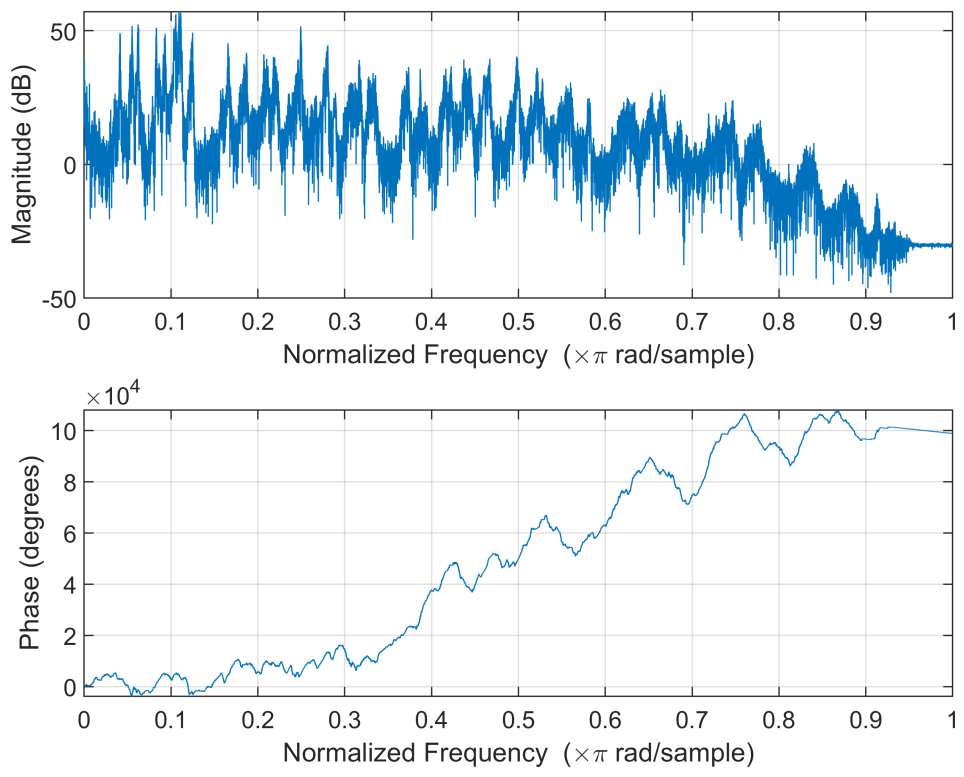


```
Bad case
```

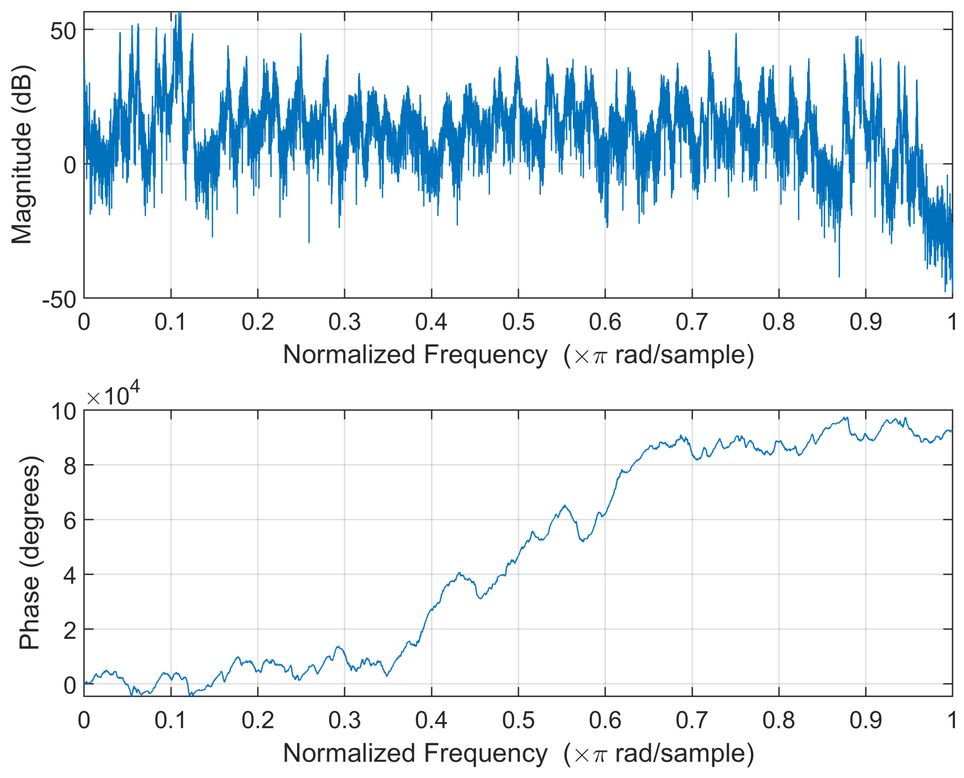


```
music16khz.wav
    "the value of N is"    "199"

Magnitude response of H0(z)
Good case
```


Magnitude (dB)
50
0
-50
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
$\times 10^4$
Phase (degrees)
10
8
6
4
2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)

```
Bad case
```


Magnitude (dB)
50
0
-50
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
$\times 10^4$
Phase (degrees)
10
8
6
4
2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)