Digital Signal Processing Laboratory Report - Lab7: Digital Filter Design and Analysis

Haodong Zhang, 12113010

***Abstract*—This experiment explores the use and analysis of digital filters. Firstly, we introduce the fundamental concepts of digital filters, including their representation through Z-transform or block diagrams. Subsequently, we delve into the design of simple FIR filters, analyze their performance, and apply them in the denoising process of audio signals to observe their effectiveness. Following that, we investigate the design of basic IIR filters, studying the influence of different coefficients on the performance of IIR filters. We then employ actual audio signals to evaluate the performance of IIR filters. In the final segment, we examine the parameters of a low-pass filter, exploring their effects. Additionally, we investigate the impact of truncation length on filter performance and conduct analysis using real audio signals.**

***Index Terms*—Digital Signal Processing, Matlab, Digital Filter Design**

I. Introduction

THIS experiment aims to use Matlab tools to design and analyze digital filters, including simple FIR and IIR filters. We introduced their basic concepts, formulated their expressions using Z-transform, analyzed the impact of different parameters, and conducted analysis using real language signals to validate the performance of our filters.

II. Background on Digital Filters

In applications of digital signal processing, there is often a need to alter the relative amplitudes of frequency components or eliminate unwanted frequencies in a signal. This process is commonly referred to as filtering. Digital filters find applications in various fields, with audio systems being a notable example explored in previous laboratories. These filters empower listeners to customize the bass (low-frequency energy) and treble (high-frequency energy) of audio signals.

Designing digital filters involves employing techniques from both the frequency domain and the time domain. This is because filter design specifications are frequently presented in the frequency domain, while the actual implementation of filters occurs in the time domain through a difference equation. Typically, frequency domain analysis is conducted using tools such as the z-transform and the discrete-time Fourier Transform (DTFT).

This is a lab report of the Digital Signal Processing (EE323) in the Southern University of Science and Technology (SUSTech).

Haodong Zhang is a undergraduate student majoring in communication engineering, Department of Electronic and Electrical Engineering, Southern University of Science and Technology (SUSTech), Shenzhen 518055, China (email: 12113010@mail.sustech.edu.cn).

In general, a linear and time-invariant causal digital filter with the input $x[n]$ and output $y[n]$ can be specified by a difference equation as below:

$$y[n] = \sum_{i=0}^{N-1} b_i x[n-i] - \sum_{k=1}^{M-1} a_k y[n-k] \tag{1}$$

where b_i and a_k are coefficients which parameterize the filter. This filter have $N-1$ zeros and $M-1$ poles. Each new value of the output signal, $y[n]$, is determined by past values of the output, and by present and past values of the input. The impulse response, $h[n]$, is the response of the filter to an input of $\delta[n]$, and is therefore the solution to the recursive difference equation as below:

$$h[n] = \sum_{i=0}^{N-1} b_i\ \delta[n-i] - \sum_{k=1}^{M-1} a_k\ h[n-k] \tag{2}$$

According to the expression above, there are two general classes of digital filters: infinite impulse response (IIR) and finite impulse response (FIR). The FIR case occurs when $a_k = 0$,for all k. Such a filter have no poles and only zeros. And the $h[n]$ is no longer recursive, the impulse response has finite duration N. For the case where $a_k \neq 0$, the difference equation usually represents an IIR filter. And then the $h[n]$ will usually generate an impulse response which has non-zero values as $n \to \infty$.

Then we can use the z-transform to analyze the frequency response of filters. Firstly, the z-transform of a discrete-time signal, $x[n]$ is that:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \tag{3}$$

And the DTFT is a special case of the z-transform where z is evaluated on the unit circle in the complex plane which can be expressed as follows:

$$X(e^{j\omega}) = X(z)|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \tag{4}$$

Then we can utilize the knowledge of the z-transform to apply z-transform on both sides of the equation and obtain that

$$Y(z)\left(1 + \sum_{k=1}^{M-1} a_k z^{-k}\right) = X(z) \sum_{i=0}^{N-1} b_i z^{-i} \tag{5}$$

We can further derive to the z-transform of the impulse response:

$$H(z) = \frac{\sum_{i=0}^{N-1} b_i z^{-i}}{1 + \sum_{k=1}^{M-1} a_k z^{-k}} \tag{6}$$

From this result, we may compute the frequency response of the filter by evaluating $H(z)$ on the unit circle, then the equation(6) can be change to:

$$H(e^{j\omega}) = \frac{\sum_{i=0}^{N-1} b_i e^{-j\omega i}}{1 + \sum_{k=1}^{M-1} a_k e^{-j\omega k}} \tag{7}$$

Therefore, we can use these equations to represent an analyze the FIR and IIR filters, respectively.

III. Design of a Simple FIR Filter

A. Basic Principle of FIR Filter Design

Zeros in the transfer function represent frequencies that are not passed through the filter. For a FIR filter with real-valued impulse response, the zeros must be complex conjugates of one another. Here, we assume the zeros are:

$$z_1 = e^{j\theta}, z_2 = e^{-j\theta} \tag{8}$$

where θ is the angle of z_1 relative to the positive real axis. And then the transfer function for this filter is determined using equation(8):

$$H_f(z) = (1 - z_1 z^{-1})(1 - z_2 z^{-1}) = 1 - 2\cos\theta z^{-1} + z^{-2} \tag{9}$$

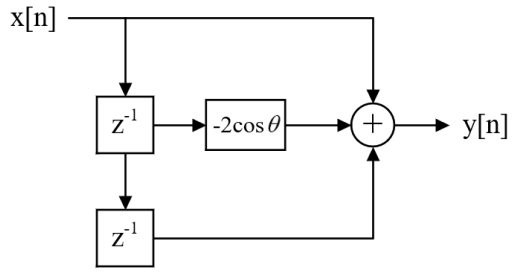
Applying inverse z-transform on equation(9), the impulse response for $H_f(z)$ is:

$$h_f[n] = \delta[n] - 2cos\theta\delta[n-1] + \delta[n-2] \tag{10}$$

Replacing $\delta[n]$ with $x[n]$, the difference equation for $H_f(z)$ is:

$$y_f[n] = x[n] - 2cos\theta x[n-1] + x[n-2] \tag{11}$$

The corresponding system diagram is:



The system diagram for equation(11)

The fact that $H_f(z)$ has zeros at $e^{\pm j\theta}$ means that the filter will not pass pure sine waves(from Euler's formula) at $\omega = \theta$. For different values of θ for i) $\theta = \pi/6$, ii) $\theta = \pi/3$ iii) $\theta = \pi/2$, we can compute and plot the magnitude of the filter's frequency response with Matlab. The magnitude of the response are shown in Fig. 1. As shown in the figures, the magnitude responses reaches 0 where $\omega = \theta$. Apart from the initial instance, as evident from the graph, the value of θ also influences the magnitude. In this case, as θ increases, the maximum magnitude decreases, and the magnitude of the low-frequency components near $\omega = 0$ gradually increases.

The Matlab codes are:

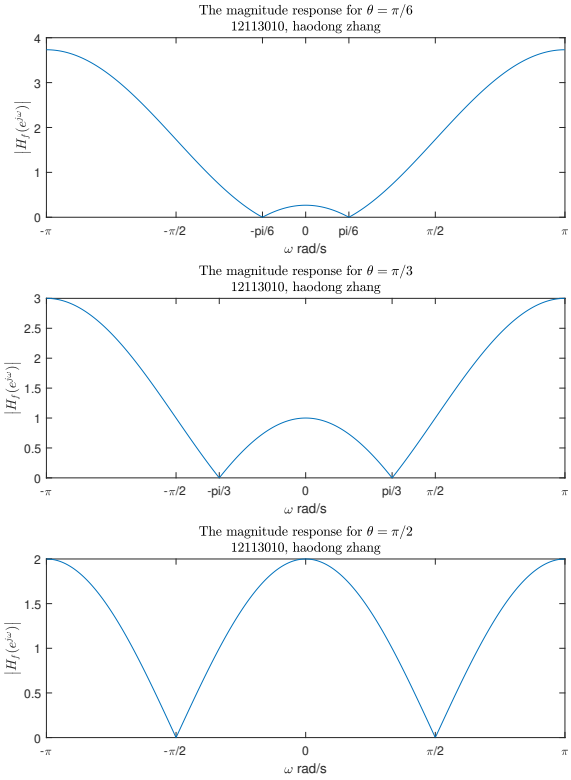


Fig. 1: The magnitude response for different θ

```
w = -pi:0.01:pi;
theata = [pi/6, pi/3, pi/2];
H = {};
for i=1:length(theata)
    H{i} = 1-2*cos(theata(i)).*exp(-1j*w)+(exp(-1j*2*w));
end
subplot(3,1,1);
plot(w,abs(H{1}));
title(['The magnitude response for $\theta=\pi/6$'
    newline '12113010, haodong zhang'], 'Interpreter
    ', 'latex');
xlabel('\omega rad/s');
ylabel('$\left | H_f(e^{j\omega}) \right | $', '
    Interpreter','latex')
xlim([-pi,pi])
xticks([-pi -pi/2 -pi/6 0 pi/6 pi/2 pi]);
xticklabels({'-\pi' '-\pi/2' '-pi/6' '0' 'pi/6' '\pi/2
    ' '\pi'});
subplot(3,1,2);
plot(w,abs(H{2}));
title(['The magnitude response for $\theta=\pi/3$'
    newline '12113010, haodong zhang'], 'Interpreter
    ', 'latex');
xlabel('\omega rad/s');
ylabel('$\left | H_f(e^{j\omega}) \right | $', '
    Interpreter','latex')
xlim([-pi,pi])
xticks([-pi -pi/2 -pi/3 0 pi/3 pi/2 pi]);
xticklabels({'-\pi' '-\pi/2' '-pi/3' '0' 'pi/3' '\pi/2
    ' '\pi'});
subplot(3,1,3);
plot(w,abs(H{3}));
title(['The magnitude response for $\theta=\pi/2$'
    newline '12113010, haodong zhang'], 'Interpreter
    ', 'latex');
```

```
xlabel('\omega rad/s');
ylabel('$\left | H_f(e^{j\omega}) \right | $','
    Interpreter','latex')
xlim([-pi,pi])
xticks ([-pi -pi/2 0 pi/2 pi]);
xticklabels ({'-\pi' '-\pi/2' '0' '\pi/2' '\pi'});
```

B. Using FIR Filter to Remove Interference from Speech Signal

And then we can use the filter designed to remove an undesirable sinusoidal interference from a speech signal. We will analyze a speech signal **nspeech1.mat**. We can plot the time-domain of nspeech1[100:200] and the samples of the DTFT of the audio signal using the DTFT.m. The results are shown in Fig. 2.

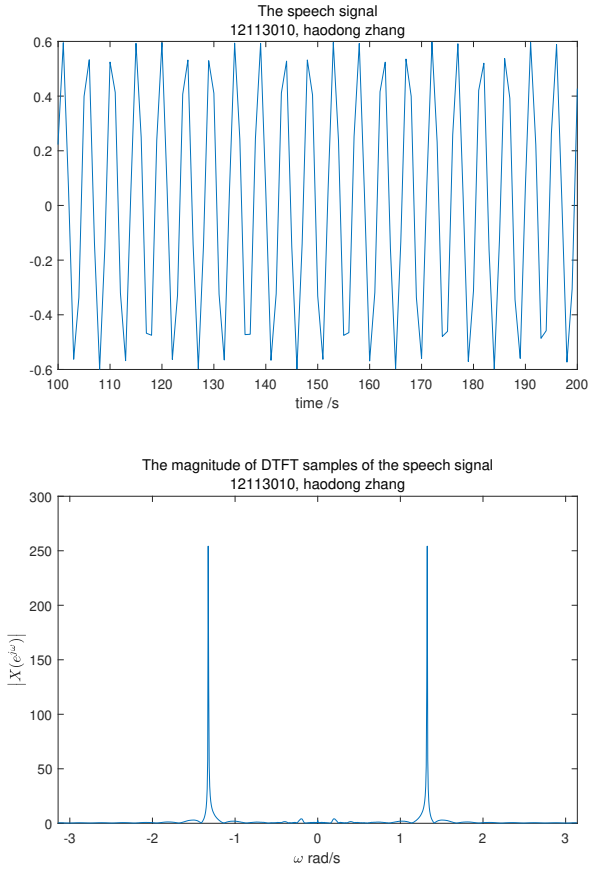


Fig. 2: The time domain samples and DTFT samples of the speech signal

The Matlab codes are that:

```
figure;
load nspeech1.mat;
sound(nspeech1);
subplot(2,1,1)
plot(nspeech1);
xlim([100,200]);
title(['The speech signal' newline '12113010, haodong
    zhang']);
xlabel('time /s');
[X,w] = DTFT(nspeech1(100:1100),0);
subplot(2,1,2)
plot(w,abs(X));
xlim([-pi,pi]);
title(['The magnitude of DTFT samples of the
    speech signal' newline '12113010, haodong zhang
    ']);
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
    Interpreter','latex');
```

where the DTFT function is:

$DTFT.m$:

```
function [X,w] = DTFT(x,M)
N = max(M,length(x));
N = 2^(ceil(log(N)/log(2)));
X = fft(x,N);
w = 2*pi*( (0:(N-1))/N );
w = w - 2*pi*(w>=pi);
X = fftshift(X);
w = fftshift(w);
```

Through the sound signal, we can hear not only the normal human voice but also high-frequency noise. As seen in the graph, the appearance of two prominent peaks is the source of the high-frequency noise. Therefore, our task is to identify the frequency of this high-frequency noise, which corresponds to the peaks in the graph, denoted as ω. This can be easily achieved in MATLAB by using the $maxk()$ function. The code is as follows:

```
[Xmax,Imax] = maxk(abs(X),2)
theata = w(Imax)
```

The results is shown in Fig. 3.

```
Xmax = 1×2
    254.2287  254.2287

Imax = 1×2
     297   729

theata = 1×2
    -1.3254    1.3254
```

Fig. 3: The result of the location of the peaks

Then, once we know the frequency of the noise, we can easily utilize the FIR digital filter we designed earlier. By leveraging its zero characteristics, we can effectively filter out the frequency corresponding to the noise. Therefore, the zeros of the filter we designed should match the frequency of the noise. We can write a Matlab function to implement the filter as follows:

$FIRfilter.m$:

```
function output = FIRfilter(x)
    [X,w] = DTFT(x,0);
    [Xmax,Imax] = max(abs(X));
    theata = w(Imax);
    h = [1, -2*cos(theata), 1];
    output = conv(x,h);
end
```

The implementation principle involves designing an FIR filter for a specified frequency using the method mentioned above, followed by obtaining the filtered result through convolution. The outcome is depicted in Figure 4, where it can be observed that the initial substantial noise has been effectively filtered out. By listening to the audio, it is evident that the original high-frequency noise has nearly disappeared, allowing for the clear perception of human voices. Therefore, the frequency component of the beep is eliminated after filtering, with the frequency component of the human voice left. The filter used is a bandstop filter, for it suppresses the frequency component between 0 and π. In the filtered signal, the human voice stands out clearly without the high frequency beep.

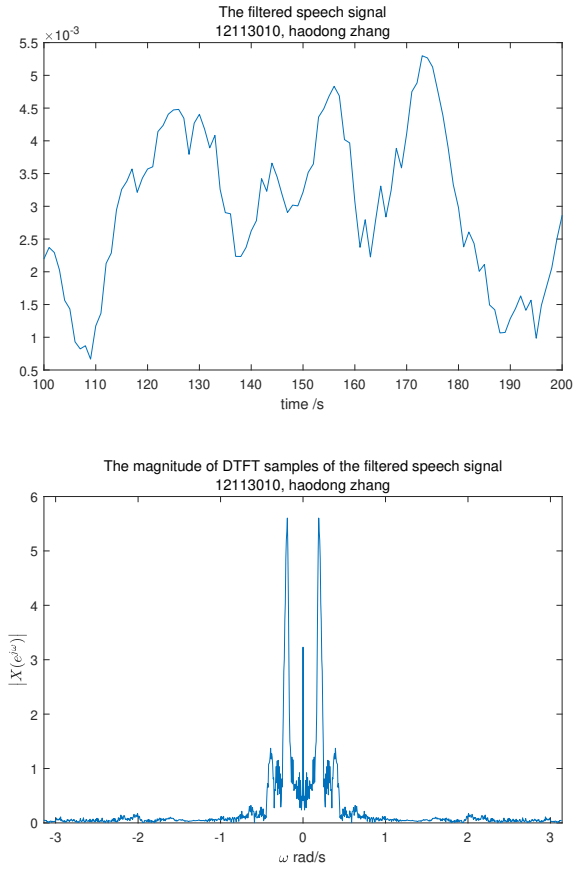


Fig. 4: the time domain samples and DTFT samples of the speech signal after filter

IV. Design of a Simple IIR Filter

A. Basic Principle of IIR Filter Design

Zeros attenuate a filtered signal, while poles, on the contrary, amplify signals in close proximity to their frequency. This characteristic makes poles valuable for the separation of narrow-band signals from adjacent noise. Commonly applied in scenarios such as radio-frequency demodulation, these filters effectively isolate modulated signals from background noise. It's crucial to note that for an IIR filter with a real-valued impulse response, the poles must be complex conjugates of each other. They have the form:

$$p_1 = re^{j\theta}, p_2 = re^{-j\theta} \tag{12}$$

The transfer function for this filter is determined by equation(6):

$$\begin{aligned} H_i(z) &= \frac{1-r}{(1-re^{j\theta}z^{-1})(1-re^{-j\theta}z^{-1})} \\ &= \frac{1-r}{1-2rcos(\theta)z^{-1}+r^2z^{-2}} \end{aligned} \tag{13}$$

Through long division, $H_i(z)$ can be rewritten as:

$$H_i(z) = (1-r)(1+2rcos\theta z^{-1} + (4cos^2\theta - 1)r^2z^{-2} + ...) \tag{14}$$

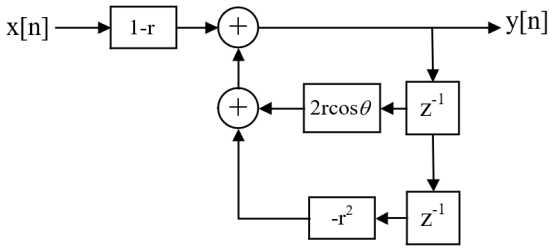
Applying inverse z-transform on equation(13) we also can obtain that:

$$h_i[n] = \frac{1-r}{1-e^{-2j\theta}}(re^{j\theta})^n\mu[n] + \frac{1-r}{1-e^{2j\theta}}(re^{-j\theta})^n\mu[n] \tag{15}$$

Then the difference equation can obtained as follows:

$$y[n] - 2rcos\theta y[n-1] + r^2y[n-2] = (1-r)x[n] \tag{16}$$

The system diagram is depicted below:



The system diagram for equation(16)

From the theory of z-transforms, we understand that a causal filter is stable if and only if its poles are located within the unit circle. This implies stability when $|r| < 1$. However, we will observe that by positioning the poles close to the unit circle, the filter's bandwidth can be made extremely narrow around θ. We will explore the influence of the different r, For $\theta = \frac{\pi}{3}$ and different values of r, the magnitude responses of $H_i(z)$ are depicted in Fig. 5. From the graph, it can be observed that the response function produces a significant peak at the frequency where the poles are located, specifically at positions where θ is equal to $\pm\pi/3$. By examining the graph, it can be observed that as r increases, the peaks near the poles in the response function become narrower, and the peak values also change. As r becomes closer to 1, the peak value of the magnitude response decreases from 0.68 to 0.5. We can mathematically validate this phenomenon through theoretical derivation. From equation(13) we replace z with $e^{j\omega}$, the magnitude response is given by:

$$\left|H_i(e^{j\omega})\right| = \frac{|1-r|}{|e^{j\omega} - re^{j\theta}||e^{j\omega} - re^{-j\theta}|} \tag{17}$$

where $|e^{j\omega} - re^{j\theta}|$ is the length $|AB|$ and $|e^{j\omega} - re^{-j\theta}|$ is the length $|BC|$ which shown as below.

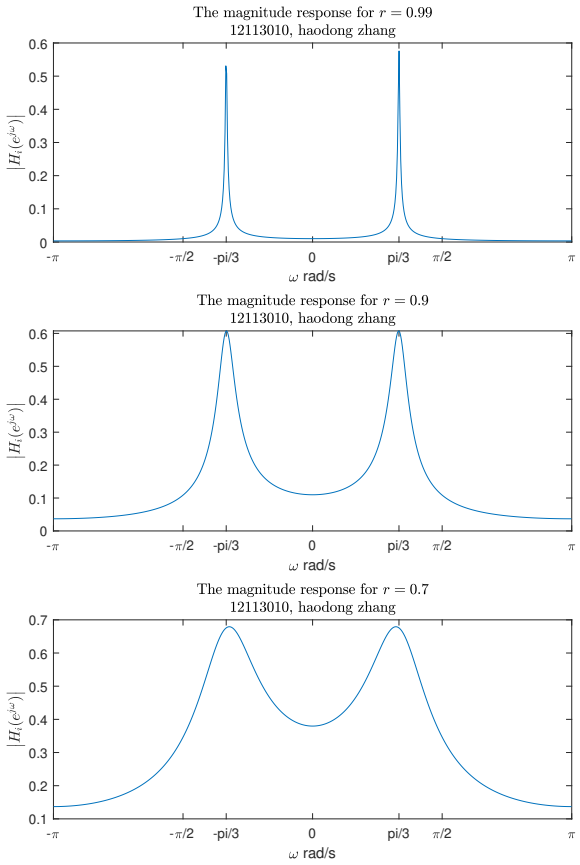
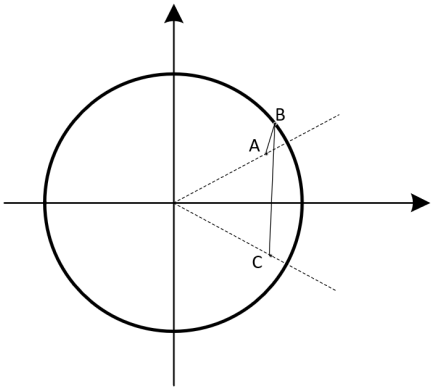


Fig. 5: The magnitude response for different r



schematic diagram

A notable observation is that as r approaches 1, $|1 - r|$ decreases more than $|AB| \cdot |BC|$ does, leading to a decreased overall magnitude level. This trend is evident in Fig. 5, where the peak value of the magnitude response decreases as r becomes closer to 1. Additionally, as r approaches 1, $|AB| \cdot |BC|$ changes more harshly around $\pm\theta$ (i.e., $e^{\pm j\theta}$), because $|AB| \cdot |BC|$ approaches 0 and moves away from 0 more easily around $\pm\theta$. Therefore, in Fig. 5, as r approaches 1, the response becomes increasingly sharp.

B. Using IIR Filter to Separate Modulated Signal From Background Noise

And then we can use the filter designed to separate a modulated sinusoid from background noise. We will analyze a speech signal **pcm.mat**. We can plot the time-domain of nspeech1[100:200] and the samples of the DTFT of the audio signal nspeech1[100:1100] using the DTFT.m. The results are shown in Fig. 6.

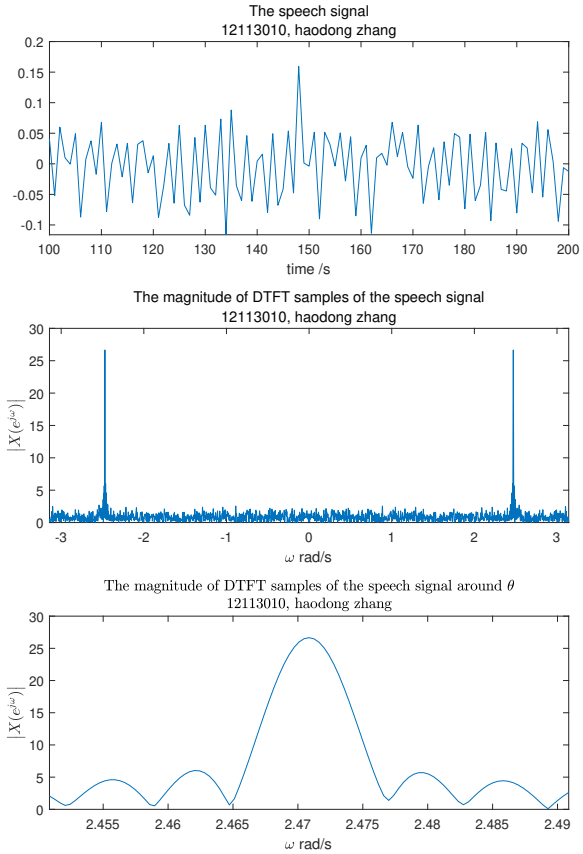


Fig. 6: The time domain samples and DTFT samples of the speech signal

The two peaks in the spectrum correspond to the center frequency of the modulated signal. The low amplitude wideband content represents the background noise. It is evident that the expected speech signal is engulfed in a substantial amount of background noise. This is apparent from the chaotic signal in the time-domain plot, and the auditory experience also reveals

a significant amount of underlying noise. Therefore, we can use the IIR filter described above to amplify the desired signal relative to the background noise.

The MATLAB codes are that:

```
figure;
load pcm.mat;
sound(pcm);
subplot(3,1,1);
plot(pcm);
xlim([100,200]);
title(['The speech signal' newline '12113010, haodong
    zhang']);
xlabel('time /s');

[X,w] = DTFT(pcm(100:1100),10000);
subplot(3,1,2);
plot(w,abs(X));
xlim([-pi,pi]);
title(['The magnitude of DTFT samples of the
    speech signal' newline '12113010, haodong zhang
    ']);
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
    Interpreter','latex');
theata2=-3146/4000*pi
subplot(3,1,3);
plot(w,abs(X));
xlim([3146/4000*pi-0.02,3146/4000*pi+0.02]);
title(['The magnitude of DTFT samples of the
    speech signal around $\theta$' newline '
    12113010, haodong zhang'],'Interpreter','
    latex');
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
    Interpreter','latex');
```

Given that the desired signal is modulated at 3146Hz and sampled at 8kHz, so 2π corresponds to 8kHz. Therefore, the unified frequency of the desired signal is given by: $\omega = \frac{3146}{4000}\pi = 0.7865\pi$. In order to better depict the curve near the poles and make it smoother, we increased the number of points in the DTFT in the above process.

Then, once we know the frequency of the speech signal, we can easily utilize the IIR digital filter we designed earlier. By leveraging its pole characteristics, we can effectively amplify the speech signal and filter out the background noise. Therefore, the poles of the filter we designed should match the frequency of the speech signal. We can write a Matlab function to implement the filter as follows: $IIRfilter.m$:

```
function output = IIRfilter(x)
    theata=3146/4000*pi;
    r = 0.995;
    y = zeros(1,length(x));
    y(1) = (1-r)*x(1);
    y(2) = (1-r)*x(2)+2*r*cos(theata)*y(1);
    for i = 3:length(x)
        y(i) = (1-r)*x(i)+2*r*cos(theata)*y(i-1)-r^2*y(i-2);
    end
    output = y;
end
```

The implementation principle involves designing an IIR filter for a specified frequency using the method mentioned above, followed by obtaining the filtered result through convolution. The outcome is depicted in Figure 7, where it can be observed that the background noise has been effectively filtered out. By listening to the audio, it is evident that the original background noise has nearly disappeared, allowing for the clear perception of human voices. Therefore, the frequency component of the speech signal is amplified after filtering. The value of r should not be excessively large; for instance, taking r as 0.9999999, as shown in Fig. 8, results in a very small magnitude, causing significant attenuation to the signal. Therefore, such values are not advisable. Additionally, due to the overly sharp response, it not only filters out the noise signal but also removes portions of the signal around the actual voice signal. This phenomenon is evident in the third graph around θ shown in Fig. 8.

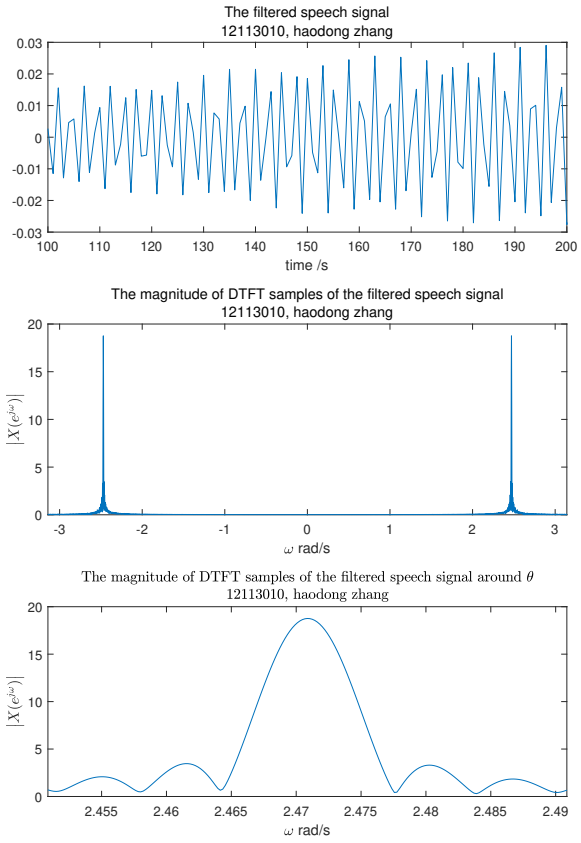


Fig. 7: the time domain samples and DTFT samples of the speech signal after filter

V. Lowpass Filter Design Parameters

There are 4 important parameters in filter design: passband edge ω_p, stopband edge ω_s, passband ripple δ_p, stopband ripple δ_s. For nspeech2, its high-frequency noise is band-limited to $|\omega| > 2.2$, and its low-frequency signal is band-limited to $|\omega| < 1.8$. The magnitude of the filter response, $H(e^{jw})$, is constrained in the passband and stopband by the following two equations:

$$|H(e^{j\omega}) - 1| \leq \delta_p \quad \text{for} \quad |\omega| \leq \omega_p$$
$$|H(e^{j\omega})| \leq \delta_s \quad \text{for} \quad \omega_s \leq |\omega| \leq \pi \tag{18}$$

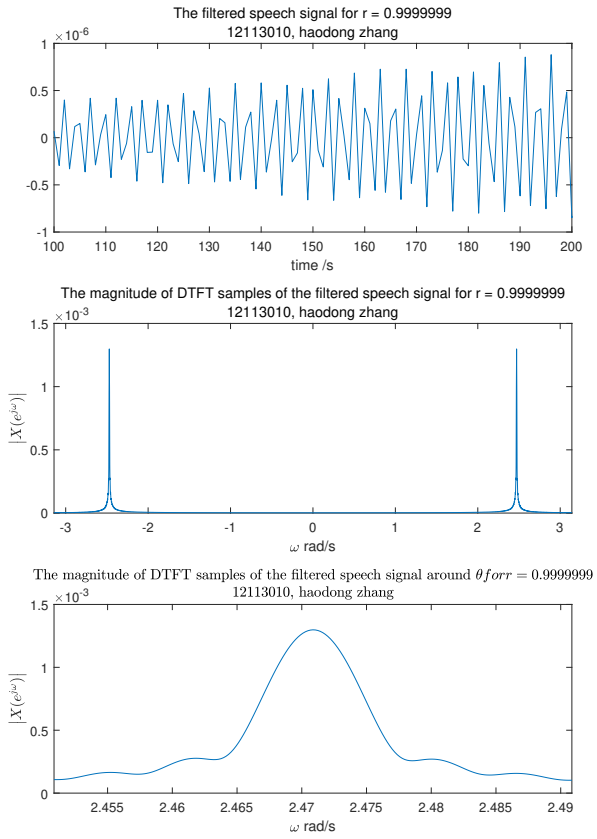


Fig. 8: the time domain samples and DTFT samples of the speech signal after filter for r=0.9999999

In nspeech2, there is a human speech signal and some high frequency buzz sound are played together. In the next section, a LPF will be designed and used to separate the speech and noise and then we will analyze the influence of the parameters above.

VI. FILTER DESIGN USING TRUNCATION

An ideal lowpass filter is specified by the frequency response:

$$H_{ideal}(e^{j\omega}) = \begin{cases} 1 & |\omega| \le \omega_c \\ 0 & \omega_c < |\omega| \le \pi \end{cases} \tag{19}$$

and corresponding impulse response:

$$h_{ideal}[n] = \frac{\omega_c}{\pi}\,\mathrm{sinc}\left(\frac{\omega_c n}{\pi}\right) \quad \text{for} \quad -\infty < n < \infty \tag{20}$$

However, the impulse response of a real-world lowpass filter can not include infinite number of entries. So, one way to mimic the ideal lowpass filter is to truncate the ideal impulse response to finite number of entries:

$$h_{\text{trunc}}[n] = \begin{cases} \frac{\omega_c}{\pi}\mathrm{sinc}\left(\frac{\omega_c}{\pi}n\right) & \text{for } n = -M, \ldots, 0, 1, \ldots, M \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

To make the filter causal, $h[n]$ needs to be shifted right to ensure $h[n] = 0$ for $n < 0$:

$$h[n] = \begin{cases} \frac{\omega_c}{\pi}\,\mathrm{sinc}\left(\frac{\omega_c}{\pi}\left(n - \frac{N-1}{2}\right)\right) & \text{for } n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

The time shift introduces a multiplying factor of $e^{-j\omega(N-1)/2}$ to the frequency response, which doesn't affect the magnitude but add a factor of $-j\omega(N-1)/2$ to the phase response. Such filter is called linear phase filter because the phase is a linear function of ω, then the signal can be passed through the filter without distortion. Equation(21) applies to both even N and odd N. When N is odd, then the value at n=(N-1)/2 is sampled at the peak of the sinc function. When N is even, then the two values at n=N/2 and $\mathrm{n} = (\mathrm{N}/2) - 1$ surround the peak.

Next we examine what impact the filter size has on the frequency response of the filter. A MATLAB function $LPFtrunc.m$is implemented to compute the truncated and shifted impulse response of size N for a lowpass filter with a cutoff frequency of ω_c=2. The function is shown below:

$LPFtrunc.m$:

```
function [h,H] = LPFtrunc(N)
    h = zeros(1,N);
    for i = 0:N-1
        h(i+1) = 2/pi*sinc(2/pi*(i-(N-1)/2));
    end
    [H,w] = DTFT(h,512);
end
```

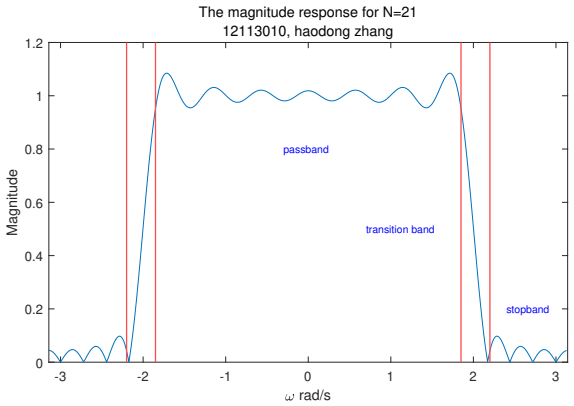
Then the resultant filters' magnitude responses are depicted in Fig. 9 and the dB expression are shown in Fig. 10.

From these figures above, we can further analyze it. What causes the prominent oscillations near the band edge of the filter in the magnitude response? The truncation process can be likened to the multiplication of $h[n]$ by a rectangular window. In the frequency domain, this multiplication is akin to convolving the DTFT (Discrete-Time Fourier Transform) of $h[n]$ (a lowpass) with the DTFT of the rectangular window (a sinc function). When the sinc, with its large constant-area sidelobes, is convolved with the desired ideal frequency response of a lowpass, the oscillations arise as these sidelobes traverse the discontinuity present in the ideal lowpass.

With an increase in the window size N, the main lobe's width decreases, making it narrower. Consequently, the smoothing effect provided by the convolution with the sinc diminishes, resulting in more frequent and abrupt ripples. Interestingly, the amplitude of the ripples does not decrease as N increases, as the abrupt edge of the rectangular window introduces a substantial amount of high-frequency components that remain unchanged with variations in N.

Both filters effectively eliminate high-frequency noise and enhance the clarity of the voice signal. However, the filter employing a broader rectangular window offers superior audio quality with a less noisy background. Therefore, we will further compare the performance of these two filters in practical case-speech signal analysis.

We use the speech signal "nspeech2.mat" to verify the performance of these two filters. Since these are FIR filters,



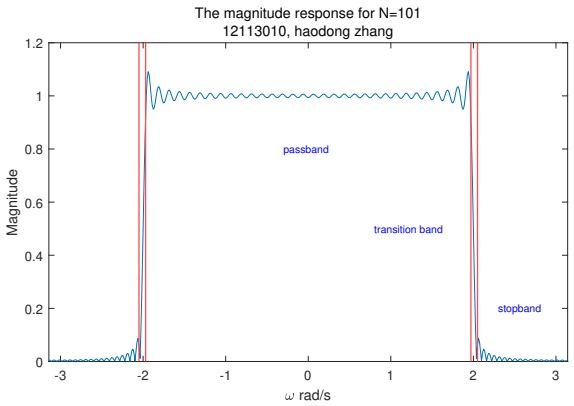
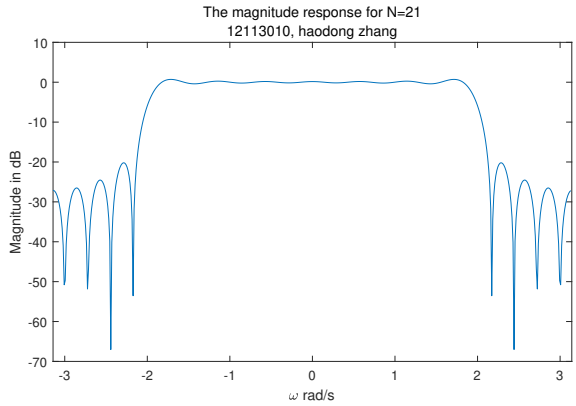


Fig. 9: the magnitude response of h[n] for different N



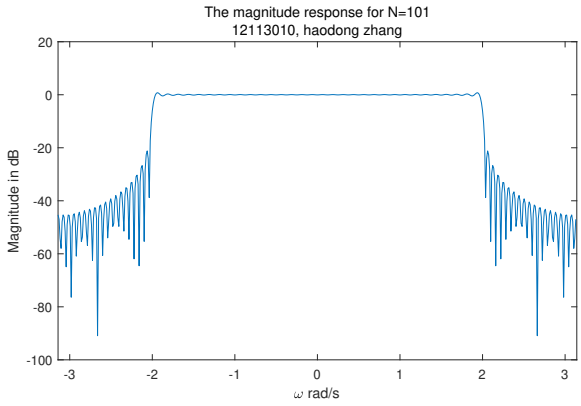


Fig. 10: the magnitude response of h[n] for different N in dB

we can simply convolve them with the audio signal. First, let's listen to the audio effects before filtering and after filtering with two different filters. It can be observed that both filters effectively suppress the noise in the speech signal, leaving only the human voice signal. Therefore, the filtering performance of both filters is quite good. However, upon closer listening, it can be noticed that the audio after filtering with the first filter still contains some noise, while the audio after filtering with the second filter is very good, with almost no noise interference.

To provide a more intuitive display and analysis, we have plotted the time-domain signals separately for both filters, as shown in Fig. 11. It can be seen that both filters effectively remove noise, but there is no significant difference in the time domain between the two filtered signals. To further examine their frequency domain effects, we have plotted their DTFT (Discrete-Time Fourier Transform) results, as shown in Fig. 12. It can be observed that in the frequency domain, the second filter allows the voice signal to pass slightly stronger than the first filter, while the noise is slightly weaker. This indicates that the passband and stopband characteristics of the second filter are better than the first, consistent with the frequency response graphs of the two filters and our real perception obtained through listening.

The MATLAB codes are as follows:

```
figure;
load nspeech2.mat;
sound(nspeech2);
subplot(3,1,1);
plot(nspeech2)
title(['The original speech signal' newline '
    12113010, haodong zhang']);
xlabel('time /s');
 signal_after_filter1  = conv(nspeech2,h1);
subplot(3,1,2);
plot( signal_after_filter1 );
title(['The speech signal after filter1 for N=21'
    newline '12113010, haodong zhang']);
xlabel('time /s');

 signal_after_filter2  = conv(nspeech2,h2);
subplot(3,1,3);
plot( signal_after_filter2 );
title(['The speech signal after filter2 for N=101'
    newline '12113010, haodong zhang']);
xlabel('time /s');
figure;
[X0,w] = DTFT(nspeech2,0);
[X1,w] = DTFT( signal_after_filter1 ,0);
[X2,w] = DTFT( signal_after_filter2 ,0);
subplot(3,1,1);
plot(w,abs(X0));
title(['The magnitude of DTFT for original speech
    signal' newline '12113010, haodong zhang']);
xlim([-pi,pi]);
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
    Interpreter','latex');
```

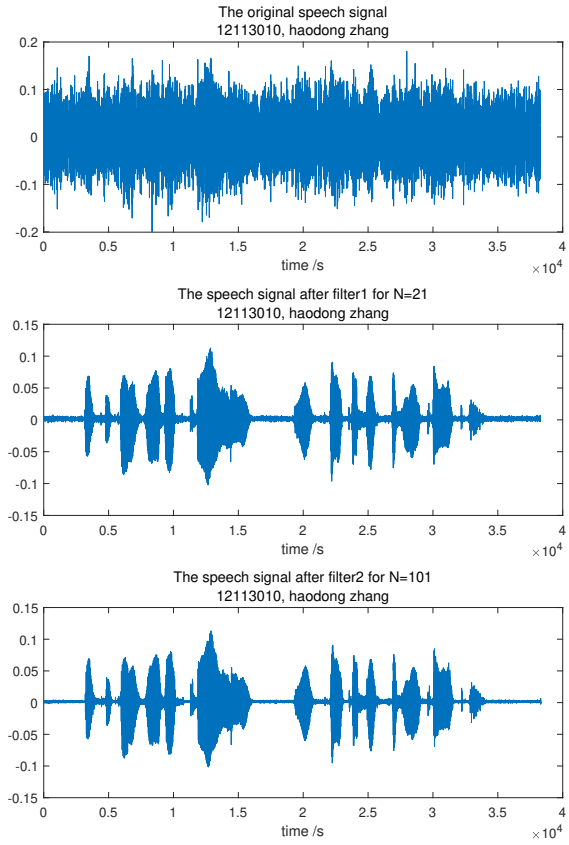


Fig. 11: the time domain of the signal before and after each filter

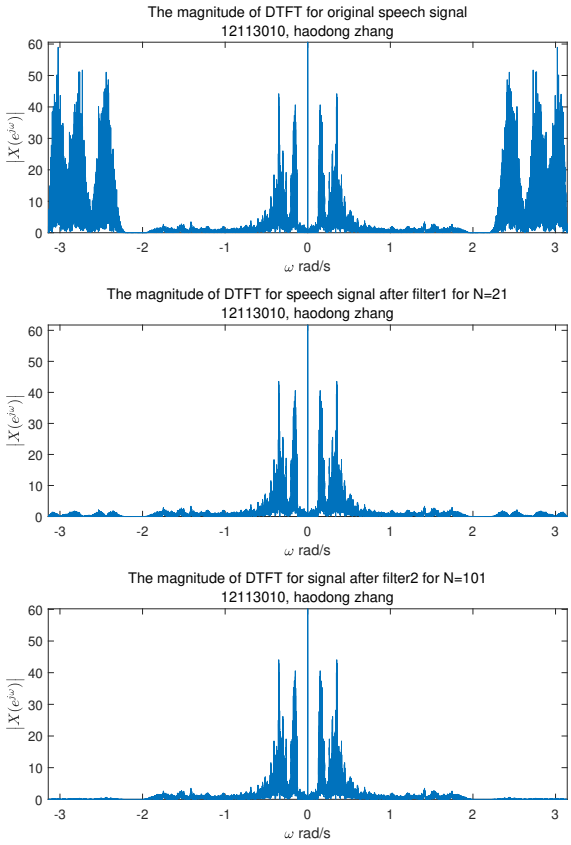


Fig. 12: the DTFT samples of the signal before and after each filter

```
subplot(3,1,2);
plot(w,abs(X1));
 title(['The magnitude of DTFT for speech signal
     after filter1 for N=21' newline '12113010,
     haodong zhang']);
xlim([-pi,pi]);
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
     Interpreter','latex');

subplot(3,1,3);
plot(w,abs(X2));
 title(['The magnitude of DTFT for signal after
     filter2 for N=101' newline '12113010, haodong
     zhang']);
xlim([-pi,pi]);
xlabel('\omega rad/s');
ylabel('$\left | X(e^{j\omega}) \right | $','
     Interpreter','latex');
```

VII. Conclusion

In this experiment, we introduced the basic concepts and knowledge of digital filters. Subsequently, we designed FIR and IIR filters, utilizing the zero points of the FIR filter to eliminate noise at specified frequencies and leveraging the poles of the IIR filter to amplify signals at specified frequencies, thereby eliminating background noise. We also explored the impact of different values of r on the IIR filter, where a larger r results in a sharper response function but also a further weakening of the magnitude. Therefore, r should not be too small or too large. We validated our analysis and conclusions by using language signals, and the theoretical predictions aligned with the experimental results.

Lastly, we investigated the parameters of a low-pass filter and analyzed the influence of the filter truncation size. It was observed that a larger truncation size led to better performance in both the passband and stopband. This conclusion was further validated through analysis using speech signals.

Acknowledgments

Thanks to the teachers and teaching assistants for their guidance. Thank the teacher for teaching us theoretical knowledge and providing the lab manual for us, which raised many meaningful questions for us and guided and inspired us to think deeply.