
# Laboratory Report of Digital Signal Processing

Lab3.Digital Filter

Name:Lu Junyi

No:517261910033

Date:2021.5.6

Score:_____

**SHANGHAI JIAOTONG UNIVERSITY**
**SJTU-ParisTech Elite Institute of Technology**

# Contents

# 1 Introduction

The main objectives of this assigment are:

- FIR filter design with window method
- IIR filter design with bilinear transform
- Filter design with poles and zeros
- Realization of filer with SOS structure

# 2 Exercise

## 2.1 Design an FIR filter with the Window method

A simple way to design FIR filter is to use window function truncate the impulse response function of an ideal filter, named the window method.

### 2.1.1 Question1

Design the FIR filters with the rectangle window, where the $\omega_c = 0.2\pi$ and the window length $N = -30 \sim 30$ and $-100 \sim 100$, respectively. Calculate the corresponding FRFs(Magnitudes and Phase) of the FIR filters. And then compare their impulse response functions and FRFs in terms of different window length.

We want to use matlab's function **fir1** to construct the FIR filter, but fir1 requires a normalized cutoff frequency or cutoff angle frequency as the second input, which takes values in the range [0,1]. The normalization method is to divide the cutoff frequency by half of the sampling frequency, which is very different from the usual normalization.

Here the meaning of $\omega_c$ is very unclear, but after carefully studying the subsequent question, we can find that the whole problem wants us to design a low-pass filter with an analog cut-off frequency $f_c = 20Hz$ to filter out the sine wave with a frequency of 23Hz, so after calculation we can find that $\omega_c = 0.2 * \pi$ is actually the normalized angular frequency multiplied by $\pi$, so the second parameter input to the **fir1** function should be 0.2

So in order to facilitate the later questions, I chose the sampling frequency $f_s = 200Hz$, that is, the analog cutoff radian frequency is $\Omega_c = \omega_c * f_s = 0.2\pi * 200 = 40\pi\ rad/s$, and then the horizontal coordinates of all frequency domain images in this problem are analog frequencies, the analogue cutoff frequency $f_c = \frac{\Omega_c}{2\pi} = 20Hz$

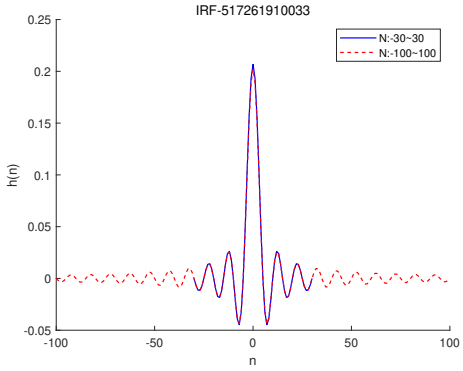


**Fig 1.** IRF of FIR filter with different length of window

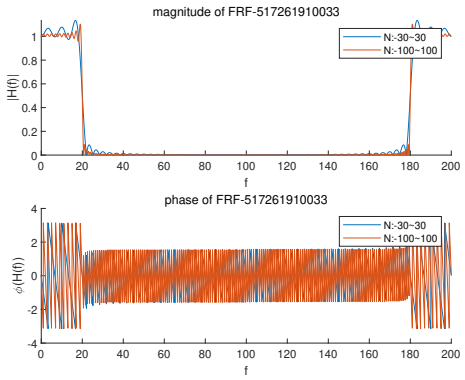


**Fig 2.** FRF of FIR filter with different length of window

Notice that the performance of the FIR filter becomes better as the length of the window increases.

### 2.1.2 Question2

Design causal FIR filters, the $\omega_c = 0.2\pi$, with the rectangle window and Kaiser window with the window length $N = 0 \sim 59$. And then compare their impulse response functions and FRFs in terms of the window type.

'causal' means that the time must not be negative

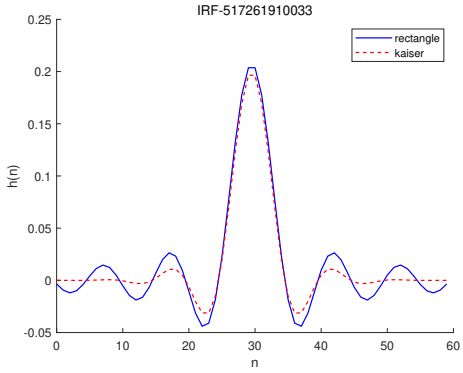


**Fig 3.** IRF of FIR filter with different type of window

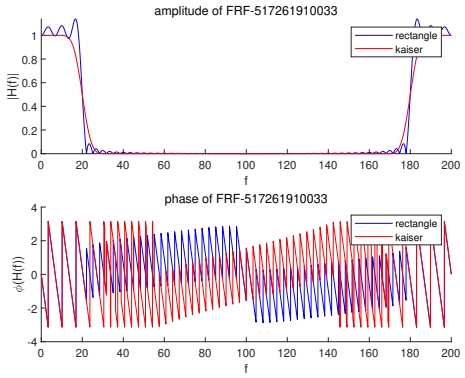


**Fig 4.** FRF of FIR filter with different type of window

Notice that the performance of the FIR filter with kaiser window is better than rectangular window.

### 2.1.3 Question3

The signal $x(t)$ is composed of two cosines at the frequencies 8,23 with the amplitudes 4 and 8 , respectively. Set a sampling frequency $f_s = 200$, and plot the corresponding discrete sequence $x[n]$ in $n = 0 : 199$. Calculate the spectrum of $x[n]$.

Here is the discrete sequence and spectrum of x[n]:

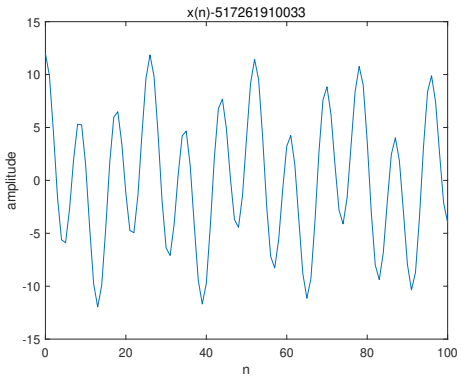


**Fig 5.** x(n) in time domain

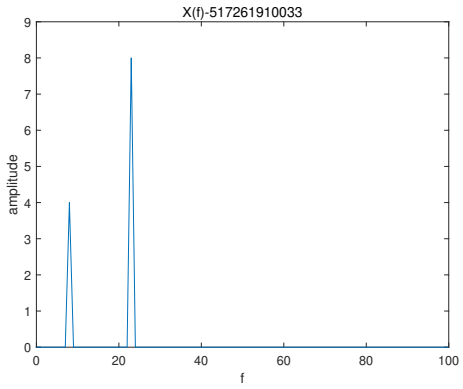


**Fig 6.** X(f) in frequency domain

### 2.1.4 Question4

In order to filter out the cosine of 23 and keep the cosine of 8 of $x[n]$ in (1.1), we need to design a digital low-pass filter with an FIR filter. The filter's overshoot should be less than 5%. Show FRF of the FIR filter.

I use filter with kaiser window designed in Question2 to filter out the cosine of 23, the FRF of this filter is present below:

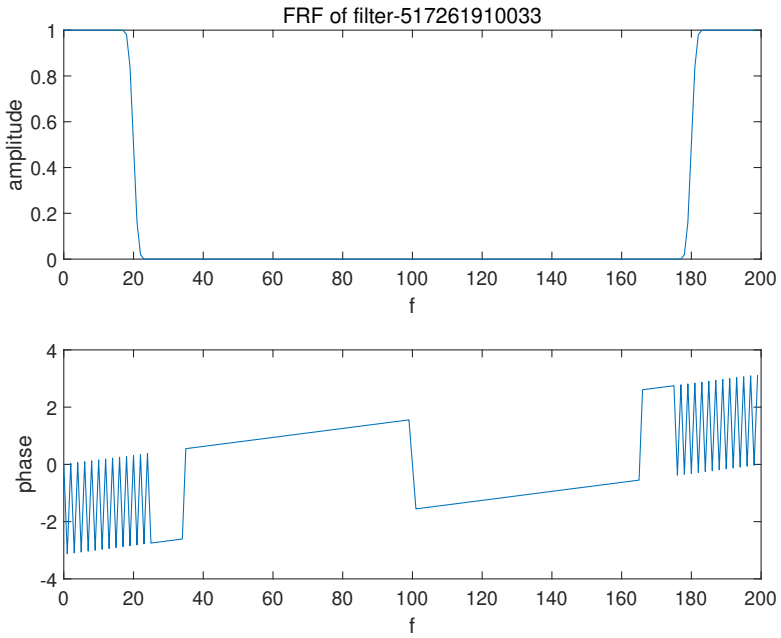


**Fig 7**

Noticed that the overshoot is less than 5

The time sequence and spectrum of the output signal are present below:

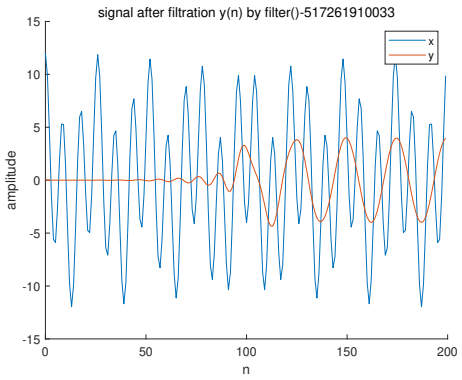


**Fig 8.** y(n) in time domain

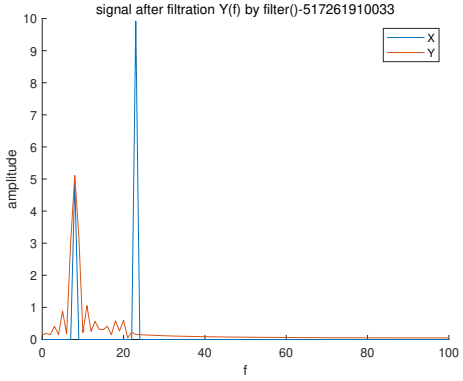


**Fig 9.** Y(f) in frequency domain

### 2.1.5 Question5

Develop a sub-function $y = my\ \text{filter}\ (bz, az, x)$ by using the I/O difference equation method.

The algoritme is presented below:

```
function y = myfilter(bz,az,x)
    M=length(bz);
    N=length(az);
    b=bz(end:-1:1);
    a=az(end:-1:2);
    y_n=zeros(1,length(x)+N-1);
    x_n=[zeros(1,M-1),x];
    id_x=1:M;

    if N>=2
        id_y=1:N-1;
        for ii=N:length(y_n)
            y_n(ii)=1/az(1)*(sum(b.*x_n(id_x))-sum(a.*y_n(id_y)));
            id_x=id_x+1;
            id_y=id_y+1;
        end
    else
        for ii=1:length(y_n)
            y_n(ii)=1/az(1)*(sum(b.*x_n(id_x)));
            id_x=id_x+1;
        end
    end
    y=y_n(N:end);
end
```

### 2.1.6 Question6

Filter $x$ with the filter in (5) and produce the output signal $y_1$. Show the $x$ and $y_1$.

The result is same with Question4.

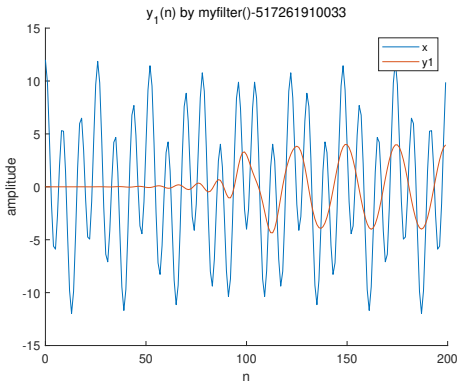


**Fig 10.** $y_1(n)$ in time domain

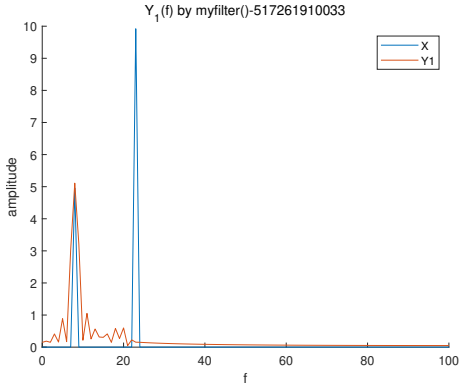


**Fig 11.** $Y_1(f)$ in frequency domain

## 2.2 Comparison of the analog filter and digital filter

Analog filters, e.g. Butterworth, Chebyshev, etc., have corresponding digital filters which can be designed by so called Bilinear transformation. And these analog filters are considered as the prototype filters of their digital ones. The prototype analog filters and their digital filters are similar but not exactly the same in their frequency response functions. Develop a 2-order analog Butterworth low-pass filter, with the cutoff frequency at $f_c$ using

$$H(s) = \frac{1}{\frac{1}{\omega_c^2}s^2 + \frac{\sqrt{2}}{\omega_c}s + 1}$$

### 2.2.1 Question1

Show as and bs as the denominator and numerator coefficients of the transfer function of this analog filter.

Noticed that we can directly get these coefficient from $H(s)$ given:

$$bs = [1]; \quad as = [1/(w_c^2), \sqrt{(2)}/w_c, 1] \tag{1}$$

### 2.2.2 Question2

Set the sampling frequency $f_s$, and use bilinear transform to convert the analog prototype filter into the corresponding digital filter. Show az and bz as the denominator and numerator coefficients of transfer function of the digital filter. Show the detail of how you calculate the result (hand calculate and not use any function)

I transfer the function Laplace into function Z by changing the variable and finally I represent all the coefficient of az and bz in function of $T$ and $w_c$, where $T$ is the interval of sampling

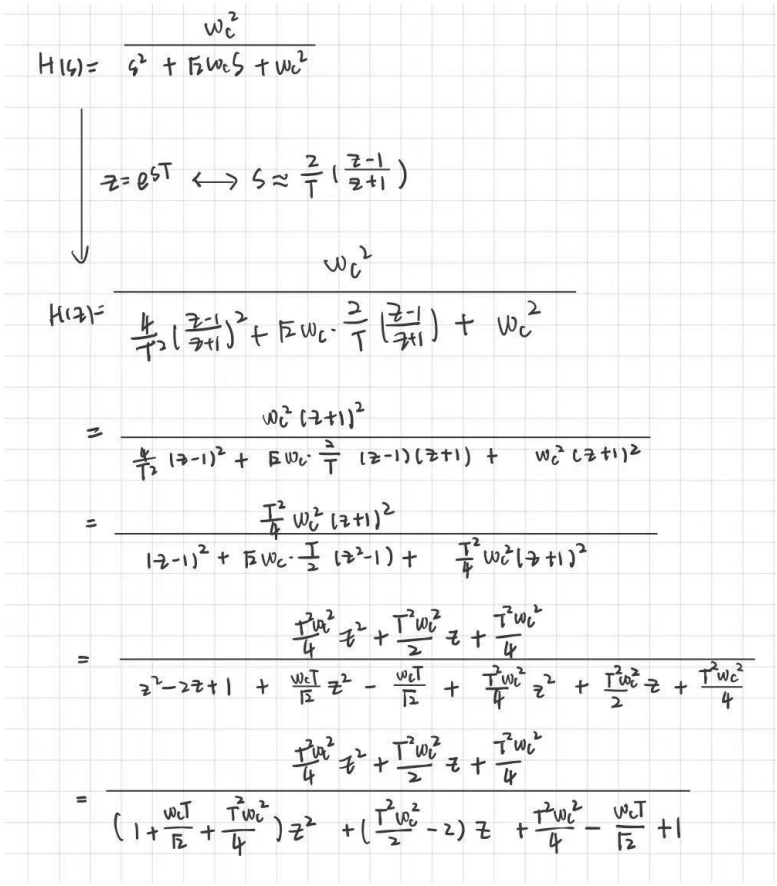

**Fig 12.** The process of my calculation

### 2.2.3 Question3

Set $f_c = 3$ and $f_s = 10$, compare the frequency respond functions (amplitudes and phases) of these two filters with appropriate axes.

I present the FRF of these two filter below:

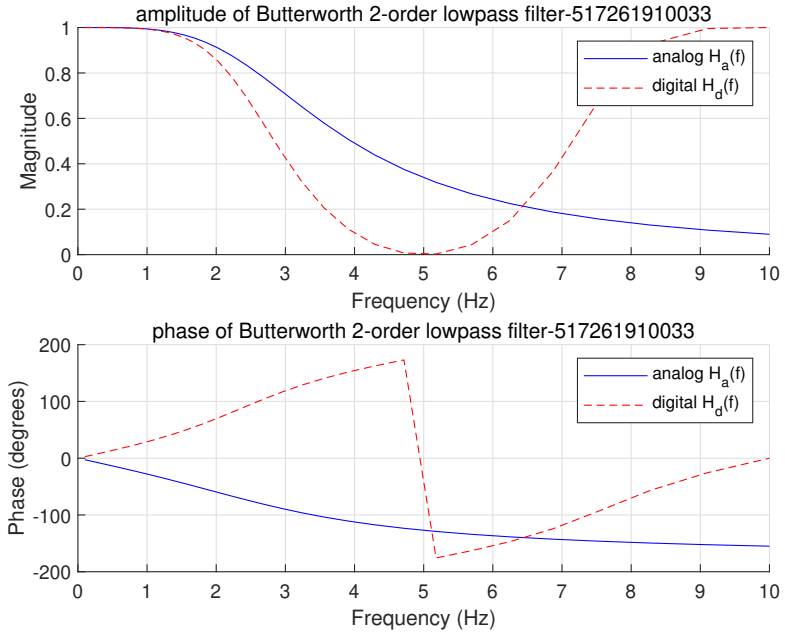


**Fig 13.** Comparison of IIR filter and FIR filter

## 2.3 Design an IIR filter with bilinear transform and realize filtering

### 2.3.1 Question1

The $x[n]$ is the same as in (1.3). In order to filter out the cosine of 23 and keep the cosine of 8 , we need to design a digital low-pass filter with the prototype filter of Butterworth by using bilinear transform, where the $f_c = 15$ and the normalized cutoff frequency $f_{cn} = \frac{f_c}{f_s}$.
Show az and bz of the filter and plot the frequency response function (FRF) of the filter in the normalized frequencies.

I developed a 9-order digital Butterworth Low pass filter, with the cutoff frequency $f_c = 15Hz$ using the function butter,I show az and bz as the denominator and numerator coefficients of the transfer function of the digital filter in the following:

```
命令行窗口

b1 =

   1.0e-04 *

    0.0067    0.0604    0.2415    0.5636    0.8453    0.8453    0.5636    0.2415    0.0604    0.0067

a1 =

    1.0000   -6.2882   17.8798  -30.1025   33.0137  -24.4253   12.1775   -3.9414    0.7509   -0.0641
```

**Fig 14.** The coefficient of the filter(note az as a1 and bz as b1)

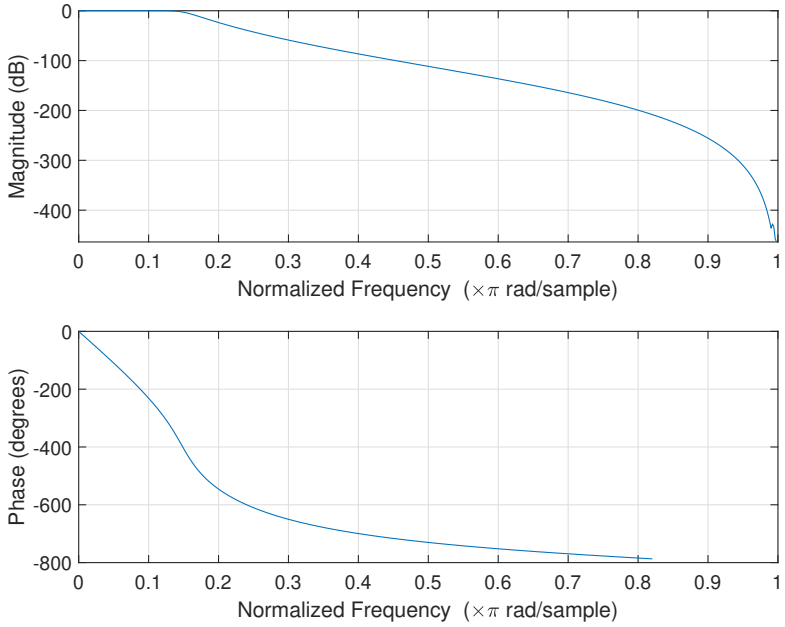


**Fig 15.** FRF of filter

### 2.3.2 Question2

Use myfilter in (1.5) to filter $x$ in (3.1), and produce the output signal $y_2$. Compare the time sequences and frequency spectra (amplitudes and phases) of $x$ and $y_1$(1.6) and $y_2$, respectively.

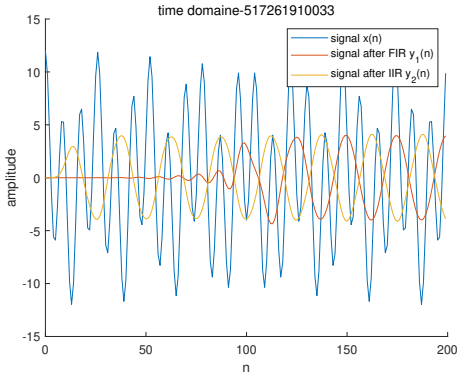


**Fig 16.** Comparison of FIR and IIR in time domain

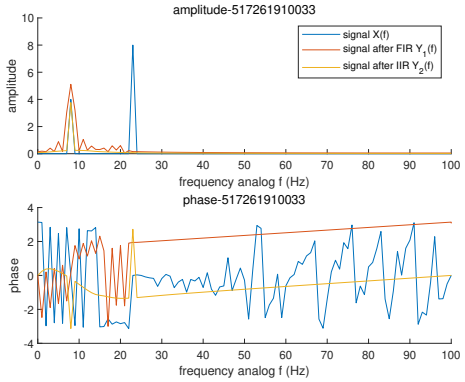


**Fig 17.** Comparison of FIR and IIR in frequency domain

Compared with the results of the IIR filter, we find a very significant delay in the signal after the FIR filter.

## 2.4 Pole/Zero Designs

Pole/zero placement can be used to design simple filters, such as the notch and comb filters.

### 2.4.1 Question 1

In order to filter out the cosine of 23 and keep the cosine of 8 of $x[n]$ in 1(3) (but change n = 0 : 1999), we need to design a notch filter with a bandwidth of 1 Hz (Try-and-error way)(Have 3 dB change in 1 Hz band around 23 Hz ). Show $H_1(z)$, and plot a map of poles and zeroes with the unit circle for reference. Then show FRFs of the notch filter.

The transfer function of notch filter is:

$$H_1(z) = \frac{(z - 1 * e^{jw_{c1}})(z - 1 * e^{-jw_{c1}})}{(z - (1 - \delta) * e^{jw_{c1}})(z - (1 - \delta) * e^{-jw_{c1}})} \tag{2}$$

Here the cutoff frequence $f_{c1} = 23Hz$

Notice that as the value of $\delta$ increases, the sine wave at 23Hz will be filtered more thoroughly, but the sine wave at 8Hz will be more affected, i.e. its shape will change more, so we have to make a compromise here, finally I choose $\delta$=0.03

I plot the map of zero/pole of this notch filter and also the FRF:

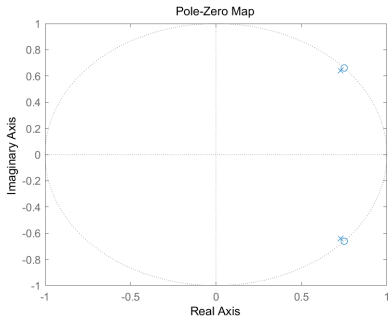


**Fig 18.** map of zero/pole of notch filter

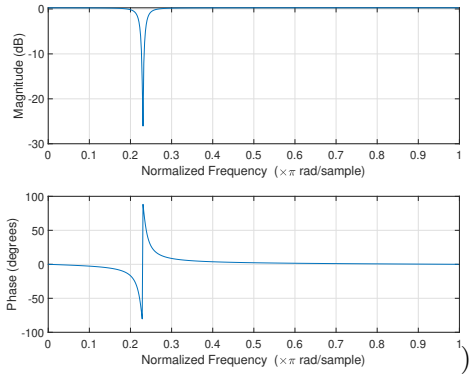


**Fig 19.** FRF of notch filter

As we can see from Fig18, the filter is stable because the pole is inside the unit circle

### 2.4.2 Question 2

Filter $x$ with the notch filter and produce the output signal $y_3$. Compare the time sequences and frequency spectra of $y_2$ and $y_3$, respectively.

I plot $y_3$ in time/frequency domain andcompare it with $y_2$ from Exe3:

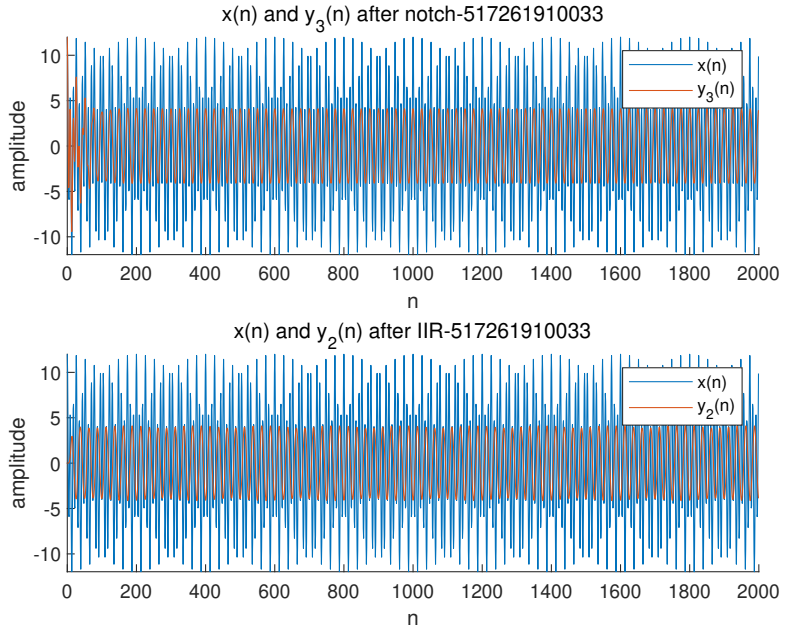


**Fig 20.** Compare of $y_2$ and $y_3$ in time domain

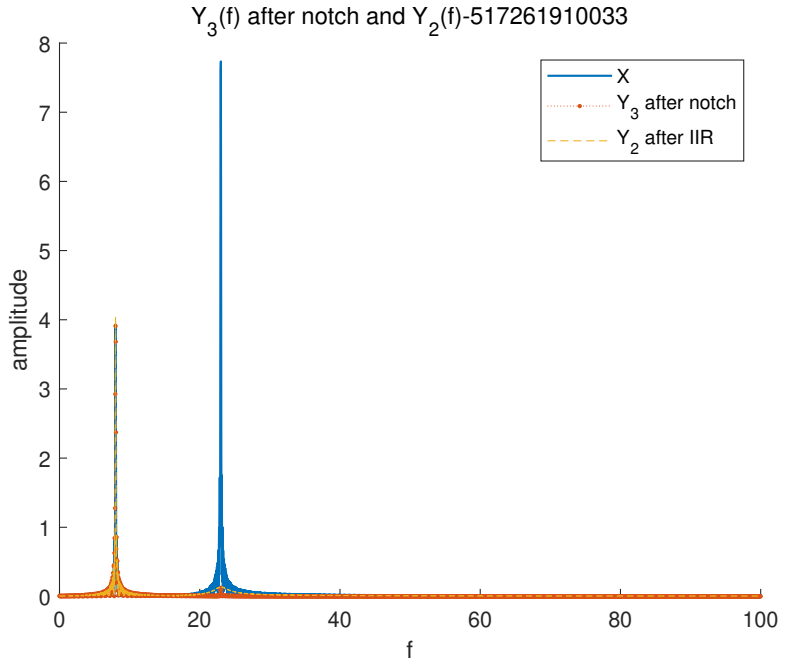


**Fig 21.** Compare of $Y_2$ and $Y_3$ in frequency domain

I will explain the difference between the two filter in question5

### 2.4.3 Question 3

In order to highlight the cosine of 8 of $x[n]$ in 1(3), we can alternatively design a comb filter to highlight 8 of $x10$ times with a bandwidth of 1 Hz. Show $H_2(z)$ and plot a map of poles and zeroes with the unit circle. Then showFRFs of the comb filter.

The transfer function of comb filter is:

$$H_1(z) = \frac{(z-(1-11\delta_2)*e^{jw_{c2}})(z-(1-11\delta_2)*e^{-jw_{c2}})}{(z-(1-\delta_2)*e^{jw_{c2}})(z-(1-\delta_2)*e^{-jw_{c2}})} \tag{3}$$

Here the cutoff frequence $f_{c2} = 8Hz$

I do the same as in Question 1 and finally I choose $\delta_2 = 0.01$ which I think can do the best compromise. I plot the map of zero/pole of comb filter and also the FRF:

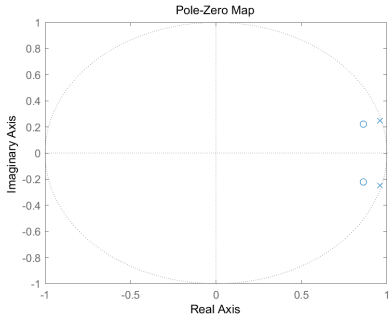


**Fig 22.** map of zero/pole of comb filter

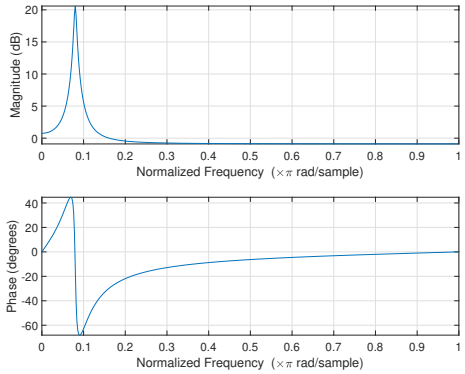


**Fig 23.** FRF of comb filter

As we can see from Fig22, the filter is stable because the pole is inside the unit circle

### 2.4.4 Question 4

Filter $x$ with the comb filter and obtain the filtered signal$y_4$. Compare the time sequences and frequency spectra of $y_3$, and $y_4$, respectively.

I plot $y_4$ in time/frequency domain and compare it with $y_3$ from Question2:

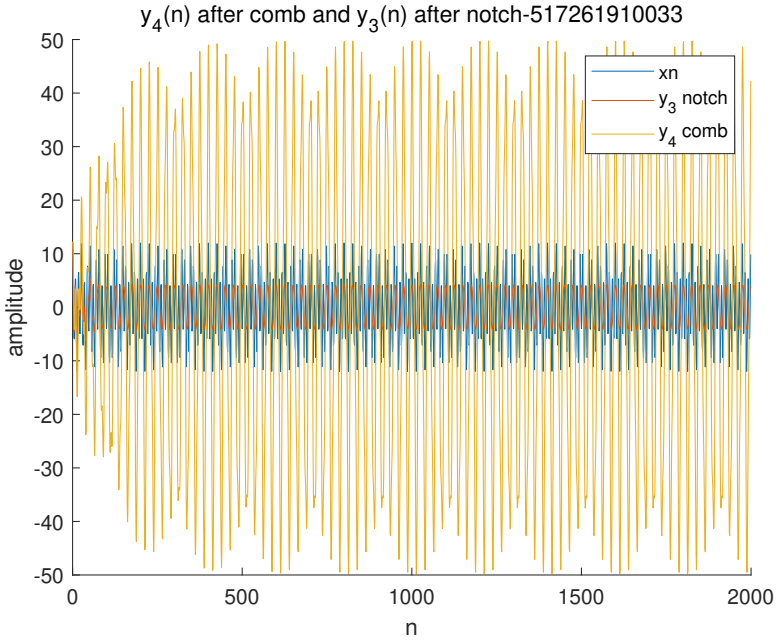


**Fig 24.** Comparison of $y_3$ and $y_4$ in time domain

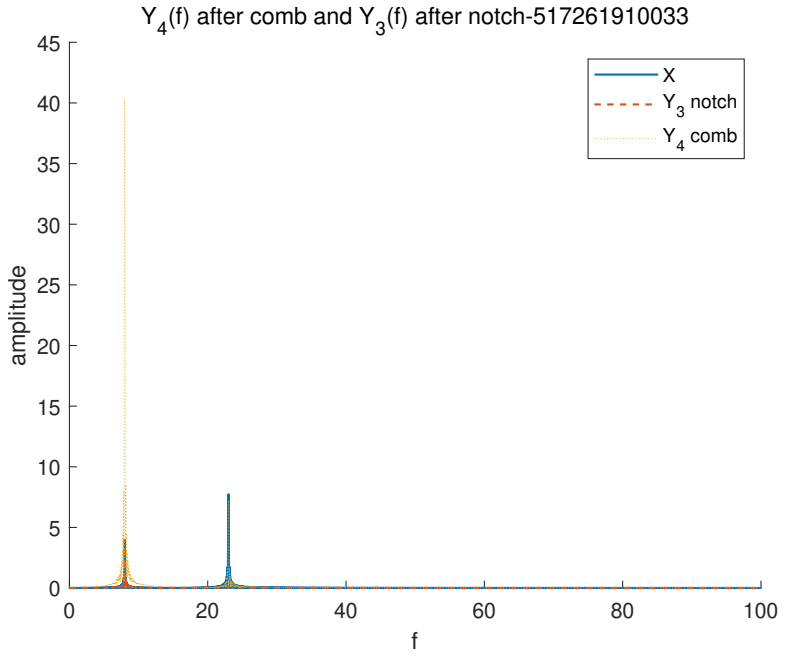


**Fig 25.** Comparison of $y_3$ and $y_4$ in frequency domain

### 2.4.5 Question 5

Give a discussion of advantages and disadvantages of all these filters according to the results in 4(1) and 4(3).

- IIR filter: The advantage is that the computational cost is very low. IIR filters are usually of low order, so that we can design simple filters to satisfy our basic needs. However,the disadvantage is that IIR filter is not always stable. As the order increases, the filter could become unstable because the quantization of the coefficients may push the poles outside the unit circle. Besides, the linear phase cannot be achieved exactly over the entire Nyquist interval, although we can make an approximation over the relevant passband of the filter.
- FIR filters: The advantage is the stability of filtering as the order increases, for there is no poles in FIR filters because it doesn't relay on the output before. Besides, the phase property of FIR filters will be linear with well-chosen window type, which will eliminate the group delay and so the distortion of signal. However, FIR filters are often in high order, which will cause a huge demand for computational resources. What's more, to enhance the filtering effect of FIR filters, we need to considerably enlarge the time span, which will cause multiple times waste of computational resources.
- Pole/Zero designed filters: The advantage is that this kind of Pole/Zero designed filters are able to filter out or to keep the exact frequencies we want, which is a great challenge for previous filters. However, as is noticed in this exercise, to enhance the filtering effect of FIR filters, we need to considerably enlarge the time span(from N=200 to N=2000), which will cause multiple times waste of computational resources.What's more, this kind of filter will add more distortion to the part of signal we want to reserve than FIR and IIR filter

## 2.5 Application

### 2.5.1 Question 1

Read the audio files: birds.wav, Bill Gates.wav, Melinda Gates.wav and mixGates.wav, and play the later 3 files(For matlab, we recommond function audioread () and sound(). And show these three signals in time domain and their frequency spectra.

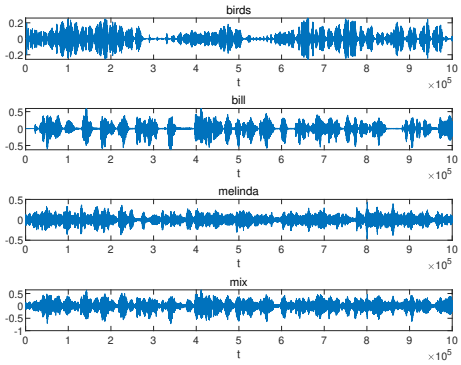


**Fig 26.** time sequence

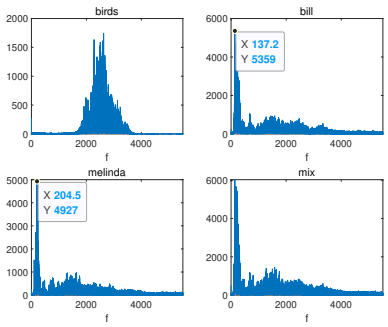


**Fig 27.** spectrum frequency

### 2.5.2 Question 2

According above information, design an IIR filter (for Matlab you can use Toolbox - filterDesign). The filter can keep the majority of Bill's speech in mixGates.wav, but eliminate the sound of Melinda's. Find out in which frequency band that contains most language information of Bill and Melinda.

After observing the frequency spectrum of Bill and Melinda's voice, we found that the information of Melinda's voice is mainly contained in the frequency component around 200Hz, while the main information of Bill's voice is contained in the frequency band around 130Hz.
So I design a Low pass filter with filterDesigner and the parameter is :

$$Fs = 44100Hz, type = butterworth, Fpass = 176.4Hz, Fstop = 250Hz, Apass = 1dB, Astop = 3dB$$

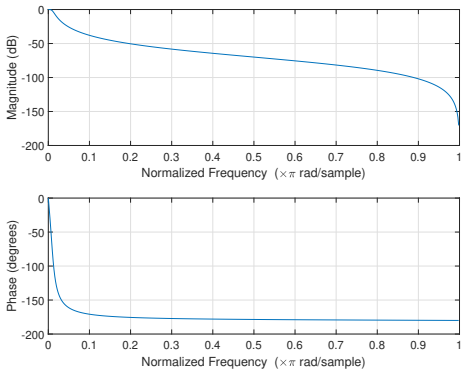


**Fig 28.** FRF of Low pass filter to filter out melinda's speak

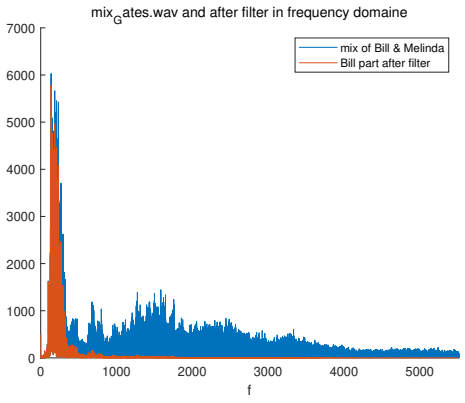


**Fig 29.** mixGates.wav after filter in frequency domain

### 2.5.3 Question 3

Mix the sound of human voice(Bill or Melinda) and bird voice(birds.wav), analysis the spectrum of them and design a filter to seperate them.

I mix the BillGates'voice and bird's call in one and plot the signal in frequency domain:

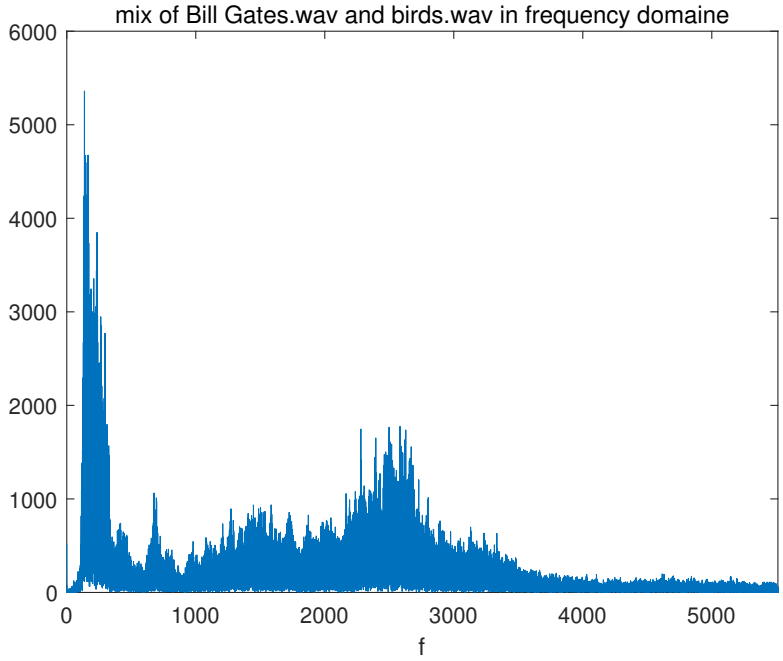


**Fig 30.** mix of Bill Gates.wav and birds.wav in frequency domain

For the first, I design a Low pass filter to filter out bird's call, and the parameter in filterDesigner is:

$$Fs = 44100Hz, type = butterworth, Fpass = 1000Hz, Fstop = 1650Hz, Apass = 1dB, Astop = 25dB$$

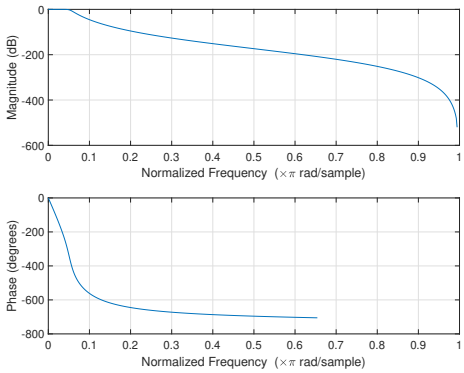


**Fig 31.** FRF of Low pass filter to filter out Bird's call

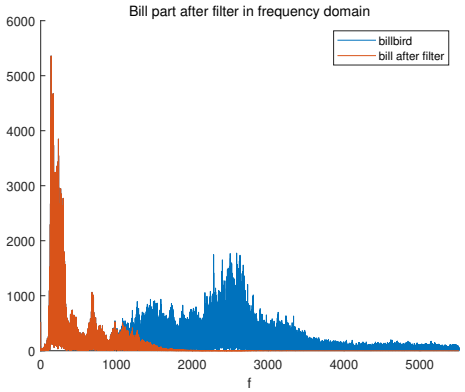


**Fig 32.** Comparison of audio original and audio after Low pass filter

Then I design a High pass filter to filter out BillGate's voice, and the parameter in filterDesigner is:

$$Fs = 44100Hz, type = butterworth, Fstop = 1000Hz, Fpass = 2000Hz, Apass = 1dB, Astop = 30dB$$

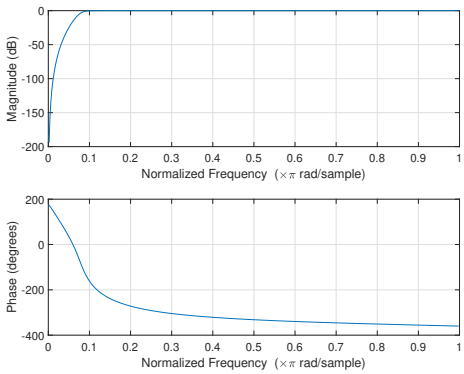


**Fig 33.** FRF of High pass filter to filter out Bill Gate's voice

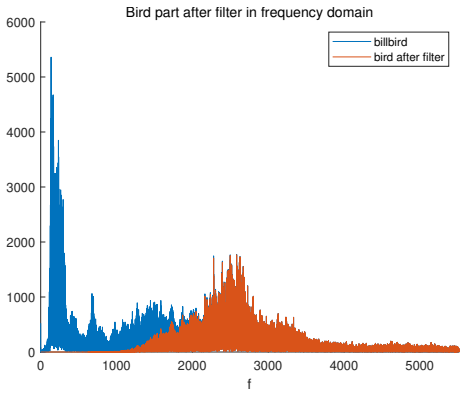


**Fig 34.** Comparision of audio original and audio after High pass filter

After listening to the audio filtered by the low-pass and high-pass filters, we found that Bill Gates' voice could not be eliminated as well as the bird's call. This is because Bill Gates' voice has a lot of high intensity frequency components in the 1600Hz-3000Hz band, which is the main frequency band of the bird's call, and this causes great interference to the bird's call. The bird call audio does not have high frequency components in the main band of the Birgit sound. Therefore, filtering out the bilge sound is not effective, and filtering out the bird call is very effective

## 2.6 SOS filter

Design a function $y = \text{mySOS filter}\ (SOS, G, x)$ to filter a signal by the SOS matrix, the SOS can be stored like the matrix (can use MATLAB help Biquad Filter) :

- Do the same thing in (3.2) but use the function mySOSfilter, compare the result of the two filter.

Noticed that the gain $G$ which returned by filterDesigner in Matlab is not a number but a vector, which represent the gain of each section but not the gain total! So we need to add one command in the second for loop :

```
temp_x=temp_x*G(  j  j);
```

And the total algoritme is present below:

```
function  y= mySOSfilter(SOS,G,x)

y = zeros(1,length(x));
w = zeros(length(x)+2, length(SOS(:,1)));
for  ii = 1:length(x)
     temp_x = x(ii);
     for  jj = 1:length(SOS(:,1))
          w(ii+2, jj) = temp_x - SOS(jj,5)*w(ii+1, jj) - SOS(jj,6)*w(ii , jj);
          temp_x = SOS(jj,1)*w(ii+2, jj) + SOS(jj,2)*w(ii+1, jj) + SOS(jj,3)*w
              (ii , jj);
          temp_x=temp_x*G(jj);
```

```
11        end
12        y(ii) = temp_x;
13    end
14
15    end
```

I design a 9-order Low-pass filter using the same parameter in Exe3.Ques2 with filterDesigner and export the parameter (SOS,G), then I use the function mySOSfilter to filtrate the signal and get the output signal $y_2'(n)$ and its FT $Y_2'(f)$

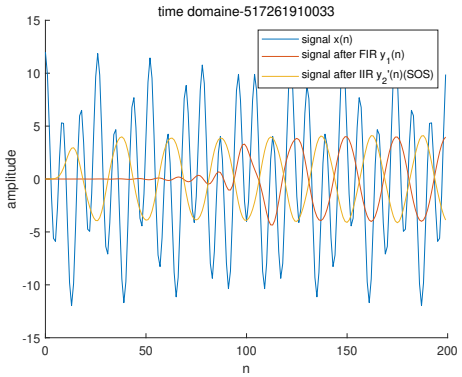


**Fig 35.** time domain

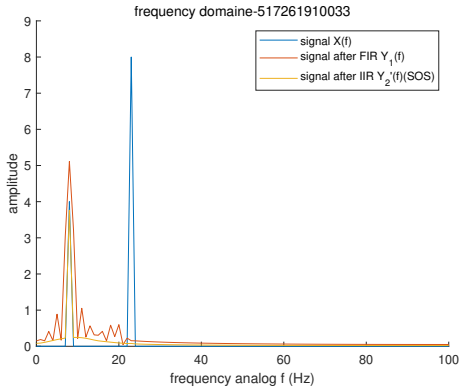


**Fig 36.** frequency domain

In compare with Fig16 and Fig17,we can find that the $y_2'$ is almost the same with $y_2$, so the SOS filter has been proved to be a correct realization of IIR filter

# 3 Summary

In this assignment, we are familiar with several method to design digital filters, including designing IIR filter with bilinear transform, designing FIR filter with windowing methos, designing filter with poles and zeros.And know how to use filterDesigner to designer SOS filter for practical applications

# A Appendix

## A.1 Ex.1

Q1

```
clc;clear;close all;
%% init
fs=200;
Ts=1/fs;
df=0.01;
f=0:df:fs-df;

wc=0.2*pi;
wcn=wc/pi;
fc=wc*fs/(2*pi);
%% window
N1=-30:30;
N2=-100:100;
L1=length(N1);
L2=length(N2);
window1=rectwin(L1);
window2=rectwin(L2);

%% Impulse response function
b1=fir1(L1-1,wcn,'low',window1);
b2=fir1(L2-1,wcn,'low',window2);

figure
hold on
plot(N1,b1,'b','linewidth',1);
plot(N2,b2,'r--','linewidth',1);
hold off
title('IRF-517261910033')
legend('N:-30~30','N:-100~100')
xlabel('n')
ylabel('h(n)')
saveas(gcf,'image\fig3_1_1(IRF)','epsc')
%% Frequency response function
Hz1=freqz(b1,1,f,fs);
Hz1_db=20*log10(abs(Hz1));
Hz2=freqz(b2,1,f,fs);
Hz2_db=20*log10(abs(Hz2));

```

```
figure
subplot(211)
hold on
plot(f,abs(Hz1));
plot(f,abs(Hz2));
hold off
title('magnitude of FRF-517261910033')
legend('N:-30~30','N:-100~100')
xlabel('f')
ylabel('|H(f)|')
subplot(212)
hold on
plot(f,angle(Hz1));
plot(f,angle(Hz2));
hold off
title('phase of FRF-517261910033')
legend('N:-30~30','N:-100~100')
xlabel('f')
ylabel('\phi(H(f))')
saveas(gcf,'image\fig3_1_1(FRF)','epsc')
```

Q2

```
clc;clear;close all;
%% init


fs=200;
Ts=1/fs;
df=0.01;
f=0:df:fs-df;

wc=0.2*pi;
wcn=wc/pi;
fc=wc*fs/(2*pi);
%% window
N=0:59;
L=length(N);
windowr=rectwin(L);
windowk=kaiser(L,10);

%% Impulse response function
br=fir1(L-1,wcn,'low',windowr);
bk=fir1(L-1,wcn,'low',windowk);
```

```
22
23 figure
24 hold on
25 plot(N,br,'b','linewidth',1);
26 plot(N,bk,'r--','linewidth',1);
27 hold off
28 title('IRF-517261910033')
29 legend('rectangle','kaiser')
30 xlabel('n')
31 ylabel('h(n)')
32 saveas(gcf,'image\fig3_1_2(IRF)','epsc')
33 %% Frequency response function
34 Hzr=freqz(br,1,f,fs);
35 Hzr_db=20*log10(abs(Hzr));
36 Hzk=freqz(bk,1,f,fs);
37 Hzk_db=20*log10(abs(Hzk));
38
39 figure
40 subplot(211)
41 hold on
42 plot(f,abs(Hzr),'b');
43 plot(f,abs(Hzk),'r');
44 hold off
45 title('amplitude of FRF-517261910033')
46 legend('rectangle','kaiser')
47 xlabel('f')
48 ylabel('|H(f)|')
49 subplot(212)
50 hold on
51 plot(f,angle(Hzr),'b');
52 plot(f,angle(Hzk),'r');
53 hold off
54 title('phase of FRF-517261910033')
55 legend('rectangle','kaiser')
56 xlabel('f')
57 ylabel('\phi(H(f))')
58 saveas(gcf,'image\fig3_1_2(FRF)','epsc')
```

Q3

```
1 clc;clear;close all;
2 %% init
3 fs=200;
4 Ts=1/fs;
```

```
df=1;
f=0:df:fs-df;

n=0:199;
Lwin=length(n);
%% paramter
f1=8;
f2=23;
A1=4;
A2=8;

xn=A1*cos(2*pi*f1/fs*n)+A2*cos(2*pi*f2/fs*n);
X_f=fft(xn,length(f));

figure
plot(n,xn)
title('x(n)-517261910033')
xlabel('n')
ylabel('amplitude')
xlim([0,fs/2])
saveas(gcf,'image\fig3_1_3(time)','epsc')
figure
plot(f,2*abs(X_f)/Lwin)
xlabel('f')
ylabel('amplitude')
title('X(f)-517261910033')
xlim([0,fs/2])
saveas(gcf,'image\fig3_1_3(freq)','epsc')
```

Q4

```
clc;clear;close all;
%% init
fs=200;
Ts=1/fs;
df=1;
f=0:df:fs-df;

n=0:199;
Lwin=length(n);


%% signal
f1=8;
```

```
f2=23;
A1=4;
A2=8;
xn=A1*cos(2*pi*f1/fs*n)+A2*cos(2*pi*f2/fs*n);

%% filter
wc=0.2*pi;
wcn=wc/pi;
fc=wc*fs/(2*pi);
windowk=kaiser(Lwin,10);
bk=fir1(Lwin-1,wcn,'low',windowk);
%% Frequency response function
Hzk=freqz(bk,1,f,fs);
Hzk_db=20*log10(abs(Hzk));

figure
subplot(211)
plot(f,abs(Hzk));
title('FRF of filter-517261910033')
xlabel('f')
ylabel('amplitude')
subplot(212)
plot(f,angle(Hzk));
xlabel('f')
ylabel('phase')
saveas(gcf,'image\fig3_1_4(FRF)','epsc')
%% filtrate by filter() of matlab
y=filter(bk,1,xn);
Y_f=fft(y,length(f));

figure
plot(n,y)
title('signal after filtration y(n) by filter()-517261910033');
xlabel('n');
ylabel('amplitude')
saveas(gcf,'fig3_1_4(time)','epsc')

figure
plot(f,2*abs(Y_f)*2.48/Lwin)
title('signal after filtration Y(f) by filter()-517261910033');
xlabel('f');
ylabel('amplitude')
xlim([0,fs/2])
```

```
57 saveas(gcf,'image\fig3_1_4(freq)','epsc')
```

Q6

```
1  clc;clear;close all;
2  %% init
3  fs=200;
4  Ts=1/fs;
5  df=1;
6  f=0:df:fs-df;
7
8  n=0:199;
9  Lwin=length(n);
10
11
12 %% signal
13 f1=8;
14 f2=23;
15 A1=4;
16 A2=8;
17 xn=A1*cos(2*pi*f1/fs*n)+A2*cos(2*pi*f2/fs*n);
18
19 %% filter()
20 wc=0.2*pi;
21 wcn=wc/pi;
22 fc=wc*fs/(2*pi);
23 windowk=kaiser(Lwin,10);
24 bk=fir1(Lwin-1,wcn,'low',windowk);
25
26 %% filtrate by myfilter()
27 y1=myfilter(bk,1,xn);
28 Y1_f=fft(y1,length(f));
29 save('data\y1_FIR.mat','y1','Y1_f');
30 figure
31 plot(n,y1)
32 title('y_1(n) by myfilter()-517261910033');
33 xlabel('n');
34 ylabel('amplitude')
35 saveas(gcf,'image\fig3_1_6(time)','epsc')
36
37 figure
38 plot(f,2*abs(Y1_f)/Lwin*2.48)
39 title('Y_1(f) by myfilter()-517261910033');
40 xlabel('f');
```

```
ylabel('amplitude')
xlim([0,fs/2])
saveas(gcf,'image\fig3_1_6(freq)','epsc')
```

## A.2 Ex.2

```
clc;clear;close all;
%% init
fc=3;
wc=2*pi*fc;
f = logspace(-1,1);
w=2*pi*f;

fs=10;
T=1/fs;

%% analog
bs = [1];
as = [1/(wc^2) sqrt(2)/wc 1];
H_a = freqs(bs,as,w);
mag_a = abs(H_a);
phase_a = angle(H_a);
phasedeg_a = phase_a*180/pi;

%% digital
bz=[T^2*wc^2/4,T^2*wc^2/2,T^2*wc^2/4];
az=[T^2*wc^2/4-T*wc/sqrt(2)+1,T^2*wc^2/2-2,T^2*wc^2/4+T*wc/sqrt(2)+1];
H_d=freqz(bz,az,f,fs);
mag_d = abs(H_d);
phase_d = angle(H_d);
phasedeg_d = phase_d*180/pi;

%% plot
figure
subplot(2,1,1)
hold on
semilogx(f,mag_a,'b')
semilogx(f,mag_d,'r--')
hold off
grid on
title('amplitude of Butterworth 2-order lowpass filter-517261910033')
legend('analog H_a(f)','digital H_d(f)')
```

```
xlabel('Frequency (Hz)')
ylabel('Magnitude')

subplot(2,1,2)
hold on
semilogx(f,phasedeg_a,'b')
semilogx(f,phasedeg_d,'r--')
hold off
grid on
title('phase of Butterworth 2-order lowpass filter -517261910033')
legend('analog H_a(f)','digital H_d(f)')
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
saveas(gcf,'image\fig3_2','epsc')
```

### A.3 Ex.3

```
clc;clear;close all;
%% init
fs=200;
fc=15;
n=0:199;
df=fs/length(n);
f=0:fs/length(n):fs-fs/length(n);
fn=f/fs;% fnÊÇ¹éÒ»»¯µÄÆµÂÊ

%% signal
f1=8;
f2=23;
A1=4;
A2=8;
xn = A1*cos(2*pi*f1/fs*n) + A2*cos(2*pi*f2/fs*n);
X_f = fft(xn);


%% filter
[b1,a1] = butter(9,fc/(fs/2))
freqz(b1,a1)
saveas(gcf,'image\fig3_3(filter)','epsc')
%% filtrate
y2 = myfilter(b1,a1,xn);
Y2_f = fft(y2);
```

```
save('data\y2_IIR.mat','y2','Y2_f');
%% plot
load('data\y1_FIR.mat')

figure
hold on
plot(n,xn);
plot(n,y1)
plot(n,y2);
hold off
title('time domaine-517261910033')
legend('signal x(n)','signal after FIR y_1(n)','signal after IIR y_2(n)');
ylabel('amplitude')
xlabel('n')
saveas(gcf,'image\fig3_3(time)','epsc')

figure
hold on
plot(f,2*abs(X_f)/length(n));
plot(f,2*abs(Y1_f)/length(n)*2.48)
plot(f,2*abs(Y2_f)/length(n));
hold off
title('frequency domaine-517261910033')
legend('signal X(f)','signal after FIR Y_1(f)','signal after IIR Y_2(f)');
ylabel('amplitude')
xlabel('frequency analog f (Hz)')
xlim([0,fs/2])
saveas(gcf,'image\fig3_3(freq)','epsc')
```

### A.4 Ex.4

```
clc;clear;close all;
%% init


fs=200;


dw=0.001;
w=0:dw:2*pi-dw;
f=fs*w/(2*pi);

```

```
12
13 n=0:1999;
14 Lwin=length(n);
15 %% paramter of signal
16 f1=8;
17 f2=23;
18 A1=4;
19 A2=8;
20
21 xn=A1*cos(2*pi*f1/fs*n)+A2*cos(2*pi*f2/fs*n);
22 X_f=fft(xn,length(f));
23
24 figure
25 plot(n,xn)
26 title('x(n)-517261910033')
27 xlabel('n')
28 ylabel('amplitude')
29 % xlim([0,fs/2])
30 saveas(gcf,'image\fig3_4_0(time)','epsc')
31
32 figure
33 plot(f,2*abs(X_f)/Lwin)
34 xlabel('f')
35 ylabel('amplitude')
36 title('X(f)-517261910033')
37 xlim([0,fs/2])
38 saveas(gcf,'image\fig3_4_0(freq)','epsc')
39 %% ------------------------------------------------------------------------
40 %% paramter of notch
41
42 fc1=23;
43 fcn1=fc1/fs;%0.115
44 wcn1=fcn1*2*pi;%0.23*pi
45
46
47 delta1=0.03;
48 rz1=1;
49 rp1=1-delta1;
50 z1=rz1*exp(1j*wcn1);
51 p1=rp1*exp(1j*wcn1);
52
53 %% FRF of notch
54 z=exp(1j*w);
```

```
55 Hz1=((z-z1).*(z-conj(z1)))./((z-p1).*(z-conj(p1)));
56 Hz1_db=20*log10(abs(Hz1));
57
58
59 figure
60 plot(w/pi,abs(Hz1))
61 xlim([0  1]);
62
63 %%
64 K=1;
65 Zeros1=[z1,conj(z1)];
66 Poles1=[p1,conj(p1)];
67 [bz1,az1]=zp2tf(Zeros1',Poles1',K);
68 figure
69 freqz(bz1, az1)
70 saveas(gcf,'image\fig3_4_1(notchFRF)','epsc');
71 figure
72 pzmap(tf(bz1,az1,-1))
73 saveas(gcf,'image\fig3_4_1(notchZP)','epsc');
74    %    SYS = TF(NUM,DEN,TS) creates a discrete-time transfer function with
75    %    sample time TS (set TS=-1 if the sample time is undetermined).
76
77 y3=filter(bz1,az1,xn);
78 Y3_f=fft(y3,length(f));
79
80 fc=15;
81 [b1,a1] = butter(9,fc/(fs/2));
82 y2 = myfilter(b1,a1,xn);
83 Y2_f = fft(y2,length(f));
84
85 figure
86 subplot(211)
87 hold on
88 plot(n,xn)
89 plot(n,y3)
90 hold off
91 title('x(n) and y_3(n) after notch-517261910033')
92 xlabel('n')
93 ylabel('amplitude')
94 legend('x(n)','y_3(n)')
95
96 subplot(212)
97 hold on
```

```
98  plot(n,xn)
99  plot(n,y2)
100 hold off
101 title('x(n) and y_2(n) after IIR-517261910033')
102 xlabel('n')
103 ylabel('amplitude')
104 legend('x(n)','y_2(n)')
105
106 saveas(gcf,'image\fig3_4_2(notchtime)','epsc');
107
108
109 figure
110 hold on
111 plot(f,2*abs(X_f)/Lwin)
112 plot(f,2*abs(Y2_f)/Lwin,'g:.','linewidth',1)
113 plot(f,2*abs(Y3_f)/Lwin,'r--','linewidth',1)
114
115 xlabel('f')
116 ylabel('amplitude')
117 title('Y_3(f) after notch and Y_2(f)-517261910033')
118 xlim([0,fs/2])
119 legend('X','Y_3 after notch','Y_2 after IIR')
120 saveas(gcf,'image\fig3_4_2(notchfreq)','epsc');
121
122
123
124 %% ------------------------------------------------------------------
125 %% paramter of comb
126
127
128 fc2=8;
129 fcn2=fc2/fs;
130 wcn2=fcn2*2*pi;
131
132
133 delta2=0.01;
134 rz2=1-11*delta2;
135 rp2=1-1*delta2;
136 z2=rz2*exp(1j*wcn2);
137 p2=rp2*exp(1j*wcn2);
138
139 %% FRF of comb
140 z=exp(1j*w);
```

```
141 Hz2=((z-z2).*(z-conj(z2)))./((z-p2).*(z-conj(p2)));
142 Hz2_db=20*log10(abs(Hz2));
143
144 K=1;
145 Zeros2=[z2,conj(z2)];
146 Poles2=[p2,conj(p2)];
147 [bz2,az2]=zp2tf(Zeros2',Poles2',K);
148
149 figure
150 freqz(bz2, az2)
151 saveas(gcf,'image\fig3_4_3(combFRF)','epsc');
152 figure
153 pzmap(tf(bz2,az2,-1))
154 saveas(gcf,'image\fig3_4_3(combZP)','epsc');
155 %%
156 y4=filter(bz2,az2,xn);
157 Y4_f=fft(y4,length(f));
158
159 figure
160 hold on
161 plot(n,xn)
162 plot(n,y3)
163 plot(n,y4)
164 hold off
165 title('y_4(n) after comb and y_3(n) after notch-517261910033')
166 xlabel('n')
167 ylabel('amplitude')
168 legend('xn','y_3 notch','y_4 comb')
169 saveas(gcf,'image\fig3_4_4(combtime)','epsc');
170
171 figure
172 hold on
173 plot(f,2*abs(X_f)/Lwin,'linewidth',1)
174 plot(f,2*abs(Y3_f)/Lwin,'--','linewidth',1)
175 plot(f,2*abs(Y4_f)/Lwin,':')
176 hold off
177 xlabel('f')
178 ylabel('amplitude')
179 title('Y_4(f) after comb and Y_3(f) after notch-517261910033')
180 xlim([0,fs/2])
181 legend('X','Y_3 notch','Y_4 comb')
182 saveas(gcf,'image\fig3_4_4(combfreq)','epsc');
```

## A.5 Ex.5

```
clc;close all;clear;
%%
[bird,fs1]=audioread('audio/birds.wav');
[bill,fs2]=audioread('audio/Bill Gates.wav');
[melinda,fs3]=audioread('audio/Melinda Gates.wav');
[mix,fs4]=audioread('audio/mix_Gates.wav');
% sound(bill,fs2);
% sound(melinda,fs3);
% sound(mix,fs4);

%choose signal channel to process
bird=bird(:,1);
bill=bill(:,1);
melinda=melinda(:,1);
mix=mix(:,1);



%%
L1=length(bird);
L2=length(bill);
L3=length(melinda);
L4=length(mix);

t1=0:L1-1;
t2=0:L2-1;
t3=0:L3-1;
t4=0:L4-1;

figure
subplot(411)
plot(t1,bird)
title('birds')
xlabel('t')

subplot(412)
plot(t2,bill)
title('bill')
xlabel('t')

subplot(413)
```

```
42 plot(t3,melinda)
43 title('melinda')
44 xlabel('t')
45
46
47 subplot(414)
48 plot(t4,mix)
49 title('mix')
50 xlabel('t')
51 saveas(gcf,'image\fig3_5_1(time)','epsc')
52
53 %%
54 birdf=fft(bird);
55 billf=fft(bill);
56 melindaf=fft(melinda);
57 mixf=fft(mix);
58
59 df1=fs1/(L1-1);
60 df2=fs2/(L2-1);
61 df3=fs3/(L3-1);
62 df4=fs4/(L4-1);
63
64 f1=0:df1:fs1;
65 f2=0:df2:fs2;
66 f3=0:df3:fs3;
67 f4=0:df4:fs4;
68
69 figure
70 subplot(221)
71 plot(f1,abs(birdf)*2)
72 title('birds')
73 xlim([0 fs1/8])
74 xlabel('f')
75
76
77 subplot(222)
78 plot(f2,abs(billf)*2)
79 title('bill')
80 xlim([0 fs2/8])
81 xlabel('f')
82
83 subplot(223)
84 plot(f3,abs(melindaf)*2)
```

```
85 title('melinda')
86 xlim([0 fs3/8])
87 xlabel('f')
88
89
90 subplot(224)
91 plot(f4,abs(mixf)*2)
92 title('mix')
93 xlim([0 fs4/8])
94 xlabel('f')
95 saveas(gcf,'image\fig3_5_1(freq)','epsc')
96
97 %%
98 % Fs=44100, butterworth,Fpass=176.4,Fstop=250,Apass=1,Astop=3
99 load('data/Exe3_5_2.mat')
100
101 [b,a]=sos2tf(SOS,G);
102 figure
103 freqz(b,a)
104 saveas(gcf,'image\fig3_5_2(filter)','epsc')
105
106 Filtered=filter(b,a,mix);
107 Filteredf=fft(Filtered);
108
109 figure
110 hold on
111 plot(f4,abs(mixf)*2)
112 plot(f4,abs(Filteredf)*2)
113 title('mix_Gates.wav and after filter in frequency domaine')
114 legend('mix of Bill & Melinda','Bill part after filter')
115 xlim([0 fs3/8])
116 xlabel('f')
117 saveas(gcf,'image\fig3_5_2(billafterfilterfreq)','epsc')
118
119 sound(Filtered,fs4);
120
121 %%
122 billbird=bill+bird;
123 billbirdf=fft(billbird);
124
125 figure
126 plot(t1,billbird)
127 title('mix of Bill Gates.wav and birds.wav in time domaine')
```

```
128 xlabel('t')
129
130
131 figure
132 plot(f1,abs(billbirdf)*2)
133 xlim([0 fs1/8])
134 title('mix of Bill Gates.wav and birds.wav in frequency domaine')
135 xlabel('f')
136 saveas(gcf,'image\fig3_5_3(billbirdfreq)','epsc')
137
138 sound(billbird,fs1);
139
140 %%
141 % Fs=44100, butterworth,Fpass=1000,Fstop=1650,Apass=1,Astop=25
142 load('data/Exe3_5_3(Lowpass).mat')
143
144 [b1,a1]=sos2tf(SOS,G);
145 figure
146 freqz(b1,a1)
147 saveas(gcf,'image\fig3_5_3(lowpass)','epsc')
148
149 Filteredbill=filter(b1,a1,billbird);
150 Filteredbillf=fft(Filteredbill);
151
152 figure
153 hold on
154 plot(f1,abs(billbirdf)*2)
155 plot(f1,abs(Filteredbillf)*2)
156 title('Bill part after filter in frequency domain')
157 legend('billbird','bill after filter')
158 xlim([0 fs1/8])
159 xlabel('f')
160 saveas(gcf,'image\fig3_5_3(billafterlowpassfreq)','epsc')
161
162 sound(Filteredbill,fs1);
163
164 %%
165 % Fs=44100, butterworth,Fstop=1000,Fpass=2000,Apass=1,Astop=30
166 load('data/Exe3_5_3(Highpass).mat')
167
168 [b2,a2]=sos2tf(SOS,G);
169 figure
170 freqz(b2,a2)
```

```
171 saveas(gcf,'image\fig3_5_3(highpass)','epsc')
172
173
174 Filteredbird=filter(b2,a2,billbird);
175 Filteredbirdf=fft(Filteredbird);
176
177 figure
178 hold on
179 plot(f1,abs(billbirdf)*2)
180 plot(f1,abs(Filteredbirdf)*2)
181 title('Bird part after filter in frequency domain')
182 legend('billbird','bird after filter')
183 xlim([0 fs1/8])
184 xlabel('f')
185 saveas(gcf,'image\fig3_5_3(birdafterhighpass)','epsc')
186
187
188 sound(Filteredbird,fs1);
```

### A.6 Ex.6

```
1  clc;clear;close all;
2  %% init
3  fs=200;
4  fc=15;
5  n=0:199;
6  df=fs/length(n);
7  f=0:fs/length(n):fs-fs/length(n);
8  fn=f/fs;
9
10 %% signal
11 f1=8;
12 f2=23;
13 A1=4;
14 A2=8;
15 xn = A1*cos(2*pi*f1/fs*n) + A2*cos(2*pi*f2/fs*n);
16 X_f = fft(xn);
17
18
19
20 %% filtrate
21
```

```
% Fs=200, butterworth,n=9,Fc=15
load('data/Exe3_6.mat');

y2prime = mySOSfilter(SOS,G,xn);
Y2prime_f = fft(y2prime);



%% plot
load('data\y1_FIR.mat')

figure
hold on
plot(n,xn);
plot(n,y1)
plot(n,y2prime);
hold off
title('time domaine-517261910033')
legend('signal x(n)','signal after FIR y_1(n)','signal after IIR y_2''(n)(SOS)');
ylabel('amplitude')
xlabel('n')
saveas(gcf,'image\fig3_6(time)','epsc')

figure
hold on
plot(f,2*abs(X_f)/length(n));
plot(f,2*abs(Y1_f)/length(n)*2.48)
plot(f,2*abs(Y2prime_f)/length(n));
hold off
title('frequency domaine-517261910033')
legend('signal X(f)','signal after FIR Y_1(f)','signal after IIR Y_2''(f)(SOS)');
ylabel('amplitude')
xlabel('frequency analog f (Hz)')
xlim([0,fs/2])
saveas(gcf,'image\fig3_6(freq)','epsc')
```