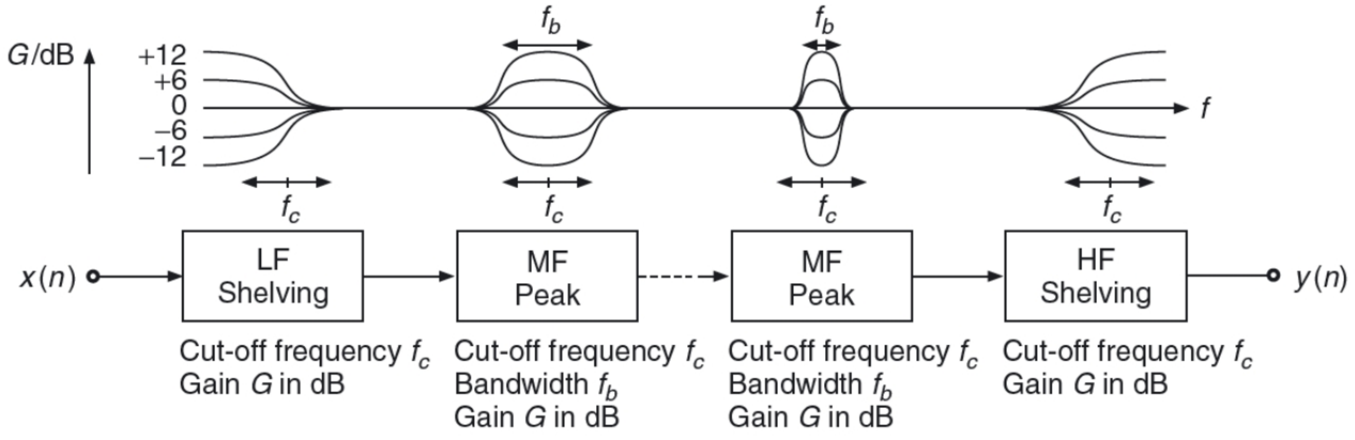
Audio Program Design and Application - 5

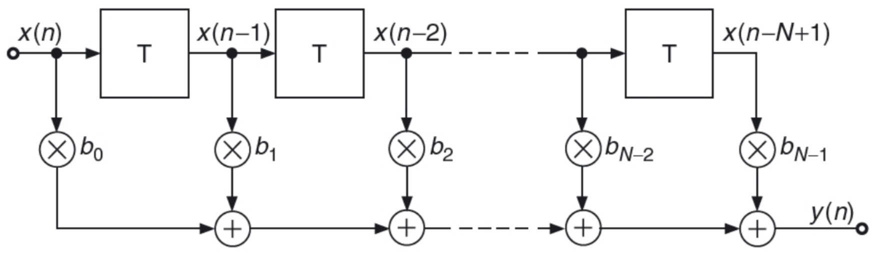
Reporter : Chuhan Qin Student Number : 2020209023026 Major : Recording Engineer

Equalizer, which is the most widely used digital audio effect , is actually the basic component of many complex audio effects, such as delay, vibrato, flanger, doubler, reverb, etc. In some ways, equalizer can be seen as an extension, or combinations, of filters. In our course, there is a picture from DAFX which clearly shows the main filter categories used in equalizer : LowPass/HighPass shelving filters and Peak filters. The picture is as following:



But these filters are not the keynote of this report. To design such filters, we must know the different construct of basic filter. In the course of last term, we studied FIR/IIR filter and their difference in pole-zero plot. However, the pole-zero plot is only an analytical perspective which cannot be used to explain the reason. Actually, explaining thier predominant difference can start with their structures and formulas.

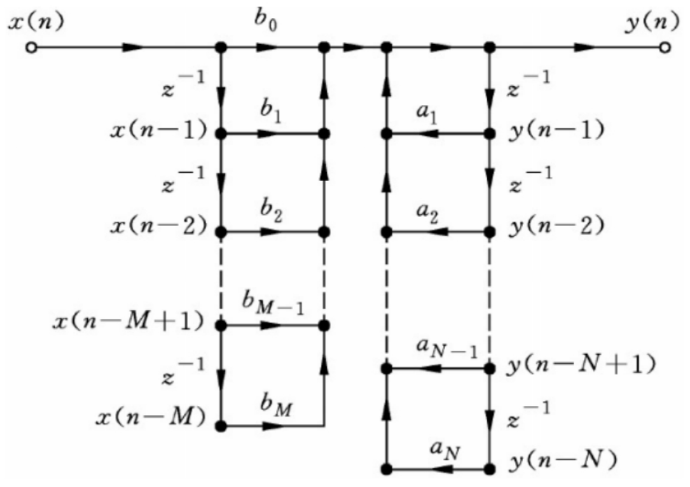
For FIR filter, it has only one direct form, whose diagram and formula are as following:



$$
\begin{aligned}
y(n) &= \sum_{i=0}^{N-1} b_i \cdot x(n-i) \\
&= b_0 x(n) + b_1 x(n-1) + \ldots + b_{N-1} x(n-N+1)
\end{aligned} \tag{1}
$$

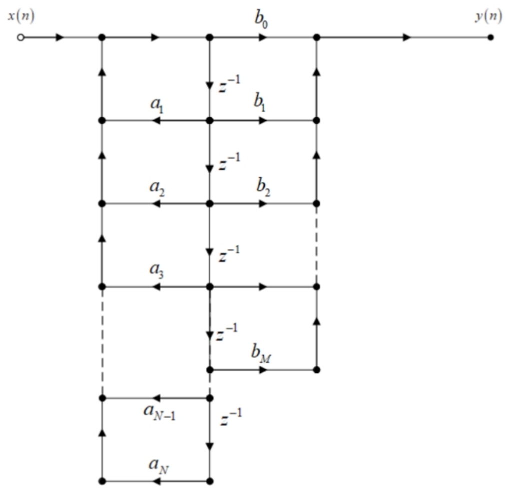
For IIR filter, its structure involves direct form I and II, and the diagrams and formulas are as following:

> Direct Form I



$$y(n) = \sum_{i=0}^{N} b_i \cdot x(n-i) + \sum_{i=1}^{M-1} a_i \cdot y(n-i) \tag{2}$$

> Direct Form II

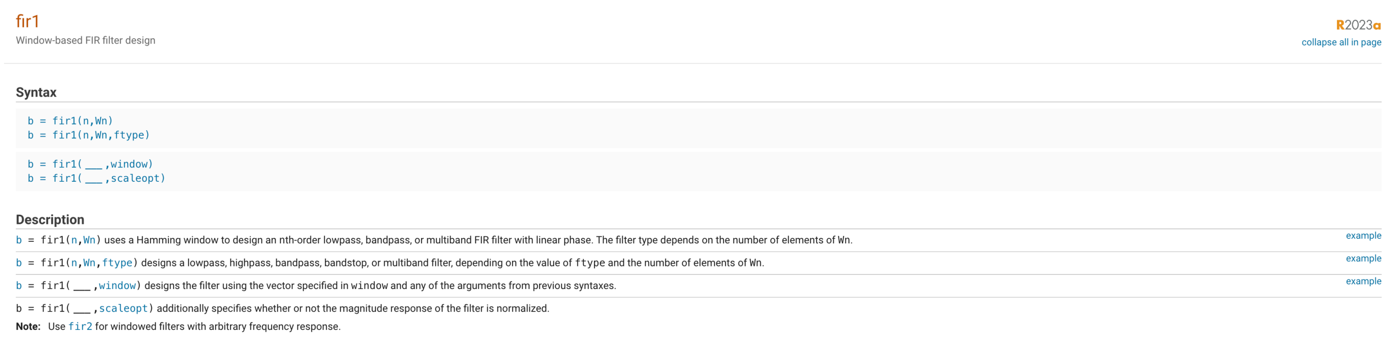


$$\begin{gathered} x_h(n) = x(n) - a_1 x_h(n-1) - \ldots - a_N x_h(n-N) \\ y(n) = b_0 x_h(n) + b_1 x_h(n-1) + \ldots + b_M x_h(n-M) \end{gathered} \tag{3}$$

Obviously, the main difference between FIR and IIR is the feedback in the structure of IIR, which causes that not all the poles of IIR filter locate at the origin. And the direct form II of IIR is also called Canonical Filters, which demands the least amount of delay units. In order to simplify the debugging complexity of high order filters, we usually decomposed the high order filter into a combination of multiple first or second order canonical filters. That's why most articles and books only provide the design formulas of first and second order filters, including some more complex filters such as wraped linear prediction filter.

In this report, we mainly focus on designing different types of FIR filters and IIR canonical filters, resembling bandpass filter. Matlab provides many functions for filter designing and I use the function *fir1* to get the impulse response of FIR highpass filter, bandpass filter and bandstop filter. Then I use the coefficients for first and second order canonical filters in DAFX, to implement the first-order highpass filter and second-order highpass canonical filter, bandpass canonical filter and bandstop canonical filter. At last, I plot their frequency response and phase response with function *freqz* in Matlab, then using function *filter* and the coefficients of them to filter the audio signal.

Function *fir1* in Matlab is actually based on least-squares approximation to compute the coefficient and smooths the impule response with window which has been discussed in our course:



fir1
Window-based FIR filter design

R2023a
collapse all in page

Syntax

```
b = fir1(n,Wn)
b = fir1(n,Wn,ftype)

b = fir1(___,window)
b = fir1(___,scaleopt)
```

Description

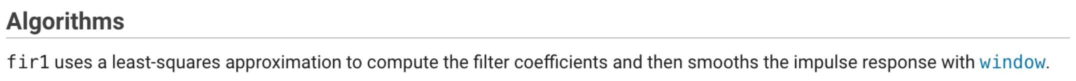
`b = fir1(n,Wn)` uses a Hamming window to design an nth-order lowpass, bandpass, or multiband FIR filter with linear phase. The filter type depends on the number of elements of `Wn`. example

`b = fir1(n,Wn,ftype)` designs a lowpass, highpass, bandpass, bandstop, or multiband filter, depending on the value of `ftype` and the number of elements of `Wn`. example

`b = fir1(___,window)` designs the filter using the vector specified in `window` and any of the arguments from previous syntaxes. example

`b = fir1(___,scaleopt)` additionally specifies whether or not the magnitude response of the filter is normalized.

Note: Use `fir2` for windowed filters with arbitrary frequency response.



Algorithms

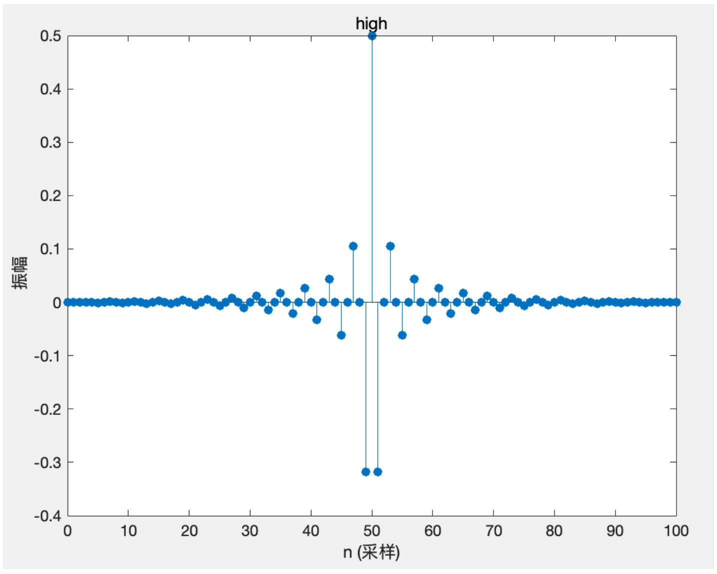
`fir1` uses a least-squares approximation to compute the filter coefficients and then smooths the impulse response with `window`.

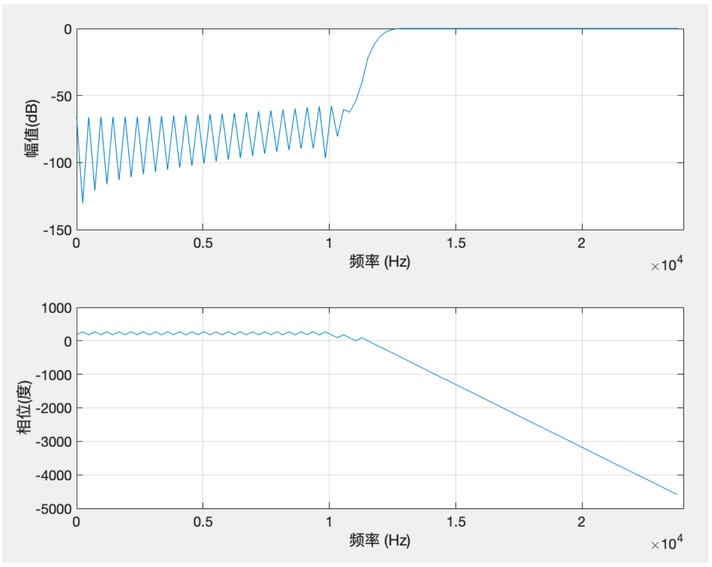
By windowing the impule response, the gain of filter's pass-band has less fluxtuations and stop-band attenuation becomes more obvious, while it also introduces some delay to the filter system. Through the use of function *fir1*, we needn't to implement the additional windowing operation and its algorithm is certainly more robust. So *fir1* is the best choice for us to finish this part of work.

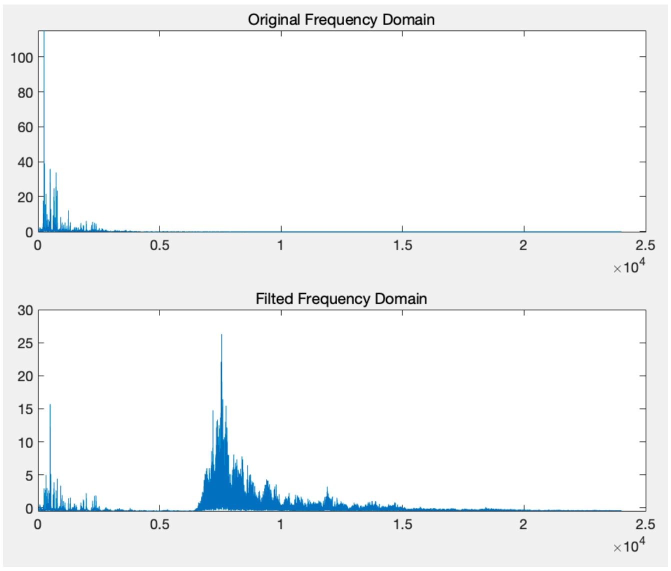
Function *fir1* mainly has three parameters for us to revise if we wanna FIR filter with different order, cut-off frequency or type. Wherein the unit of cut-off frequency parameter W_n of *fir1* should be strictly greater than 0 and smaller than 1, where 1 coresponds to the Nyquist frequency. And W_n can be seecified as a scalar, a two-element vector or a multi-element vector. When W_n is a multi-element vector, *fir1* will return an multiband filter which is a really convenient method.

Here I test an audio signal which has been used in the past reports. And I compute the FFT of original signal and filtered signal to show the difference caused by the filter. the result of FIR filter is shown as following:

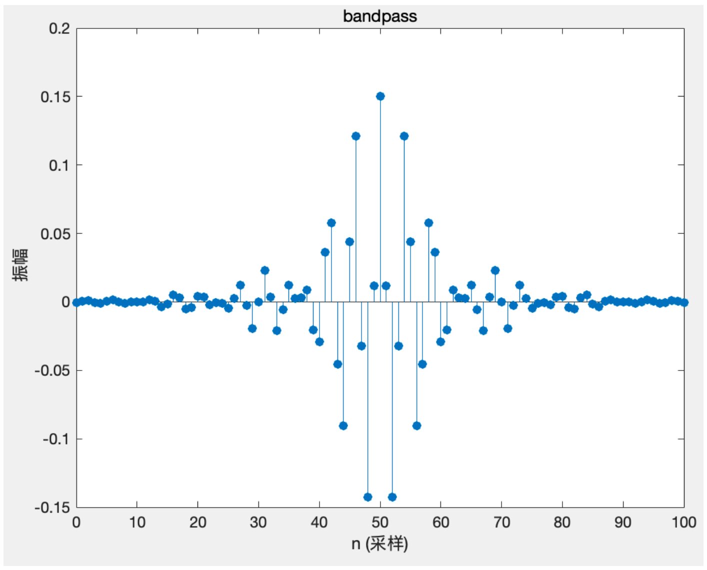
> HighPass FIR
>
> Cut-off Parameter: 0.3

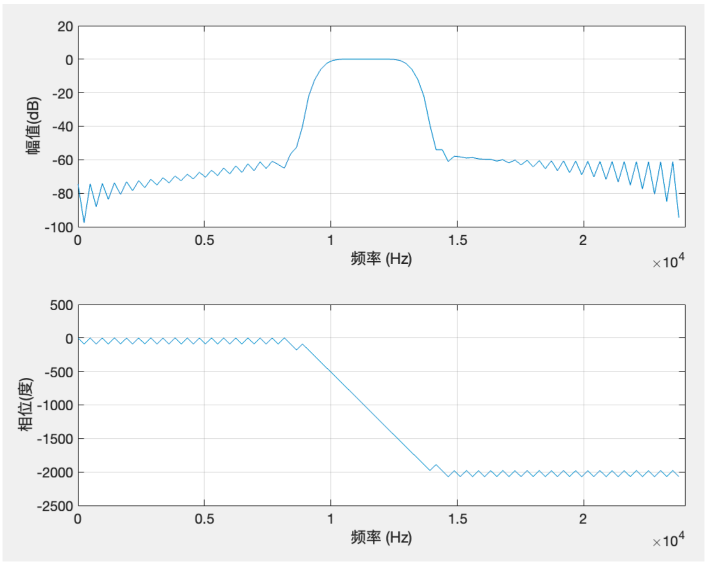


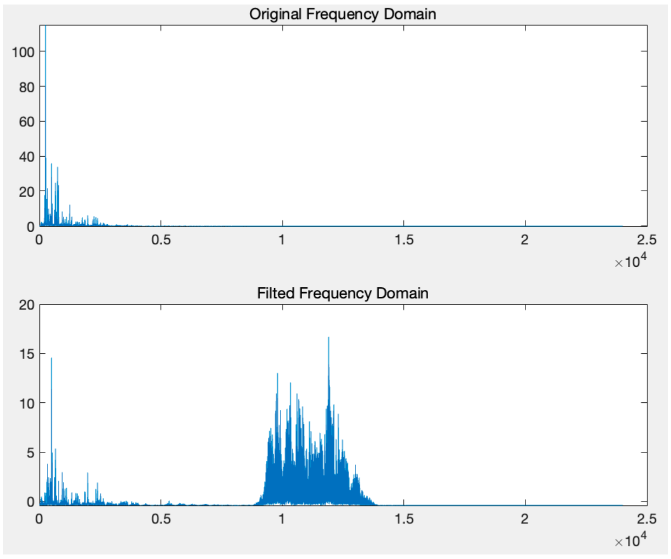




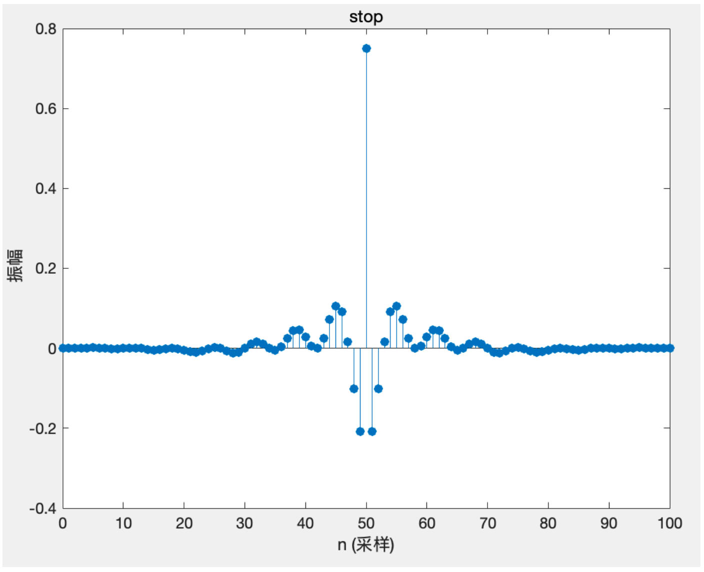
> BandPass FIR
>
> Cut-off Parameter: 0.4~0.55

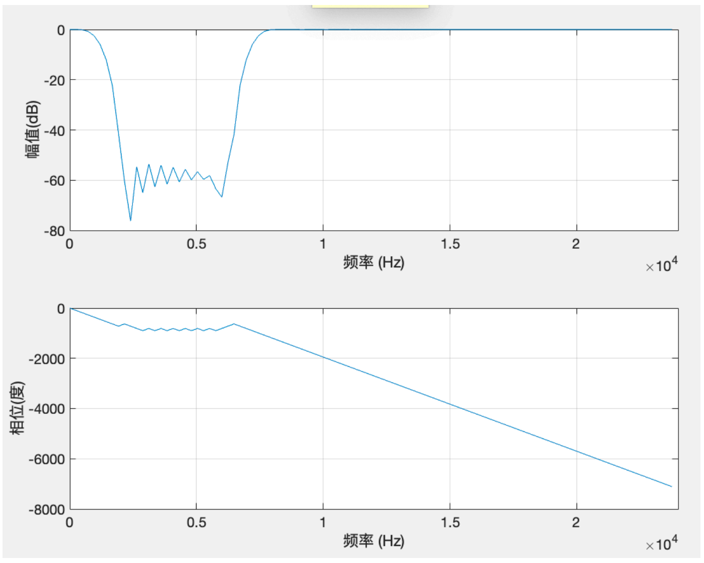


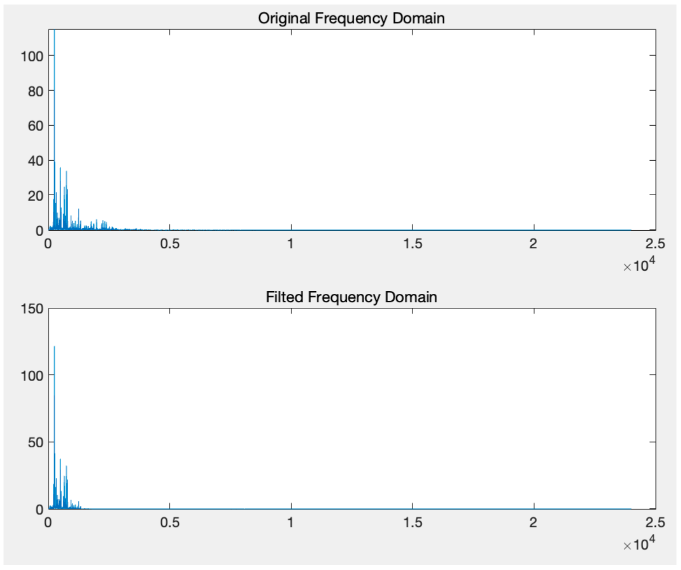




> BandStop FIR
>
> Cut-off Parameter: 0.05~0.3







To better perform the FFT results of them, I normalize the amplitude spectrum with the function *normalize* and it clearly reflects the changes caused by FIR filter. Both the spectrums of highpass and band pass filter appear a strange hump around 10kHz, that's because the predominant frequency components of origninal signals have been filtered out and the components that were orginally very small become relatively larger. From the sense of hearing, the highpass and bandpass filter has indeed filtered out the predominant frequency which even makes nothing can be heard.

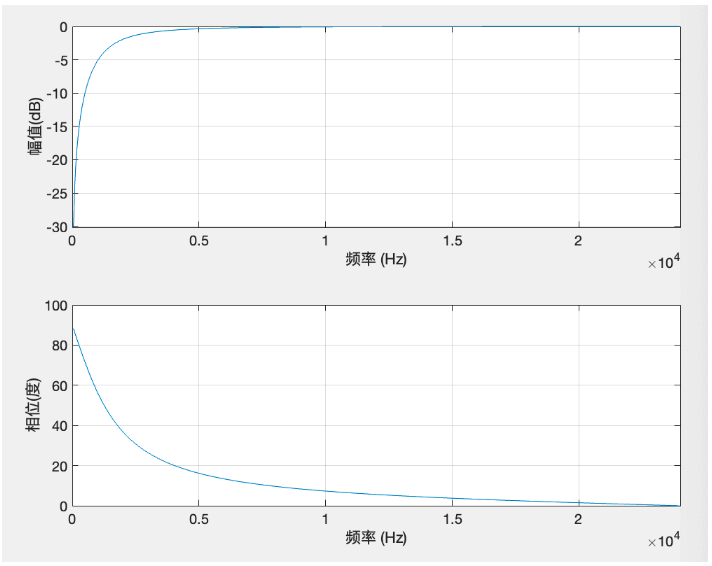
For IIR canonical filters, our course has produced us the coefficient computing formulas, which are shown as follows:

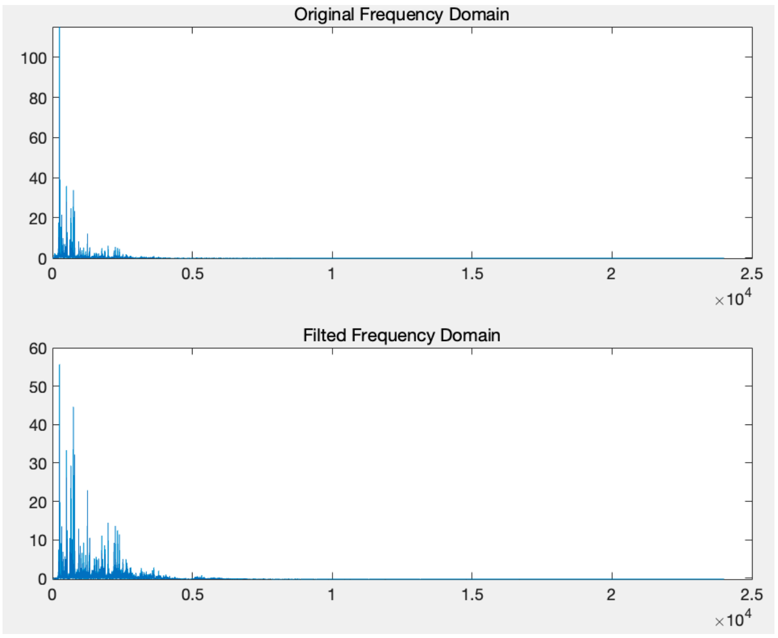
	b_0	b_1	a_1
Lowpass	$K/(K+1)$	$K/(K+1)$	$(K-1)/(K+1)$
Highpass	$1/(K+1)$	$-1/(K+1)$	$(K-1)/(K+1)$

	b_0	b_1	b_2	a_1	a_2
Lowpass	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2K^2Q}{K^2Q+K+Q}$	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2Q\cdot(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Highpass	$\frac{Q}{K^2Q+K+Q}$	$-\frac{2Q}{K^2Q+K+Q}$	$\frac{Q}{K^2Q+K+Q}$	$\frac{2Q\cdot(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Bandpass	$\frac{K}{K^2Q+K+Q}$	0	$-\frac{K}{K^2Q+K+Q}$	$\frac{2Q\cdot(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Bandreject	$\frac{Q\cdot(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q\cdot(K^2-1)}{K^2Q+K+Q}$	$\frac{Q\cdot(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q\cdot(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

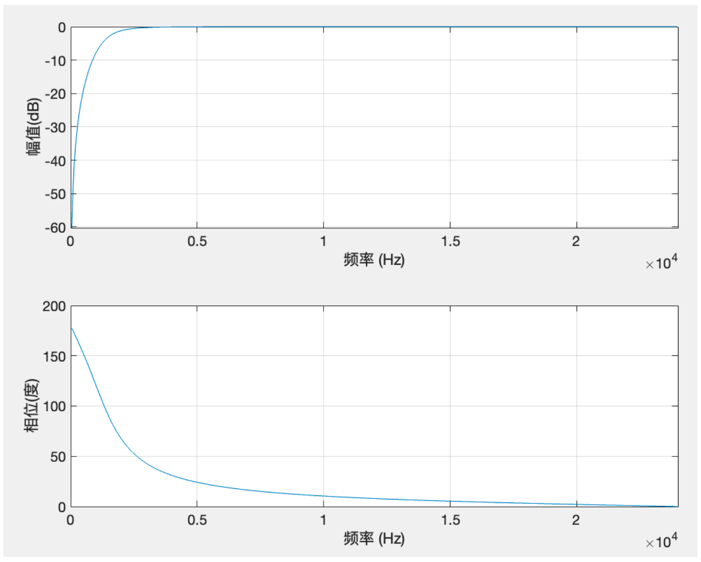
In this way, what we need to do is only implement them in Matlab. So I encapsulate a function named 'IIR_Designer', which implements all the formulas above and filtering the input audio signal with a comparison of the spectrums as well . The result of them are as the following:

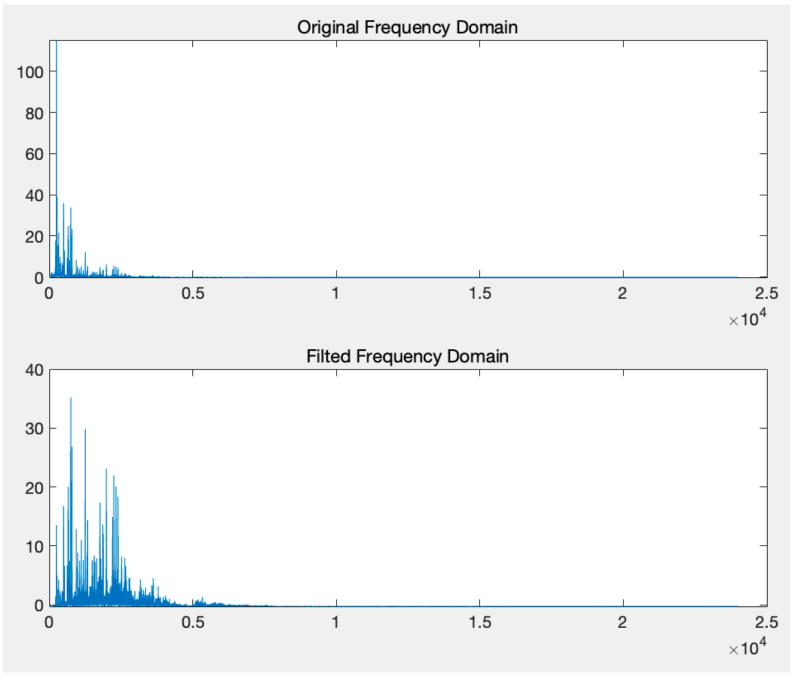
> First Order Highpass
>
> Cut-off Frequency: 1.5 kHz



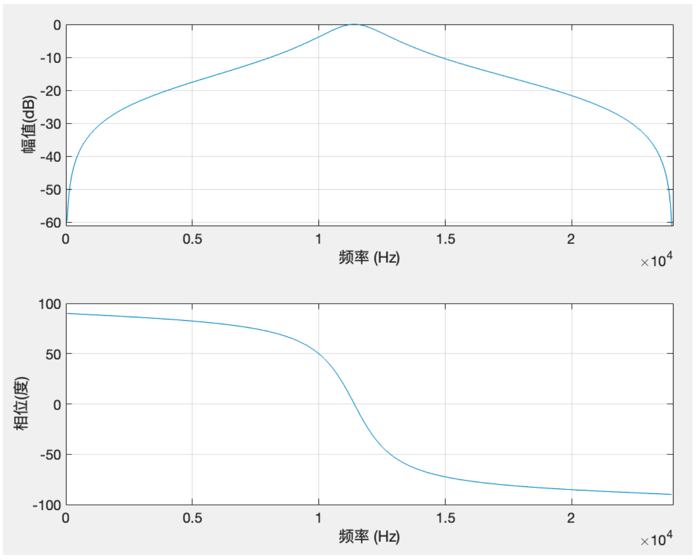


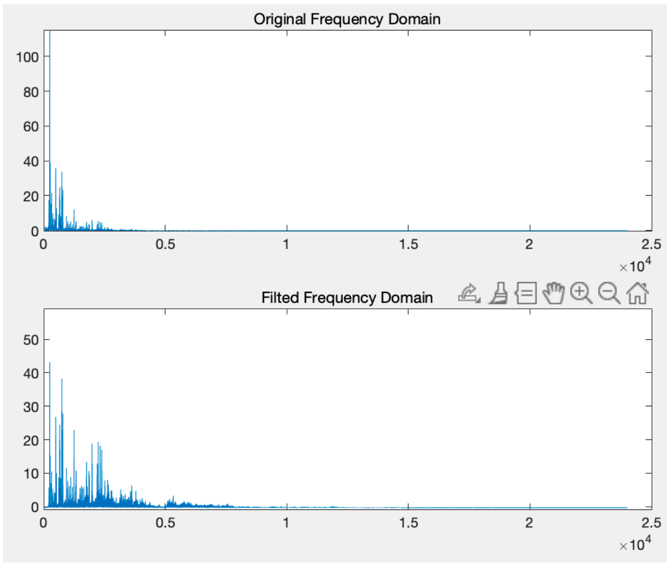
> Second Order Highpass
>
> Cut-off Frequency: 1.5 kHz



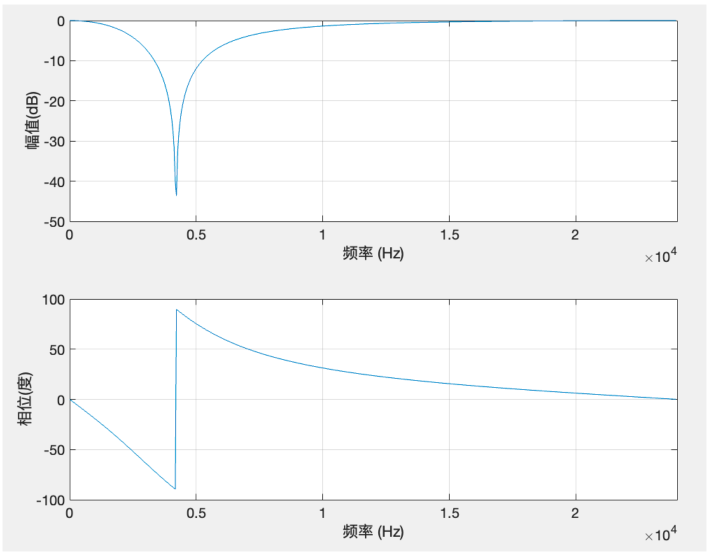


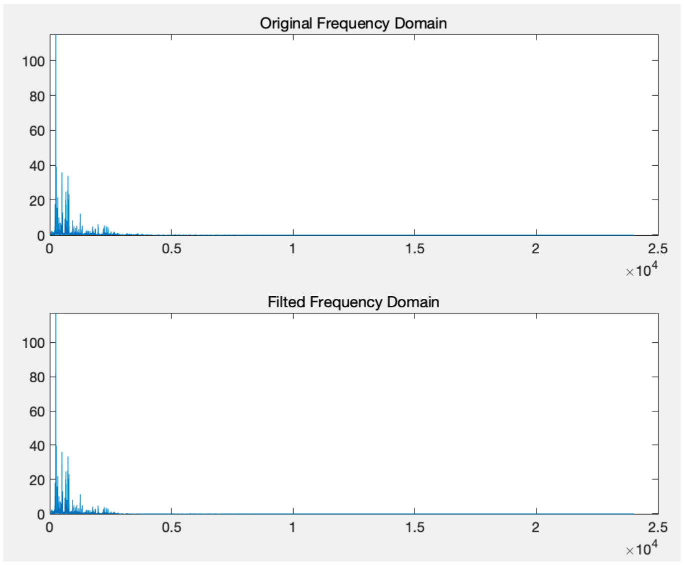
> Second Order Bandpass
>
> Cut-off Frequency: 11400 Hz
>
> Band Width: 3600 Hz





> Second Order Bandstop
>
> Cut-off Frequency: 4.2 kHz
>
> Band Width: 6 kHz





IIR filters' phase response are nonlinear, which differs from FIR filter. From the figures of frequency response, obviously, second-order filter has better attenuation in stopband than first-order. Even analyzing in hearing perception, it's pretty obvious that second-order has better effect.

为了更好地对比IIR和FIR带通带阻滤波器之间的听感区别，我特意将二阶典范带通和带阻滤波截止频率与带宽设置的和FIR基本一致，从听感上来分析，典范滤波的带通没有像FIR那样完全干净的滤除，还是保留了一些可闻的谐波成分，从它的频率响应上就可以看出，其阻带的衰减不像100阶的FIR滤波器那样显著；典范带阻滤波器同样也没有100阶的FIR滤波器阻带衰减那么明显，同时相对而言最大阻带要窄很多，曲线的形状就跟FIR滤波器不同，是内凹的而非类似矩形的形状。上述是对100阶的FIR滤波器和2阶的典范滤波比较，尽管如此2阶的典范滤波已经效果非常不错了，可以看出对于同样的设计指标，IIR滤波器设计的阶数要求要相比FIR小很多。

综上所述，完成了对FIR和典范滤波器的实现，并计算了相应的频率和相位响应，同时对音频进行了滤波和效果分析，综合对比了不同结构滤波器之间的差异。

(这周真的太赶了，写到凌晨四点才写完全部报告和代码，若有部分缺漏的地方敬请谅解