Lab7 Digital Filter Design (part 1)

11911116 徐嘉睿

**7.1 Introduction**

In this lab, we will get to know a way to design the digital filters by given filter specifications.

There are two main method introduced in this lab.

The first one is using corresponding transfer function to get the filter. We have already learnt the basic four type of FIR filters and how the cutoff frequency and bandwidth be determined in theoretical course. The procedure in lab is to implement a simple FIR filter by the theory above. For IIR filers, it is similarly, we a standard transfer function to implement the filter by setting the parameters correctly according to the filter specifications.

The second one is using truncation function to get a portion of ideal filter in time domain. For the impulse response of an ideal filter is infinite, which cannot be created in real world. The main idea is to use a kind of truncation function to get finite points of the ideal filter thus it is implementable. The specification is determined by the ideal filter and the truncation function will influence the ripple and transition band. In this lab we only focus on the simple rectangle truncation function, and the other functions will be introduced in the lab8.

Digital filter design requires the use of both frequency domain and time domain techniques. This is because filter design specifications are often given in the frequency domain, but filters are usually implemented in the time domain with a difference equation. Typically, frequency domain analysis is done using the z-transform and the discrete-time Fourier Transform (DTFT).

Note that this part needs some theoretical analysis thus there are some derivation referred from the lecture note and lab manual to get the whole lab process more clearly

**7.2 Background of Digital Filters**

In general, a linear and time-invariant causal digital filter with input and output may be specified by its difference equation

Equation 7.1

Where and are coefficients which parameterize the filter. The filter is said to have zeros and poles. The impulse response is the response of the filter to an input of , and therefore the solution to the recursive difference equation is

Equation 7.2

There are two general classes of digital filter, IIR and FIR filters.

And the next two sections will demonstrates the two filters in detail.

Implement z-transform to the two side of the equation 7.2, and we get

Equation 7.3

And when , we get the frequency response of the filter by evaluating on the unit circle

Equation 7.4

**7.3 Design of a Simple FIR Filter**

FIR filter do not use any value of previous of itself .From equation 7.2 and the basic principle of FIR filter, we can get that the FIR case occurs when for all . Such a filter is said to have no poles with only zeros. In this case, the difference equation in 7.2 is reduced to

Equation 7.5

Since equation 7.5 is no longer recursive, the impulse response has finite duration .

And from equation 7.3, the z-transform of a FIR filter is

Equation 7.6

**7.3.1 Second Order FIR Filter**

In this section, we will use a simple second order FIR filter with two zeros on the unit circle to illustrate the use of zeros in filter design. In order for the filter’s impulse response to be real-valued, the two zeros must be complex conjugates (or symmetry about Re-axis in z-plane) of one another:

Where is the angle of relative to the positive real axis.

Thus the transfer function for this filter is

Equation 7.7

Due to the property of z-transform, we can just use to replace to get the difference equation of the known z-transform. And note that the order of determines the delay of

Thus the analytical expression of the impulse response for the filter is

Equation 7.8

And replace the impulse function with input and with , we get the difference equation of the filter

Equation 7.9

From the equation above we can get a version of the system diagram of the filter. This system diagram is shown in figure 7.3.1

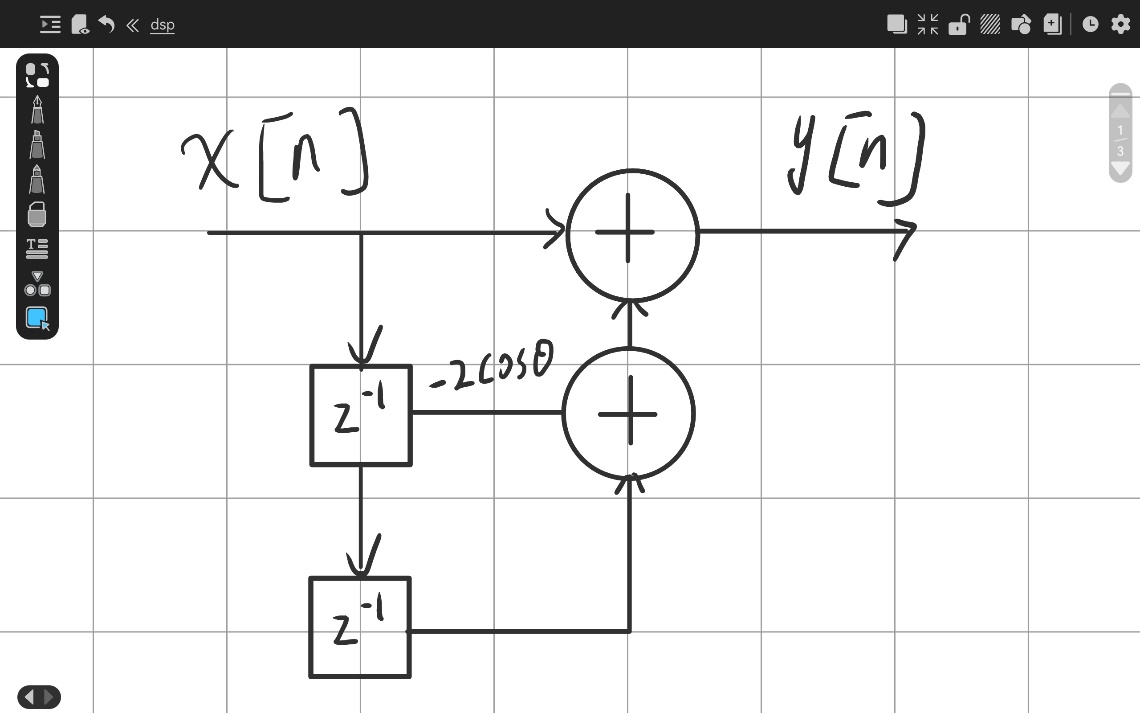


Figure 7.3.1

We choose several value of , which is , to check the frequency response of the filters.

The figure 7.3.2 below shows the magnitude response of the filter when equals to the three values above

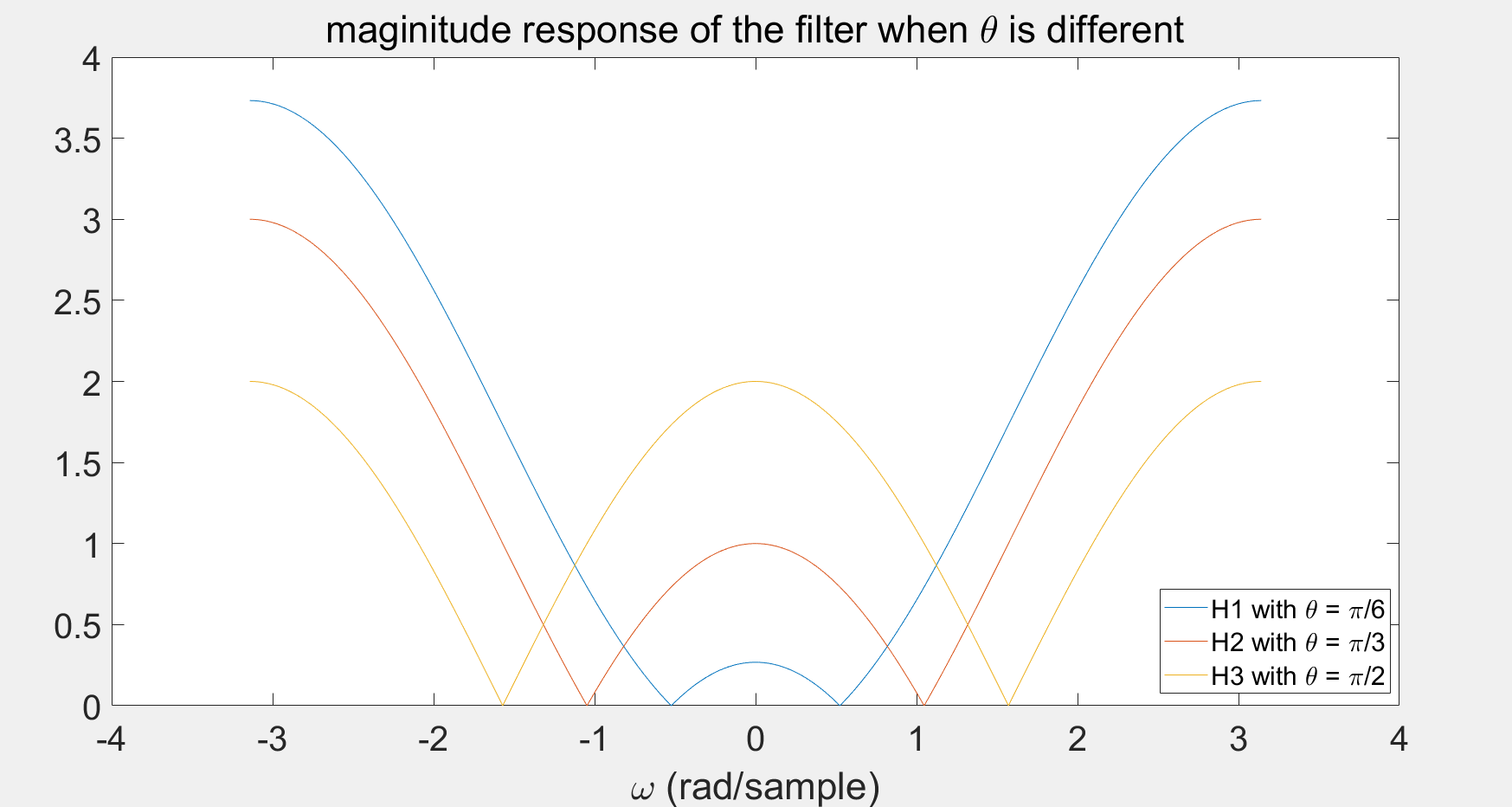
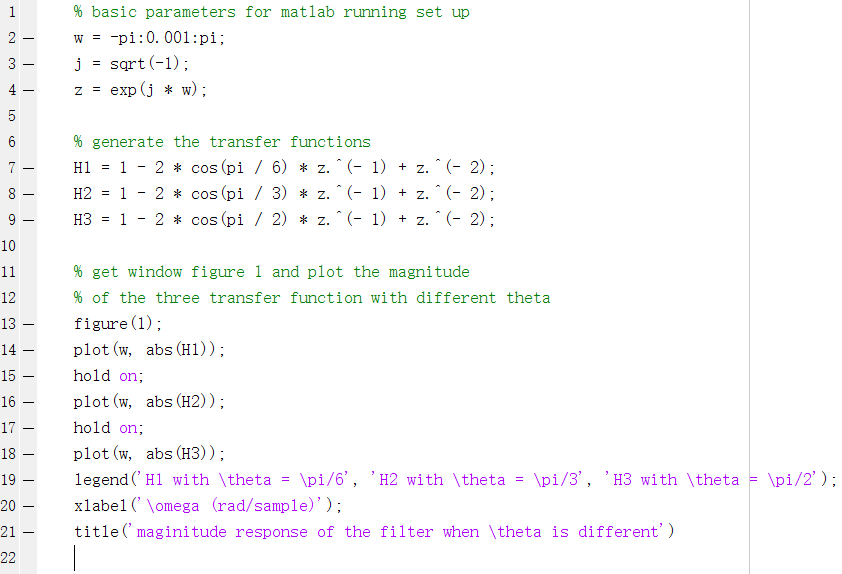


Figure 7.3.2

And the code for this script is shown as code 7.3.1 (corresponding to the file lab7\_3\_1.m)



Code 7.3.1

The figure shows that the determines where the magnitude of the filter’s frequency response gets to zero. This property can get the desired frequency be attenuated largely.

Also note that the shape of the magnitude of the filter’s frequency response is the same with different value of but without considering the absolute.

**7.3.2 Filtering the Audio Signal**

In this section, we use the filter above to filter an audio signal with intense sinusoidal interference added.

The time domain plot of 101 samples with range (100:200) is shown in the figure 7.3.3

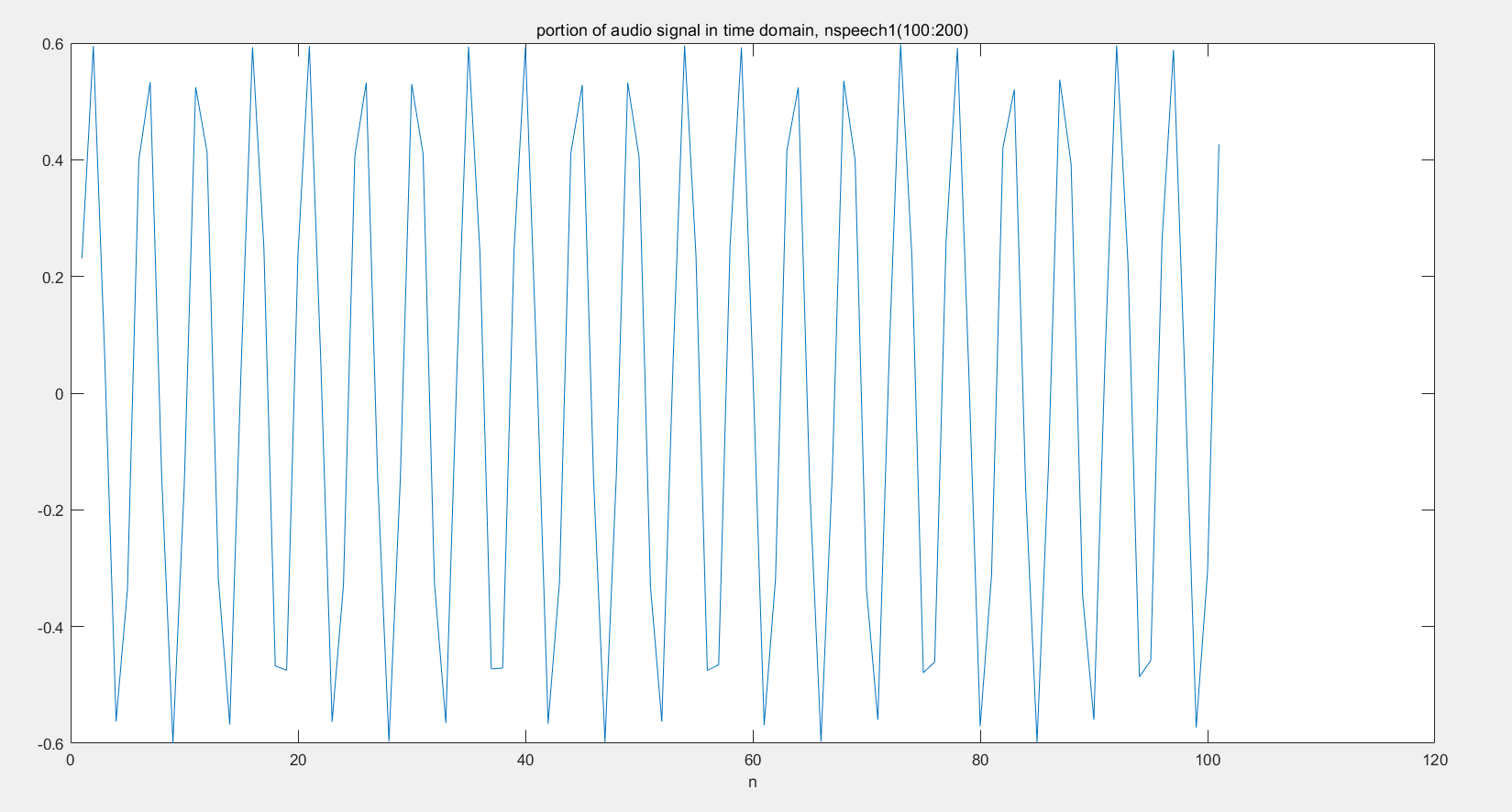


Figure 7.3.3

The figure shows that the real audio waveform can hardly be seen and most of the waveform is the sinusoidal signal

The plot of the magnitude of the DTFT for 1001 samples with range (100:1100) is shown in the figure 7.3.4. Here the length of DTFT is set to

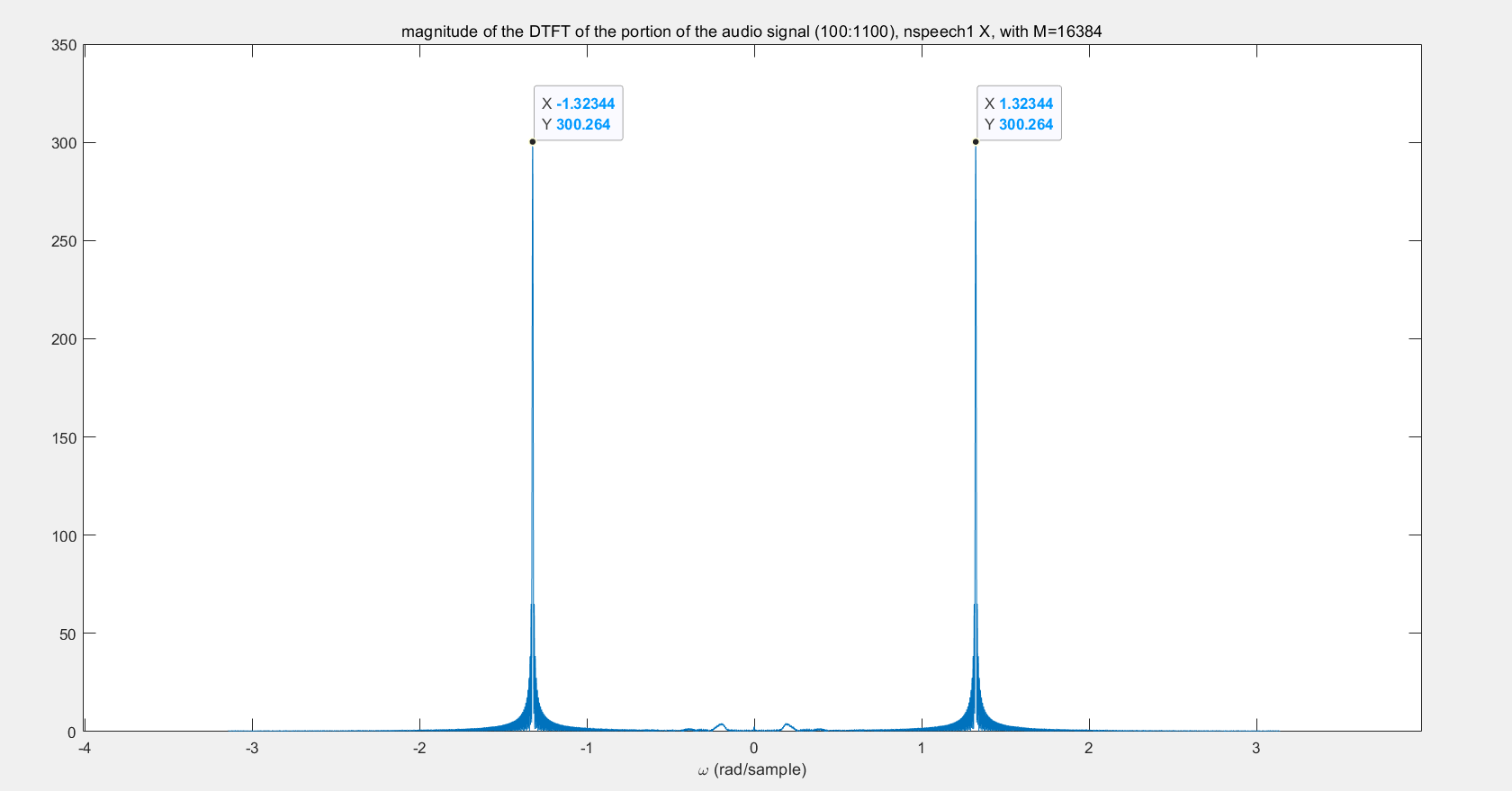


Figure 7.3.4

We can see clearly that the intense sinusoidal signal greatly suppressed the audio signal we wanted. Also the sound test shows that there only the ‘beeps’ sound that caused by the sinusoidal signal can be heard.

Additionally we can get that the estimated is around in the figure just as indicated.

We use the ‘max’ function to get the exact value of the of the sinusoidal signal and set this to the specification of FIR filter designed in the last section.

The figure 7.3.5 shows the DTFT of the designed filter in these filter specifications

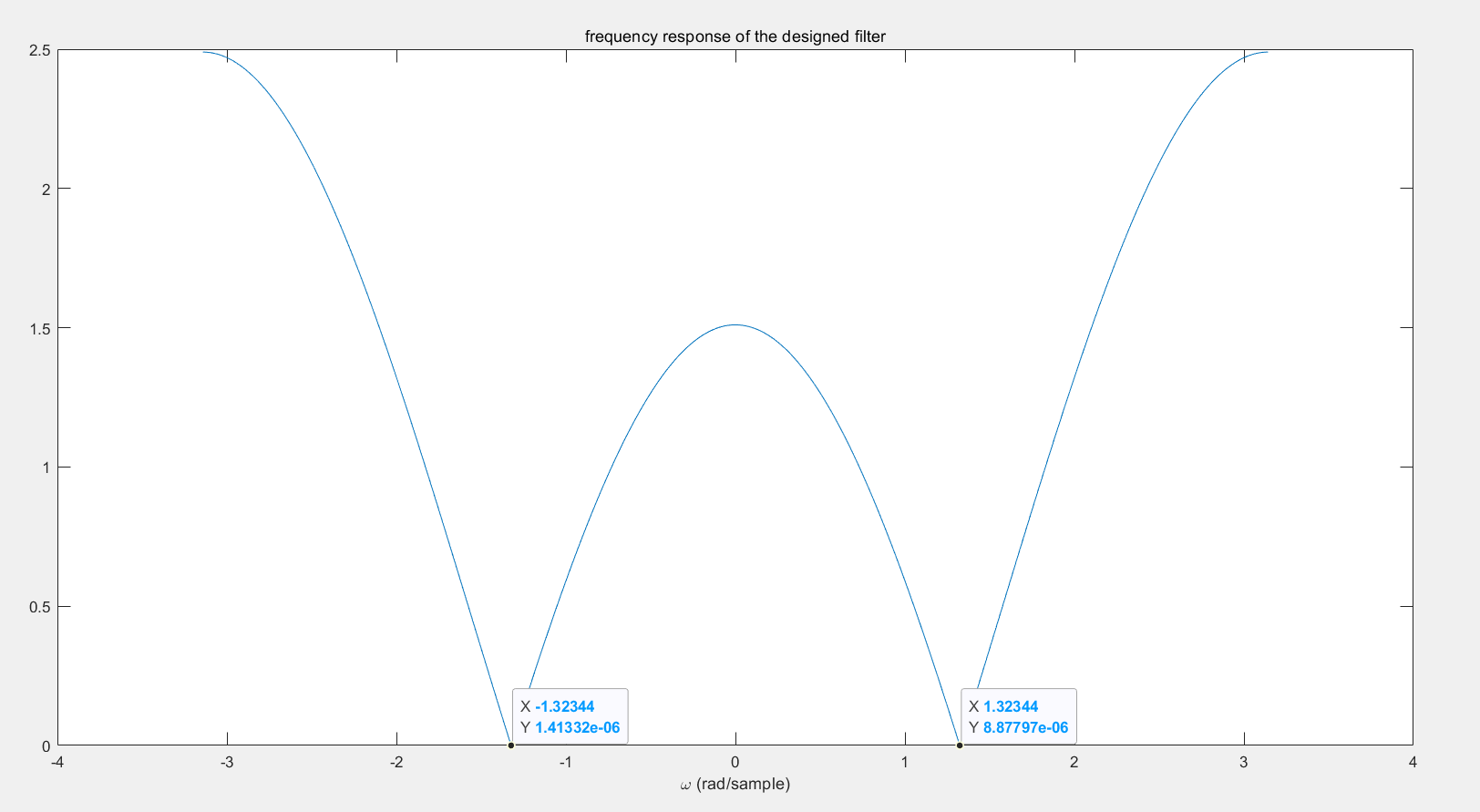


Figure 7.3.5

From the frequency response of the filter above, we know that it has two zero points. Once we set the frequency of the two zero points the one of the sinusoidal interference in the audio signal, we can attenuate the sinusoidal signal to zero in theory.

After the convolution (filtering), we can get the filtered signal without the sinusoidal signal, the time domain plot of 101 samples filtered audio signal with range (100:200) is shown in the figure 7.3.6

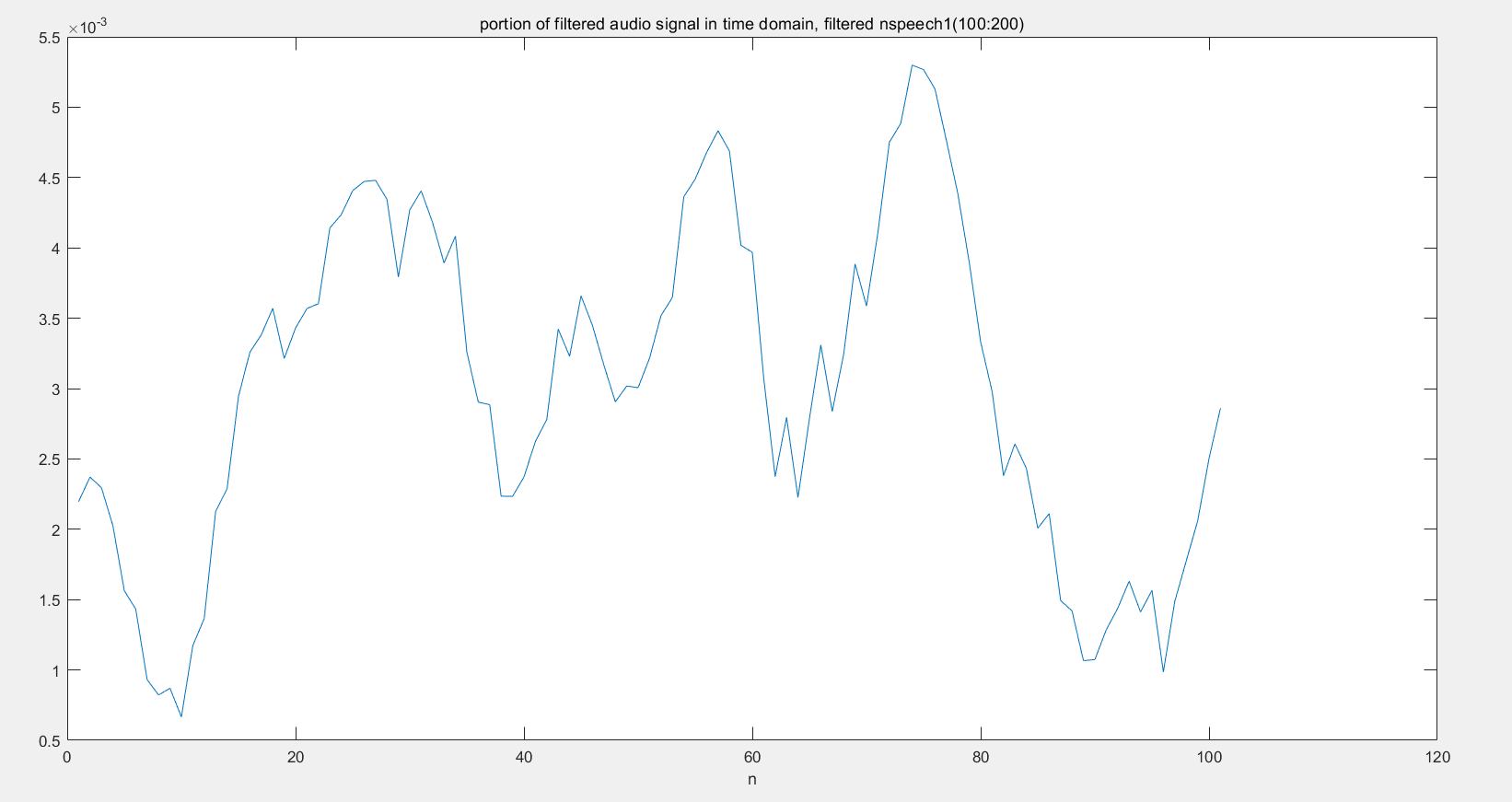


Figure 7.3.6

It seems much better than figure 7.3.3. Now we can see the waveform of the audio signal

The plot of the magnitude of the DTFT for 1001 samples filtered signal with range (100:1100) is shown in the figure 7.3.7. Here the length of DTFT is also set to

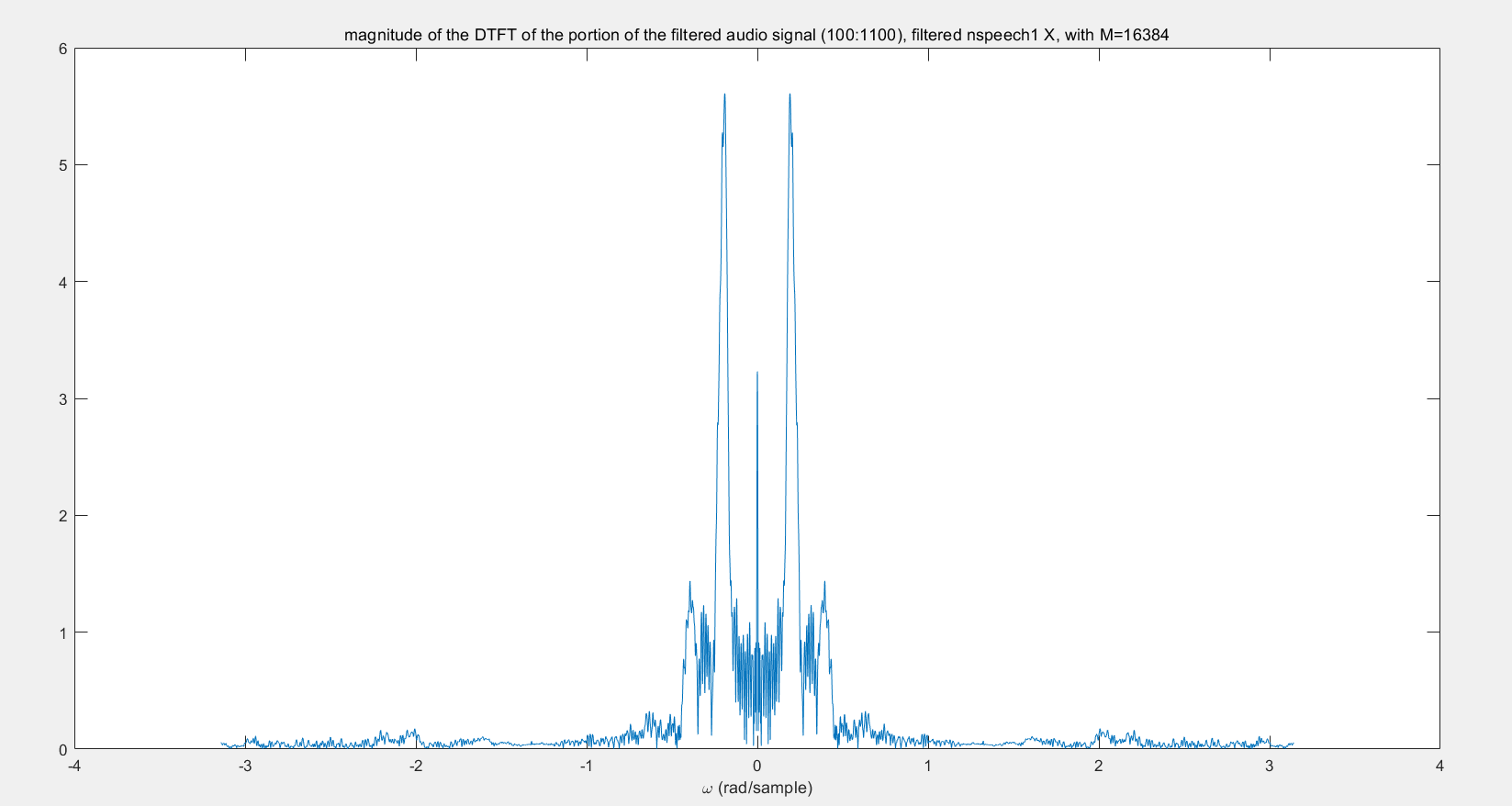


Figure 7.3.7

It shows that the sinusoidal signal has been removed (though, we cannot get it to zero because the estimated is not really ‘exact’, we just to get this sinusoidal part to be trivial). The magnitude of the DTFT of the audio signal finally can be clearly seen.

However, out filter is not that ideal, it will also get the voice signal distorted. From the frequency response of the filter we know that the frequency that closer to the stop frequency , the more attenuation it will get, vise versa.

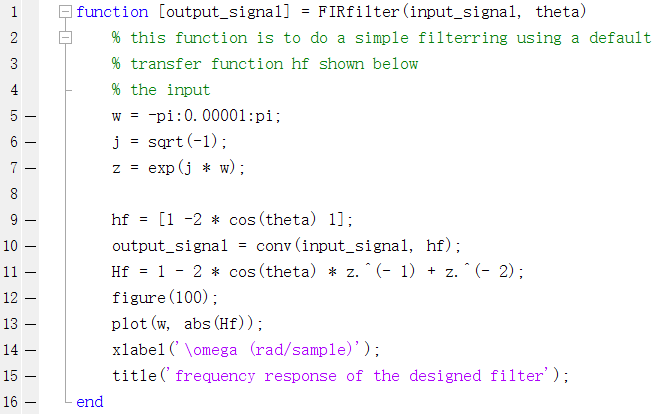
Thus the filter reduces the intense disturbing sinusoidal signal to make the voice can be heard and get the voice distorted, but the distortion is tolerable.

The filter get the intensity of some frequency to zero, and the frequency is at middle, the filter is a band-stop filter.

In the end, the sound test also get passed and we can clearly hear the voice.

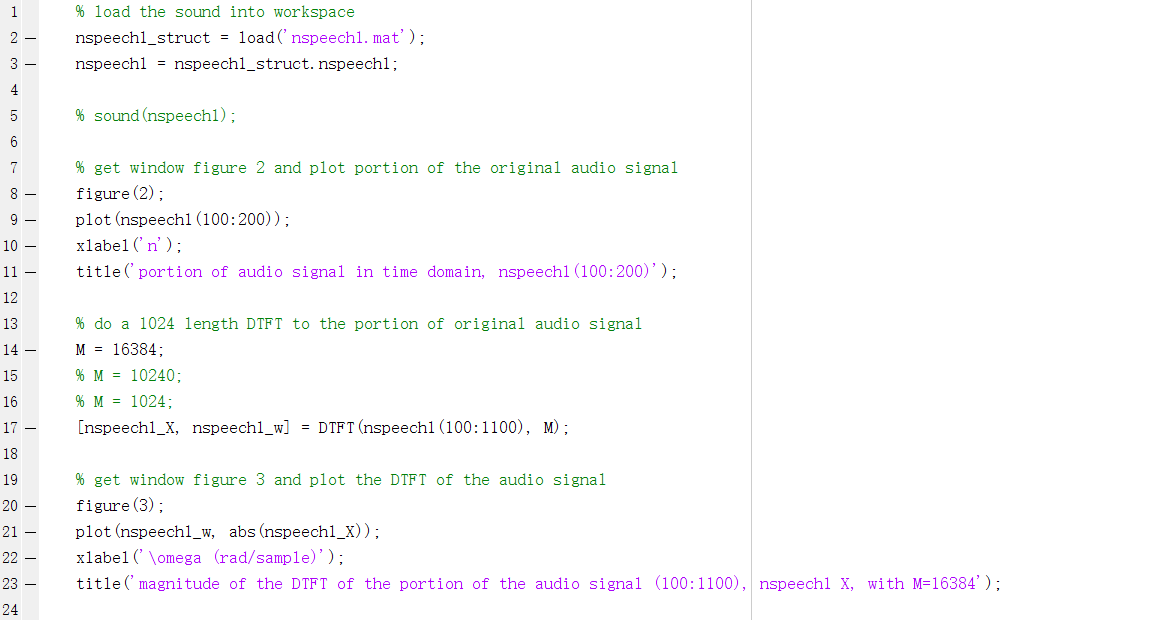
The voice is *“please get rid of this beep”*.

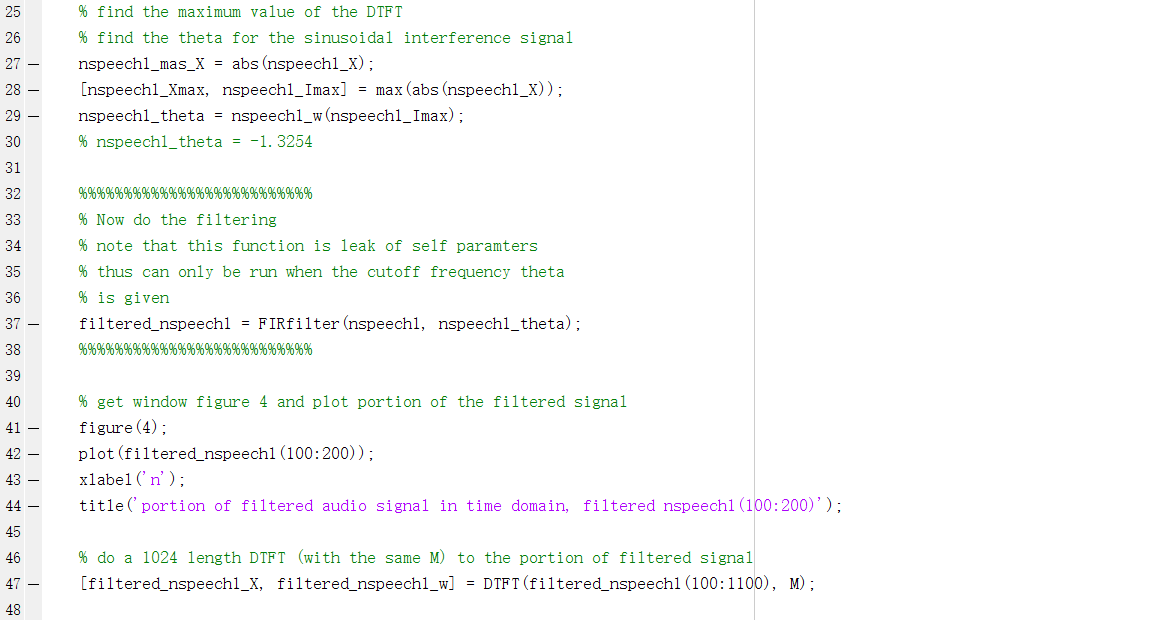
The code for function FIRfilter is shown as code 7.3.2 (corresponding to the file FIRfilter.m)

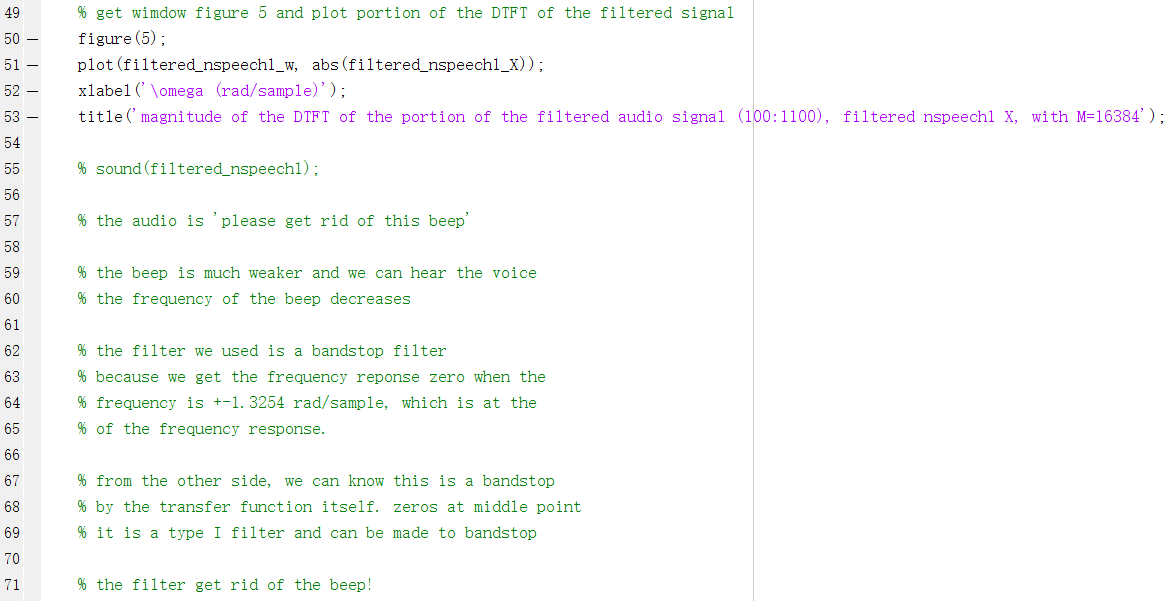


Code 7.3.2

The code for this script is shown as code 7.3.3 (corresponding to the file lab7\_3\_2.m)







Code 7.3.3

Another note in this section.

First I set the length of the DTFT to 1024, and find that the filtered signal is not good as expected, there still some remaining sinusoidal signal in the magnitude of the DTFT as figure 7.3.8 shown below

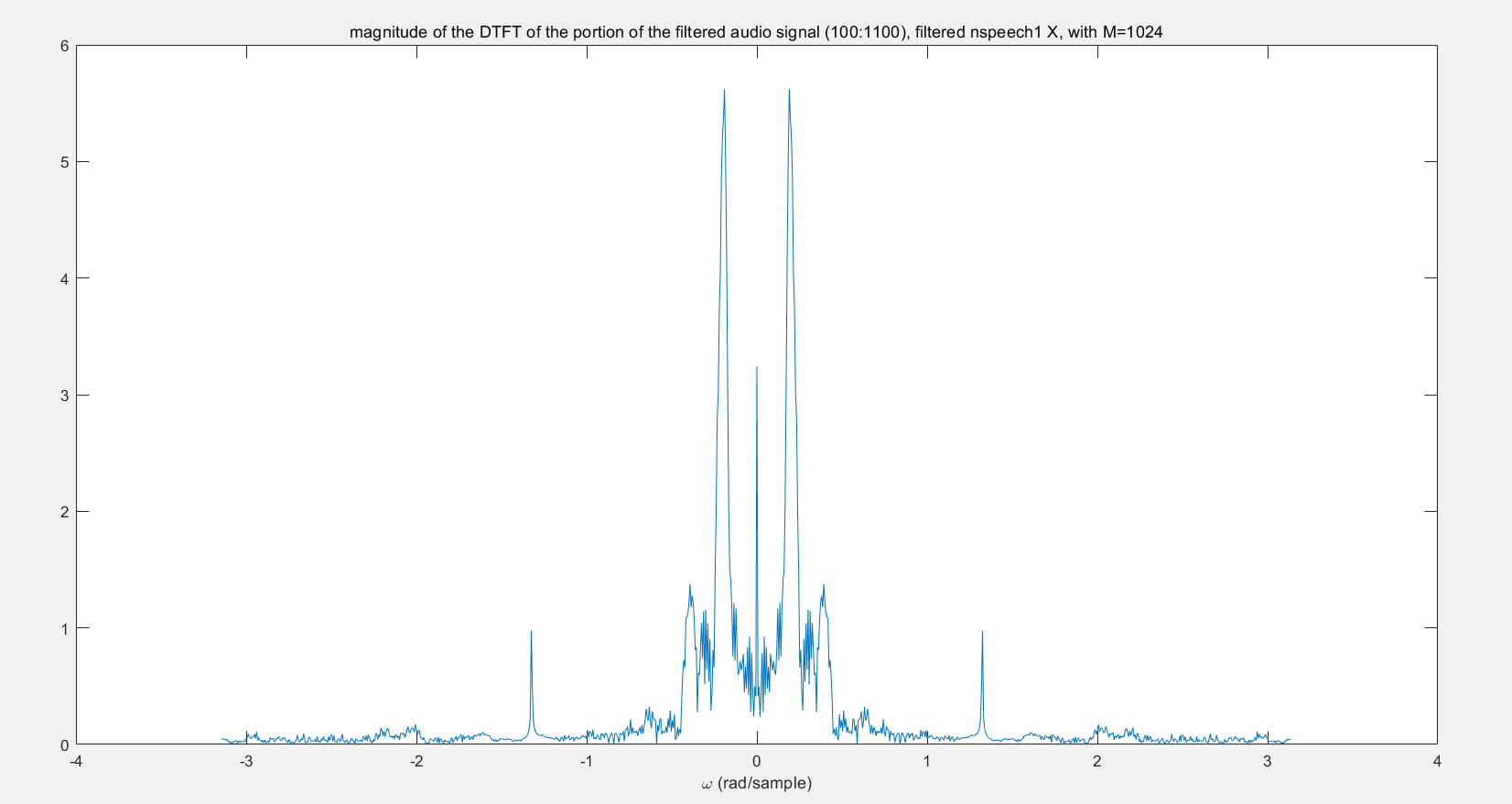


Figure 7.3.8

This is caused by an error in the estimation in , the figure below shows the value of that calculated by matlab by using ‘max’ function to find.



Figure 7.3.9

Which is a little away from our result when DTFT length is 16384



Figure 7.3.10

This shows the importance of enough length of DTFT when analysis the frequencies.

**7.4 Design of a Simple IIR Filter**

From the section above we know that zeros attenuate a filtered signal, and in this section, we can see that poles can amplify signals that are near their frequency.

We can get a normal form of these simple IIR filters from its properties. These simple IIR filter we use here has only poles and zeros only at the origin (to get the input of the filter).

Thus the impulse response of it is

Equation 7.10

And implement the z-transform to the both side of the equation above, we get

Equation 7.11

**7.4.1 Second Order IIR Filter**

In this section, we will use a simple second order IIR filter with two poles inside the unit circle to illustrate the use of poles in filter design. In order for the filter’s impulse response to be real-valued and casual, the two poles must be complex conjugates (or symmetry about the Re-axis in z-plane) of one another:

Where is the angle of relative to the positive axis and .

Thus the transfer function for this filter is

Equation 7.12

Similarly as what have been done in section 7.3.1, we can get the analytical expression of the impulse response for the filter is

Equation 7.13

And replace the impulse function with input and with , we get the difference equation of the filter

Equation 7.14

From the equation above we can get a version of the system diagram of the filter. This system diagram is shown in figure 7.4.1

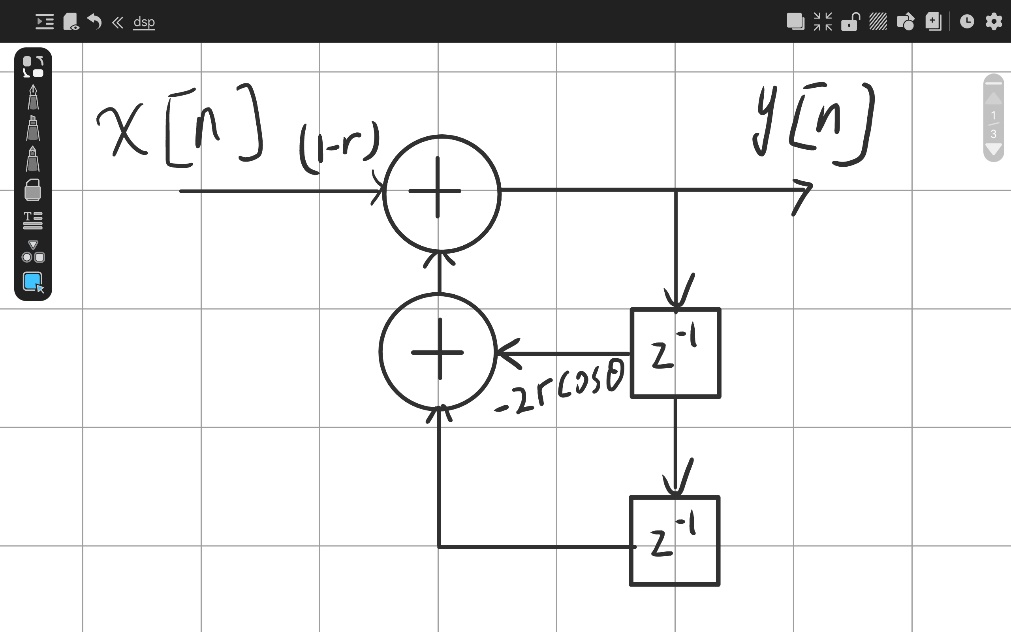


Figure 7.4.1

We choose several values of , which is , but in a fixed , to check how the magnitude of the poles will affect the bandwidth of the band-pass filter.

The figure 7.4.2 below shows the magnitude response of the filter when equals to the three values above and .

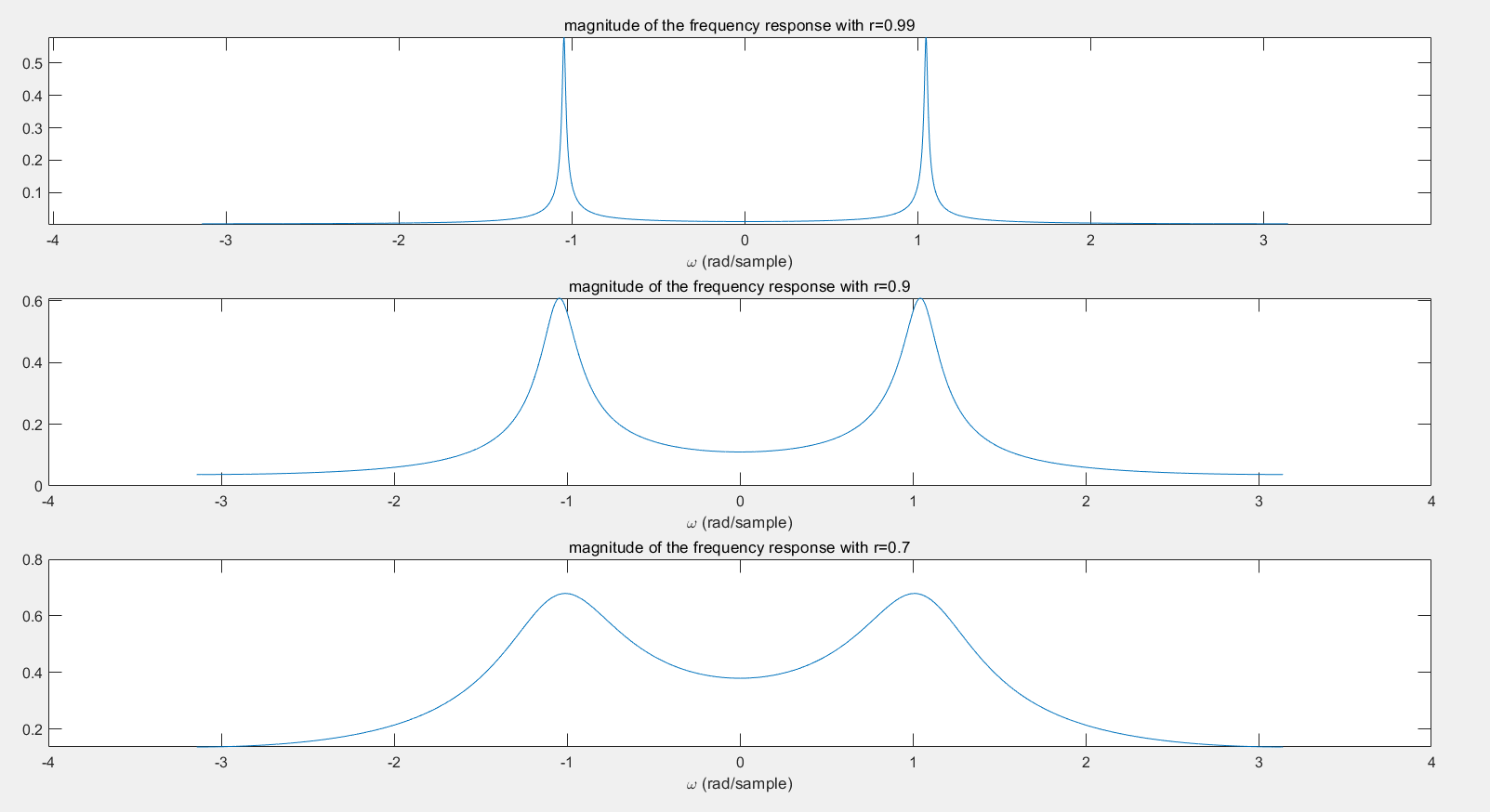


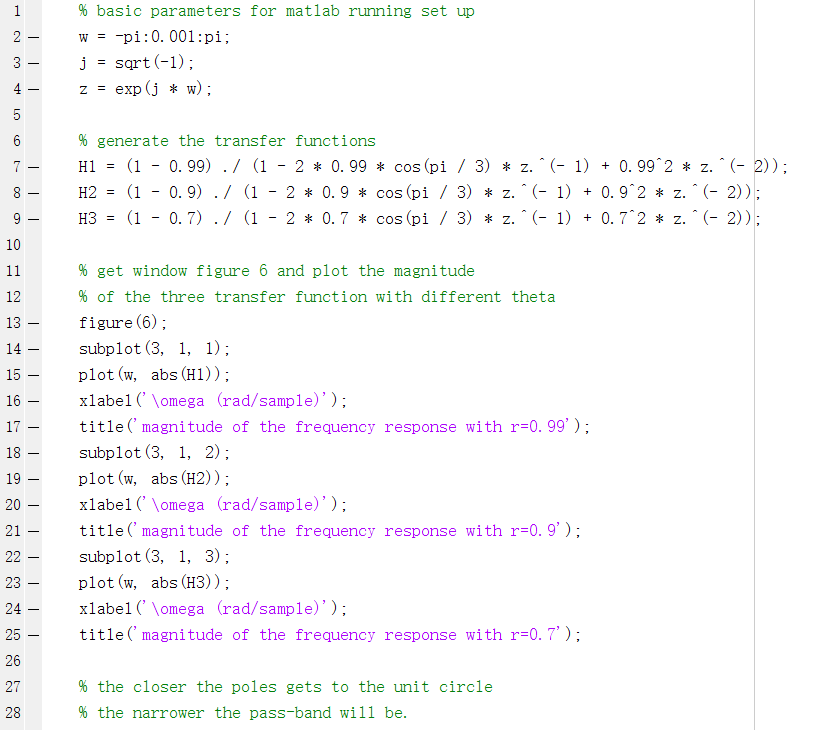
Figure 7.4.2

The figure shows that determines the width of the passband. When get closer to the unit circle, the band of this bandpass filter will have a narrower band around the desired frequency , also with other frequency getting more attenuated.

This is because actually the poles indicates the infinite, the closer to the pole, the larger the magnitude will be. And we use a parameter in numerator to get whole response normalized, thus this will widen the gap between the frequency near the poles and away from the poles.

In another way, when poles gets closer to the unit circle, the distance change between the frequency near the poles and away from the poles will get larger (obvious in ratio, especially).

And the code for this script is shown as code 7.4.1 (corresponding to the file lab\_7\_4\_1.m)



Code 7.4.1

**7.4.2 Filtering the Audio Signal**

In this section, we use the filter above to filter an audio signal with background noise in it. The target sound is a modulated sinusoid. Our target is to separate it from the background noise.

The time domain plot of 101 samples with range (100:200) is shown in the figure 7.4.3

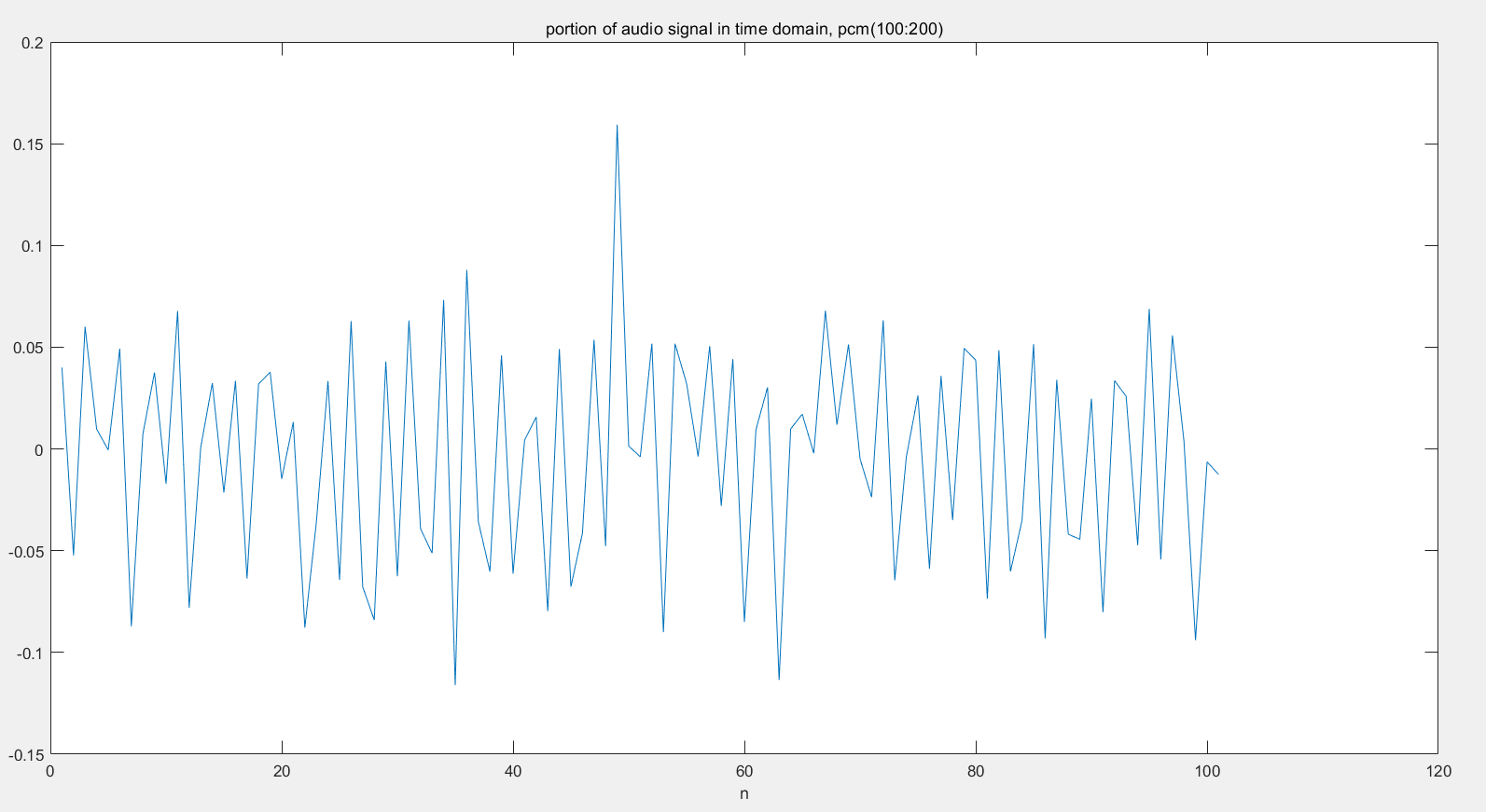


Figure 7.4.3

The figure shows that there seems a sinusoid signal but there is some noise inside it. It is hard to distinguish.

The plot of the magnitude of the DTFT for 1001 samples with range (100:1100) is shown in the figure 7.4.4. Here the length of DTFT is set to

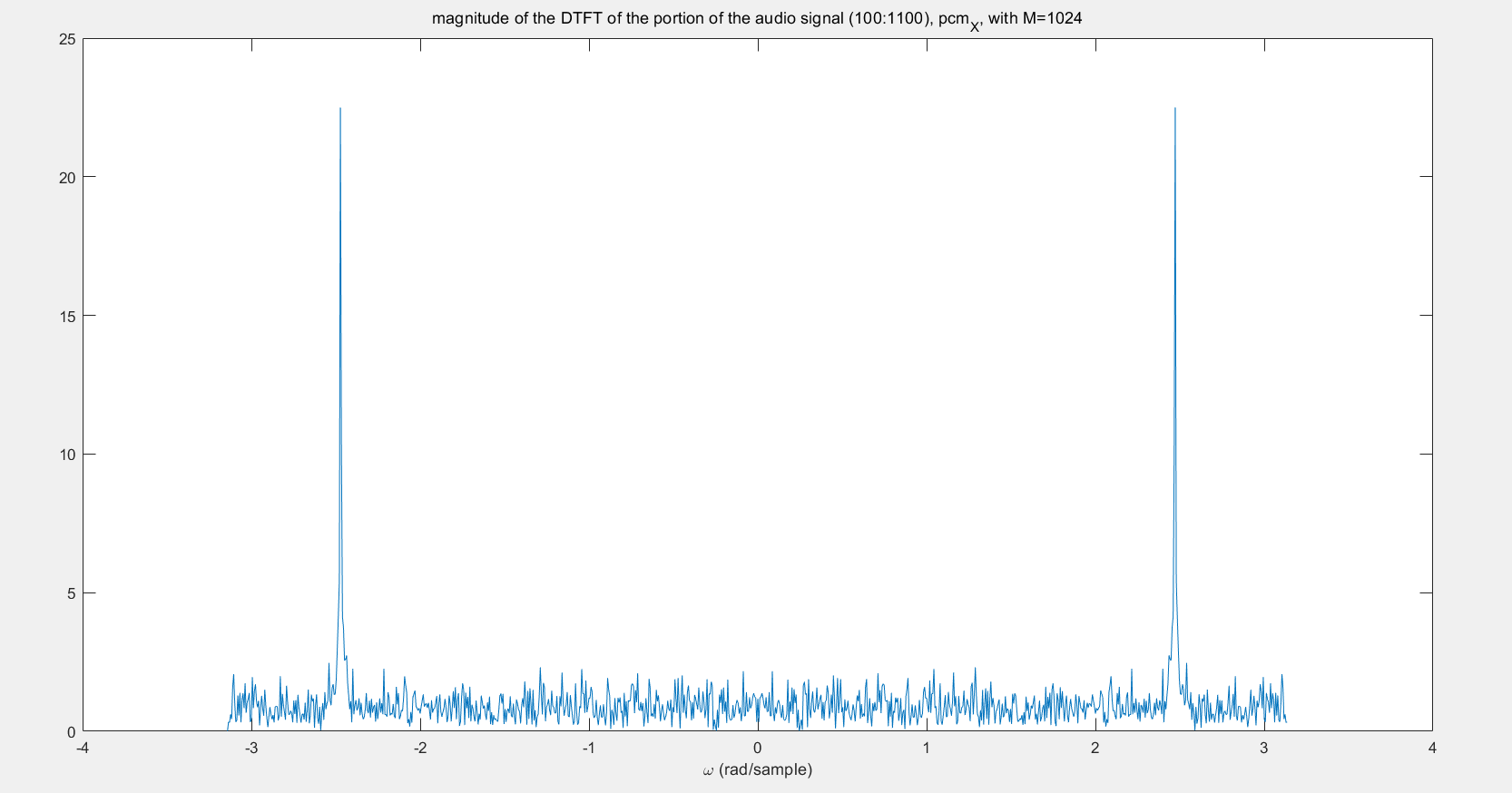


Figure 7.4.4

We can see that there are two intense sinusoid signal, but also there are some noise with average magnitude around 1 to 2. Though it is small but all the frequency has these noise.

Because we already know the specification of the filter, we just do the filtering and get the result.

The time domain plot of 101 sample filtered signal with range (100:200) is shown in the figure 7.4.5

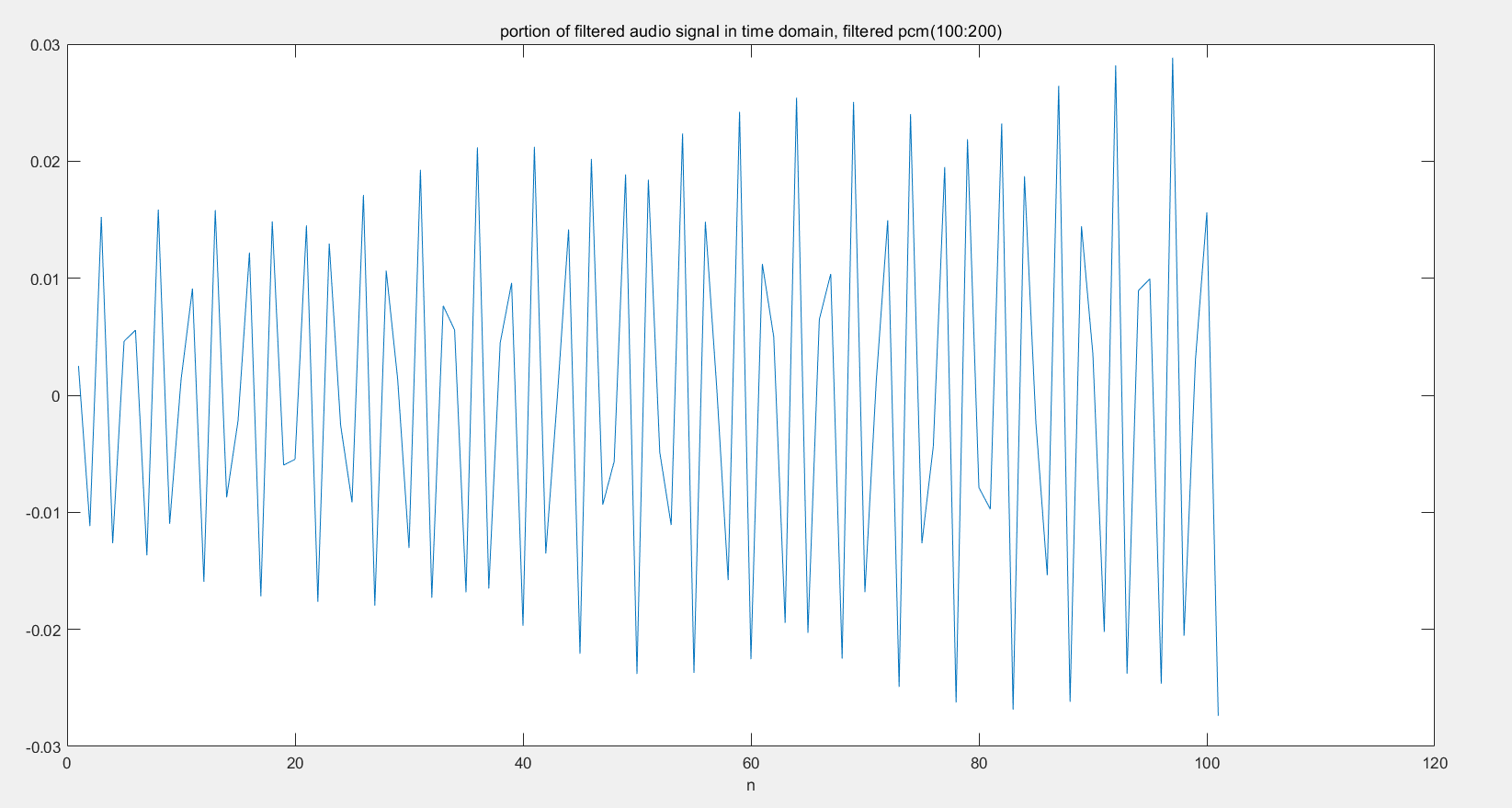


Figure 7.4.5

The sinusoid is clearer.

And the plot of the magnitude of the DTFT for 1001 samples filtered signal with range (100:1100) is shown in the figure 7.4.6. Here the length of DTFT is set to

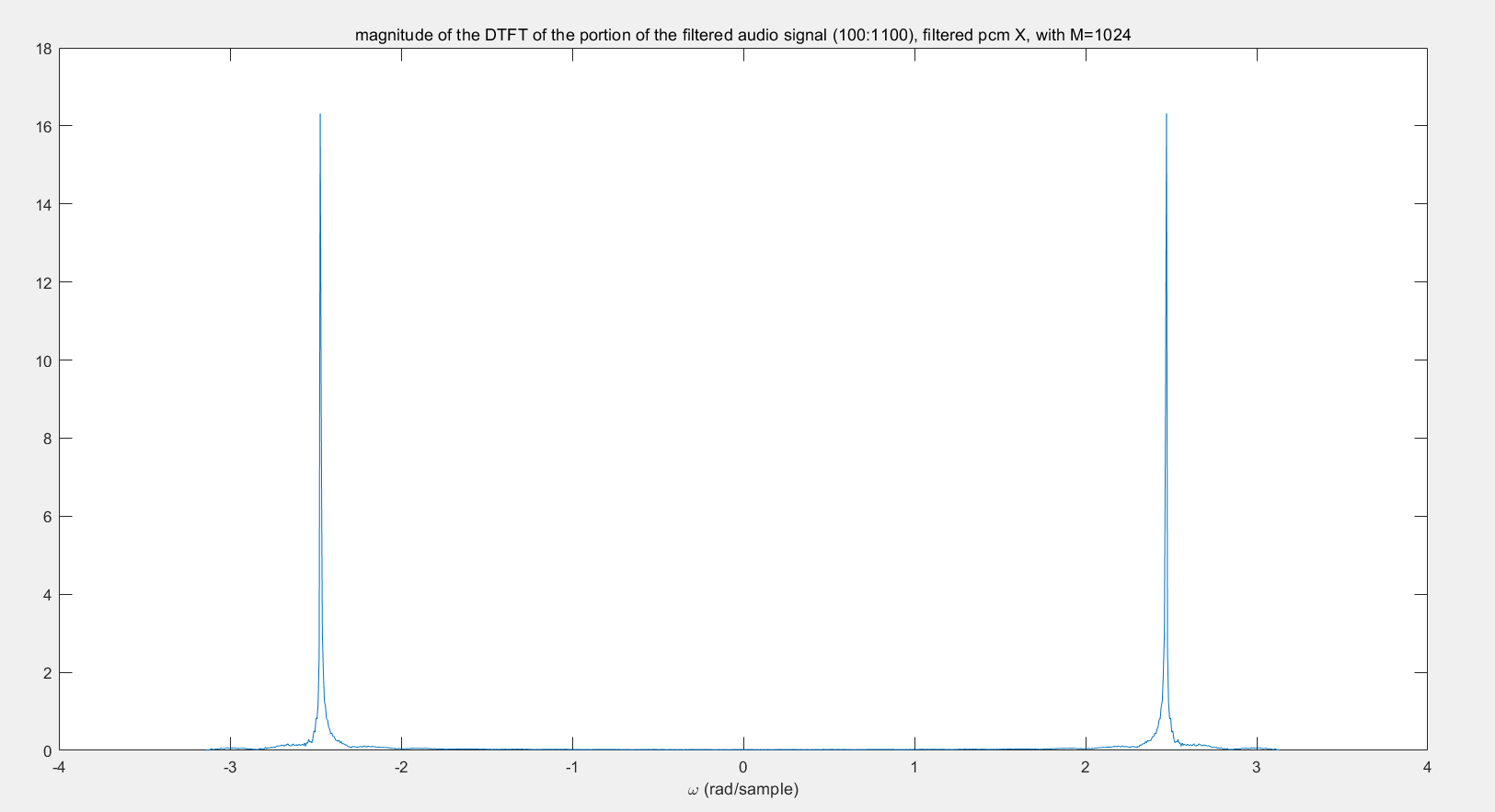


Figure 7.4.6

We can see that the noise has been remove. (Of course our filter is an ideal filter thus the ‘remove’ here means we can consider the noise to be trivial)

And the plot of the magnitude of the DTFT of both original signal and filtered signal for in the range is shown in figure 7.4.7

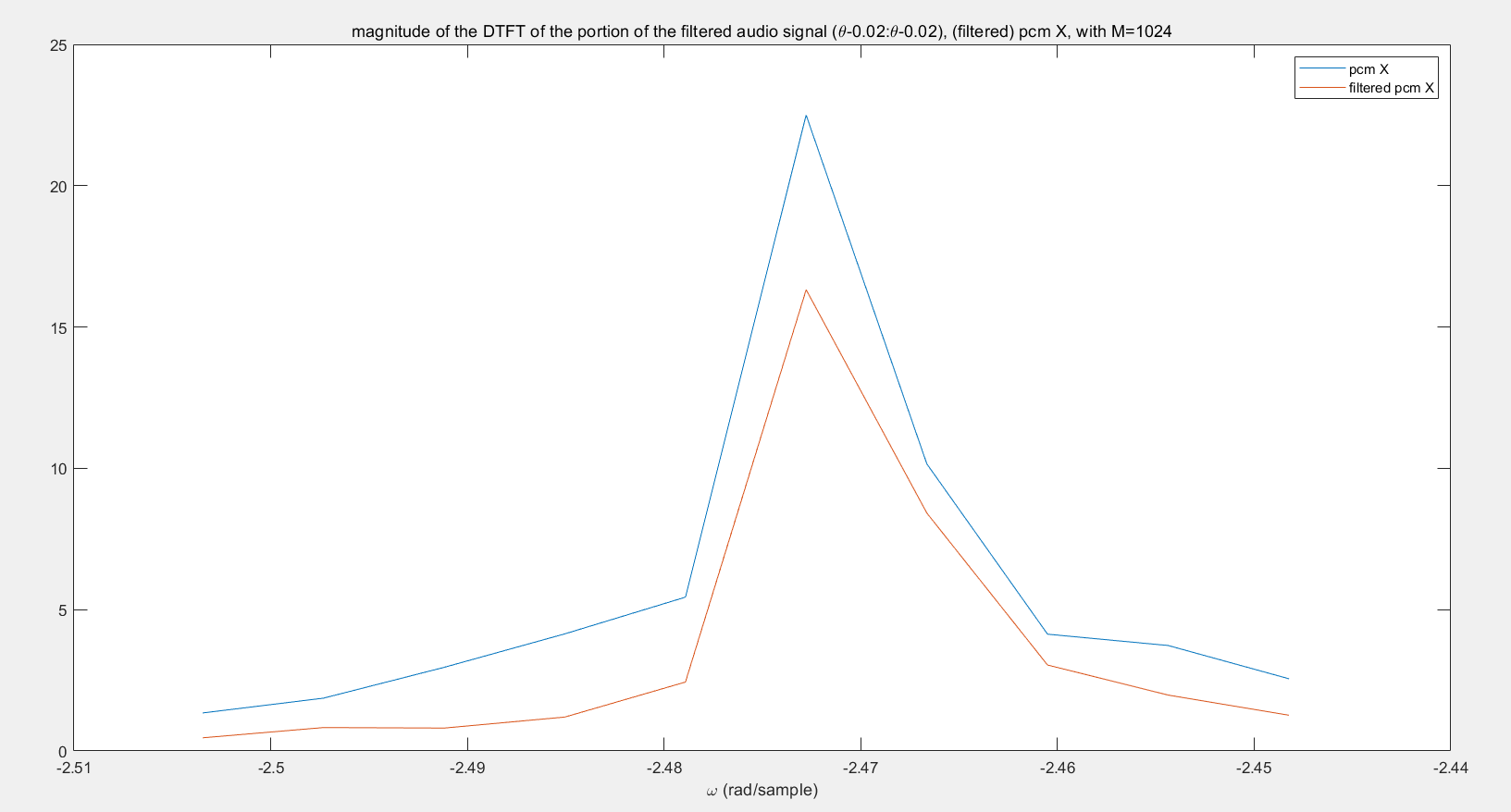


Figure 7.4.7

The frequency here also gets a little bit attenuated.

The sound test also shows that after filtering we can clearly hear the modulated sinusoidal signal and hardly heard the noise. Before filtering though we can hear the sinusoidal signal but the background noise is really annoying.

After change , the time domain plot of 101 sample filtered signal with range (100:200) is shown in the figure 7.4.8

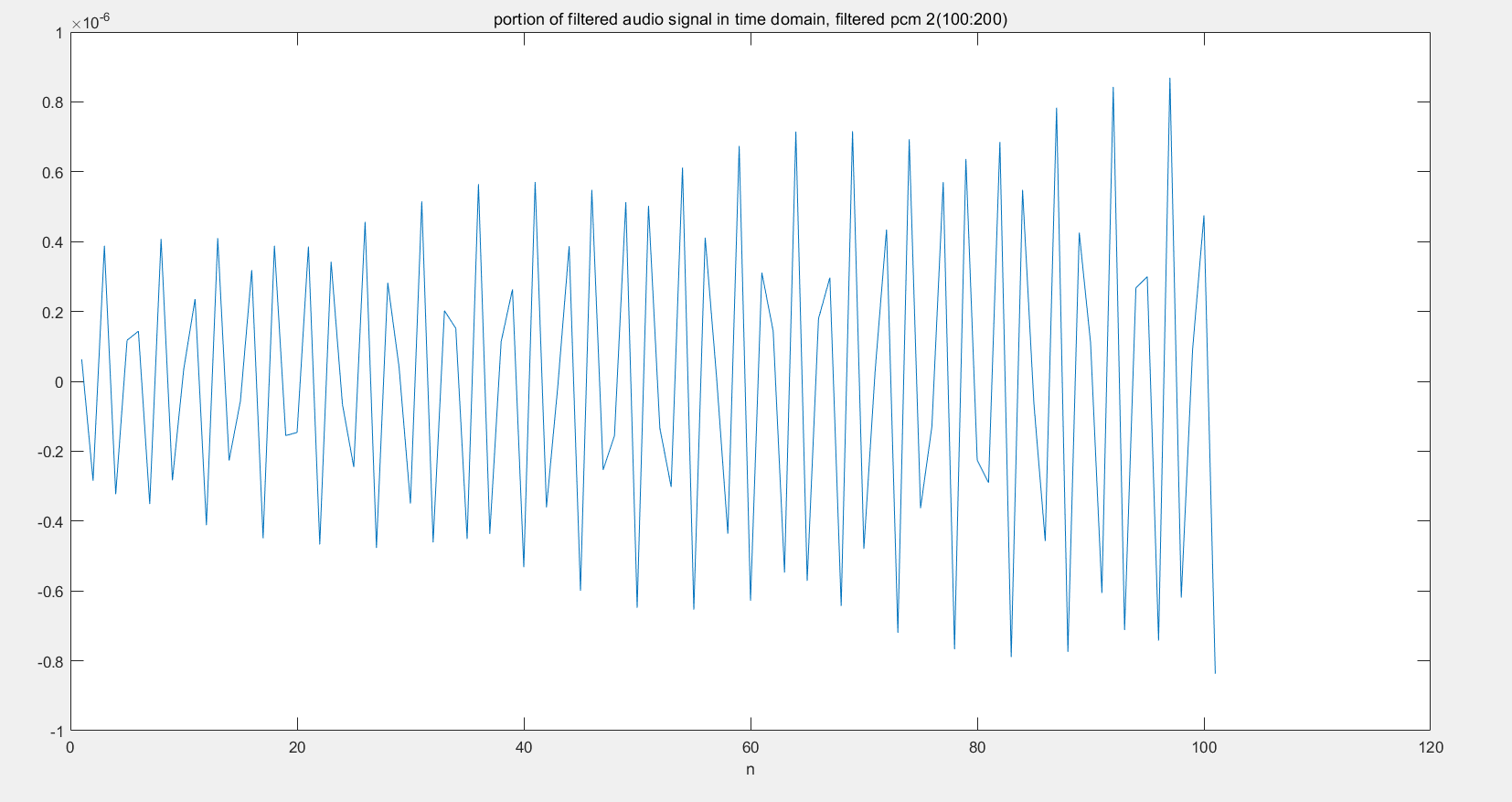


Figure 7.4.8

It is pretty similar to the one in figure 7.4.5. The main different is that the magnitude here is too small.

And the plot of the magnitude of the DTFT for 1001 samples filtered signal with range (100:1100) is shown in the figure 7.4.9. Here the length of DTFT is set to

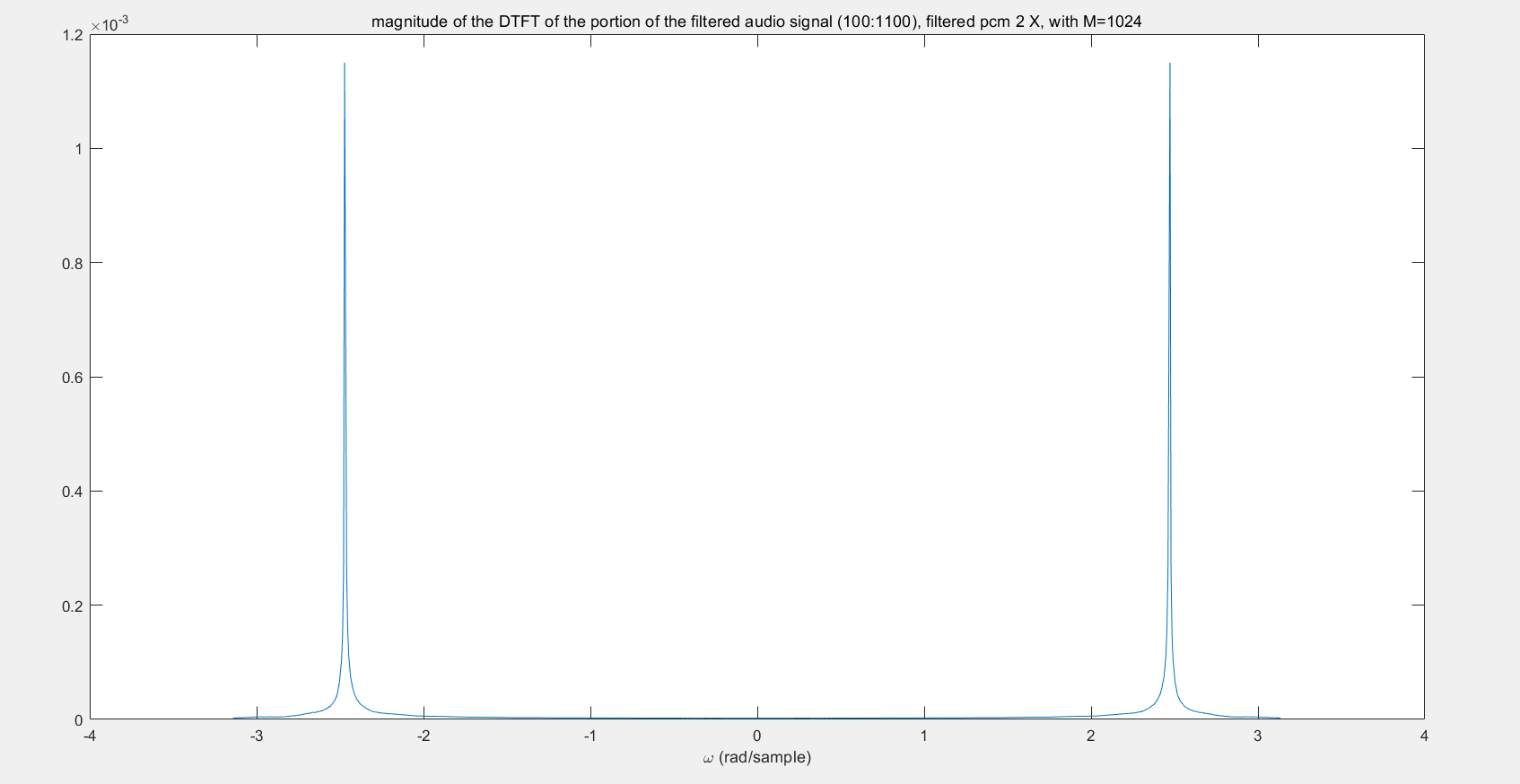


Figure 7.4.9

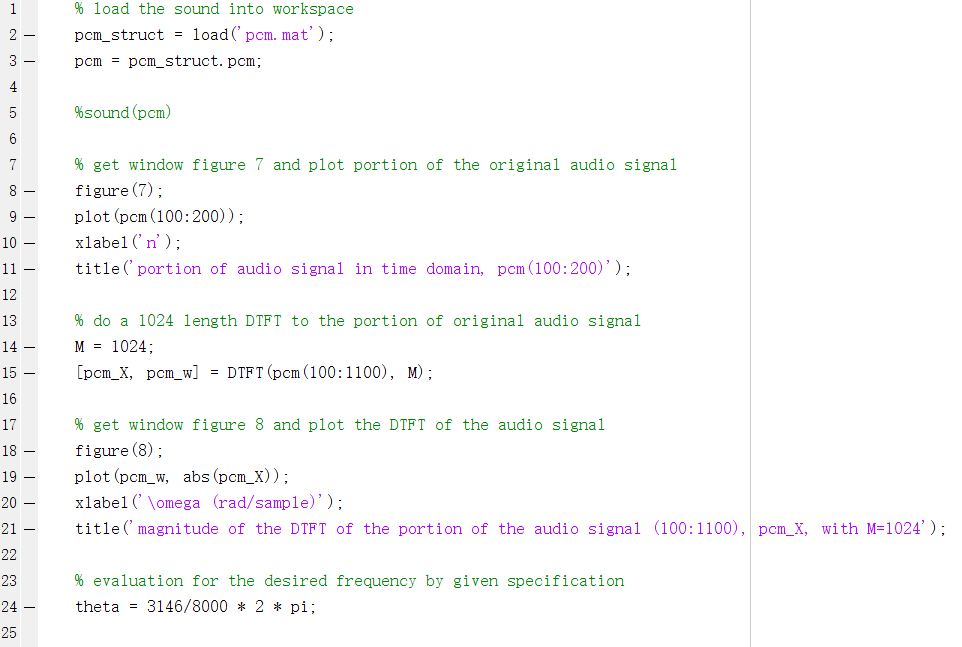
Same as the description above. The magnitude is too small.

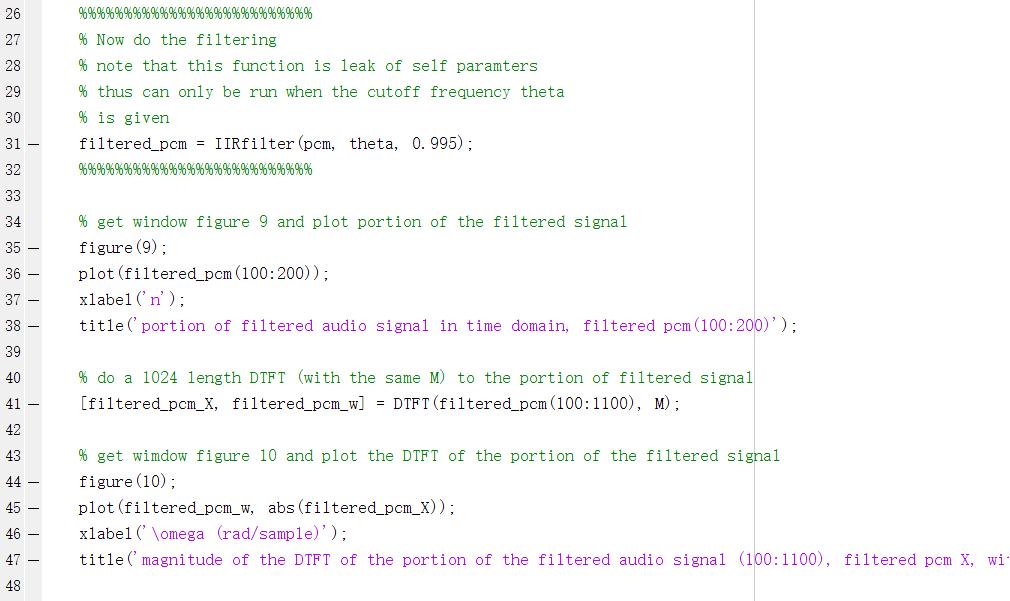
Though we can use amplifier to all frequency to get the magnitude larger, the recovered signal is still not good, this may because the band is too narrow and get the modulated signal distorted.

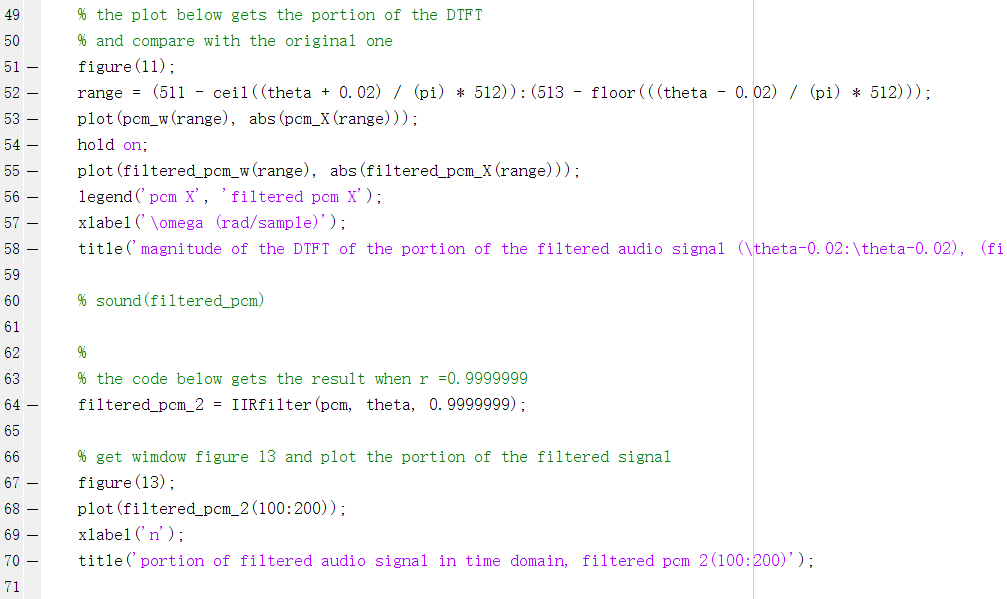
Additionally in application, such filter will consume large amount of energy, which is not good at all.

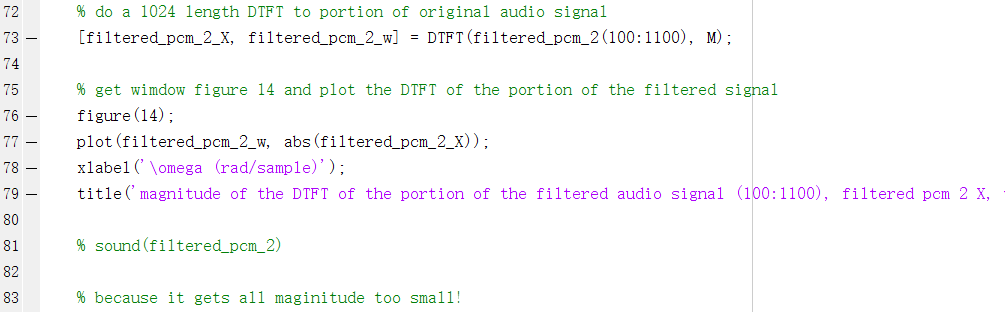
This also shows that change in r will influence the magnitude overall and the bandwidth of the filter. In general, larger r, but still less than 1, will gets less magnitude and narrower bandwidth.

The code for this script is shown as code 7.4.2 (corresponding to the file lab\_7\_4\_2.m)



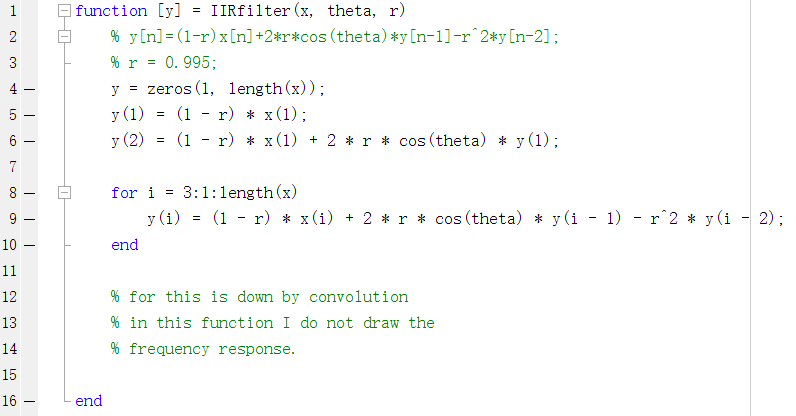






Code 7.4.2

And the code for function IIRfilter is shown as code 7.4.3 (corresponding to the file FIRfilter.m)



Code 7.4.3

**7.5 Filter Design Using Truncation**

In the previous sections we use property of zeros and poles in z-transform to design filters. Clearly we only design second order filter and do some simple filtering. If the signal gets more complicated and need larger order of the filter with more strict specification, it is hard to design the correct parameters for the filter especially when the order is too large.

Thus another way to design filter is using truncation function.

We first use specification to get an ideal filter, it has infinite impulse response thus it is not implementable. Then using truncation function to get part of the impulse response in time domain, which is finite, thus we can implement this reduced filter.

In this section we only use the simplest truncation, the rectangle truncation function, to design a low pass filter.

An ideal low-pass filter has frequency response as below

Equation 15

And use inverse DTFT we get the corresponding impulse response is

Equation 16

For

And after truncating the impulse response, we get

Equation 17

To get this filter casual, we shift the impulse response in time domain. We get our final filter

Equation 18

With final length

The figure 7.5.1 and 7.5.2 below shows the magnitude response for the two filters, where the former one shows the one with length 21.

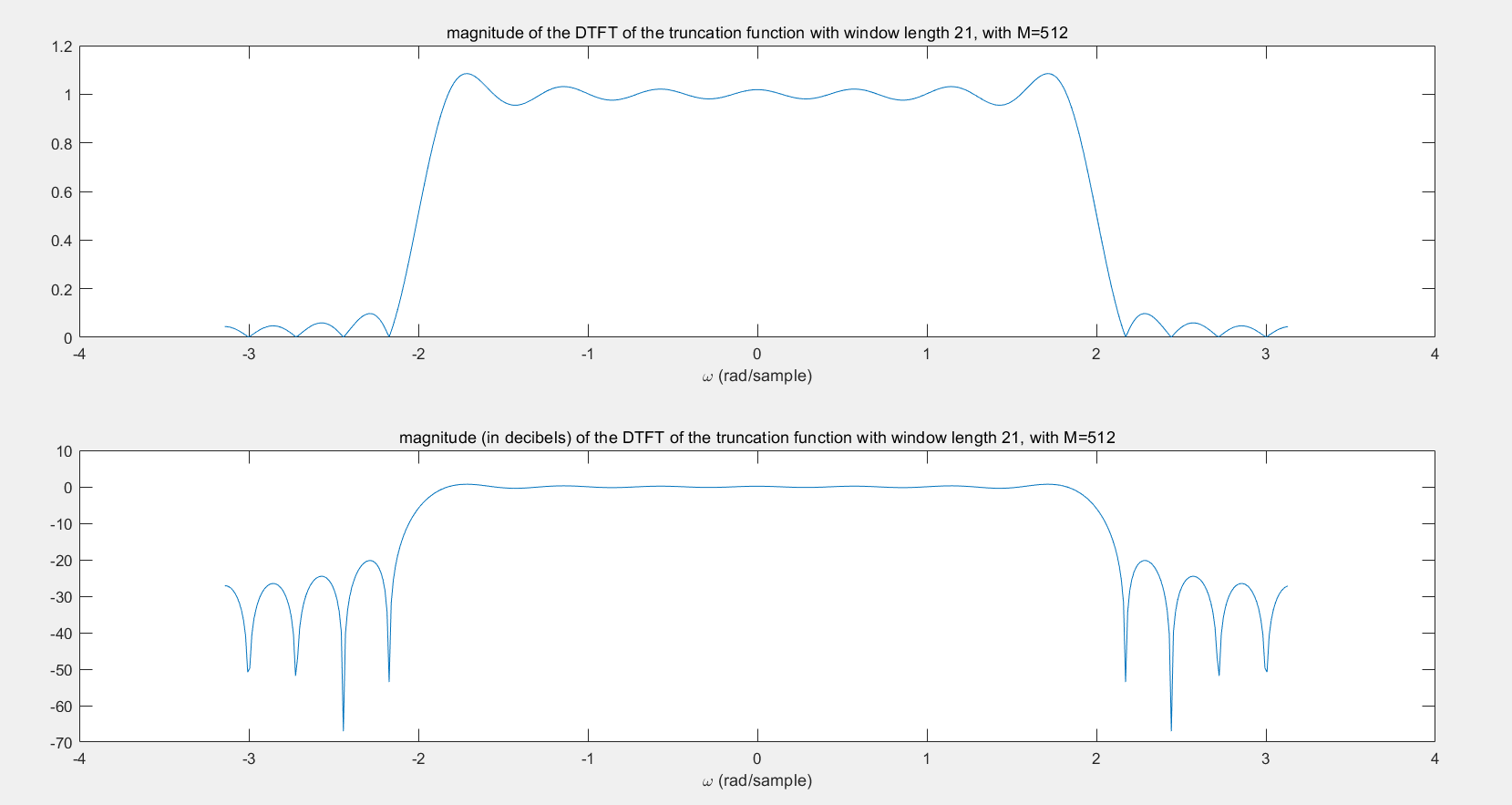


Figure 7.5.1

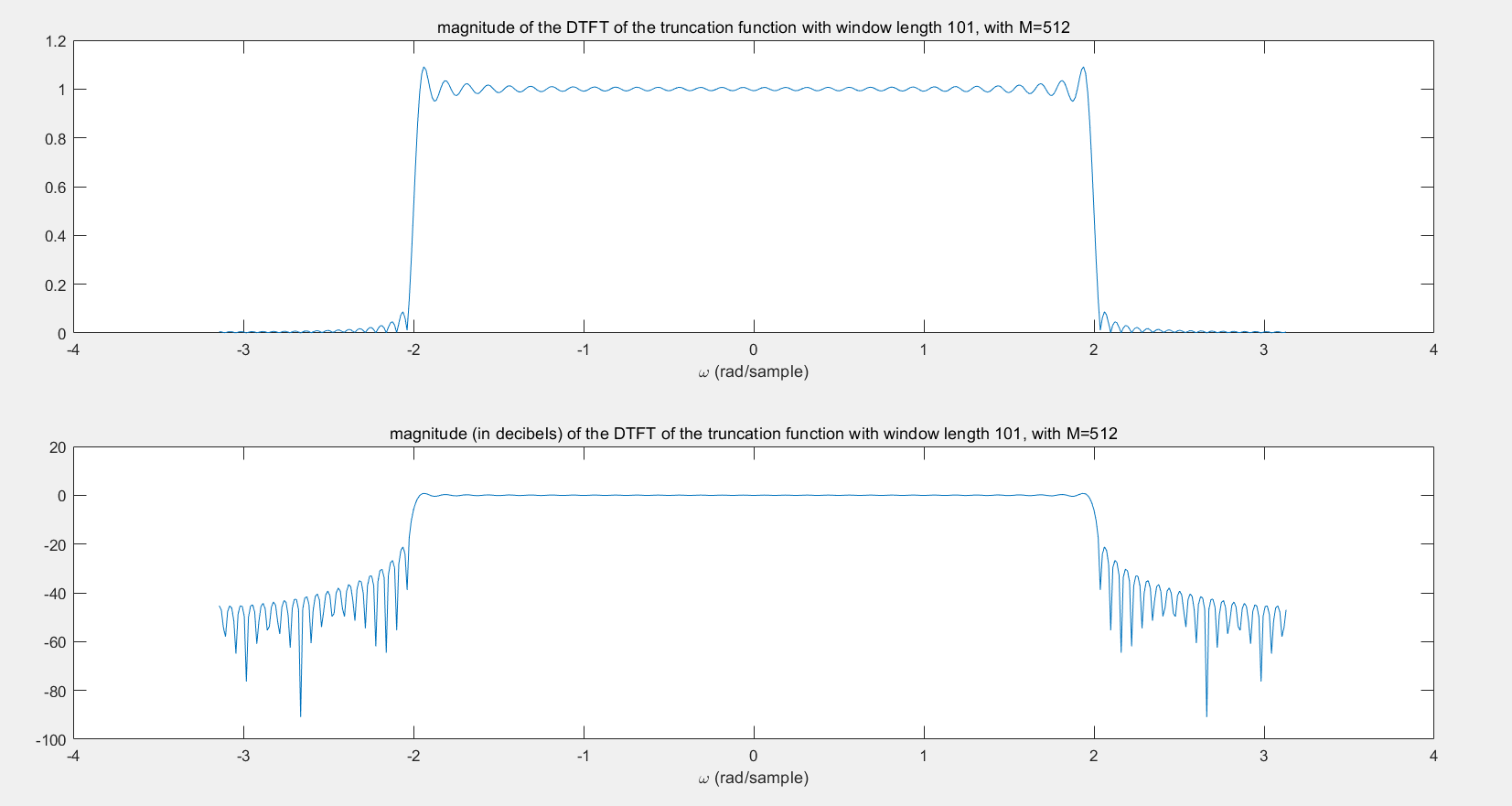


Figure 7.5.2

Note the green rectangle indicates the pass-band, the red rectangle indicates the transition-band, and the purple rectangle indicates the stop-band.

It shows that larger length of the truncation function (equal to larger length of the filter) will not influence the ripple in pass-band due to Gibbs phenomenon. But larger length of the filter will get a smaller transition band.

Larger filter length will get more ripples. ? check ?

We can get why this happen through convolution. The process we do in time domain is use the ideal filter to multiply the truncation function, thus it will cause the convolution in frequency domain. For larger length of truncation function (also means larger length of filter) will include more ripple of the ideal function while do the convolution, this is why there are more ripples.

The figure 7.5.3 shows the plot of original signal nspeech2.

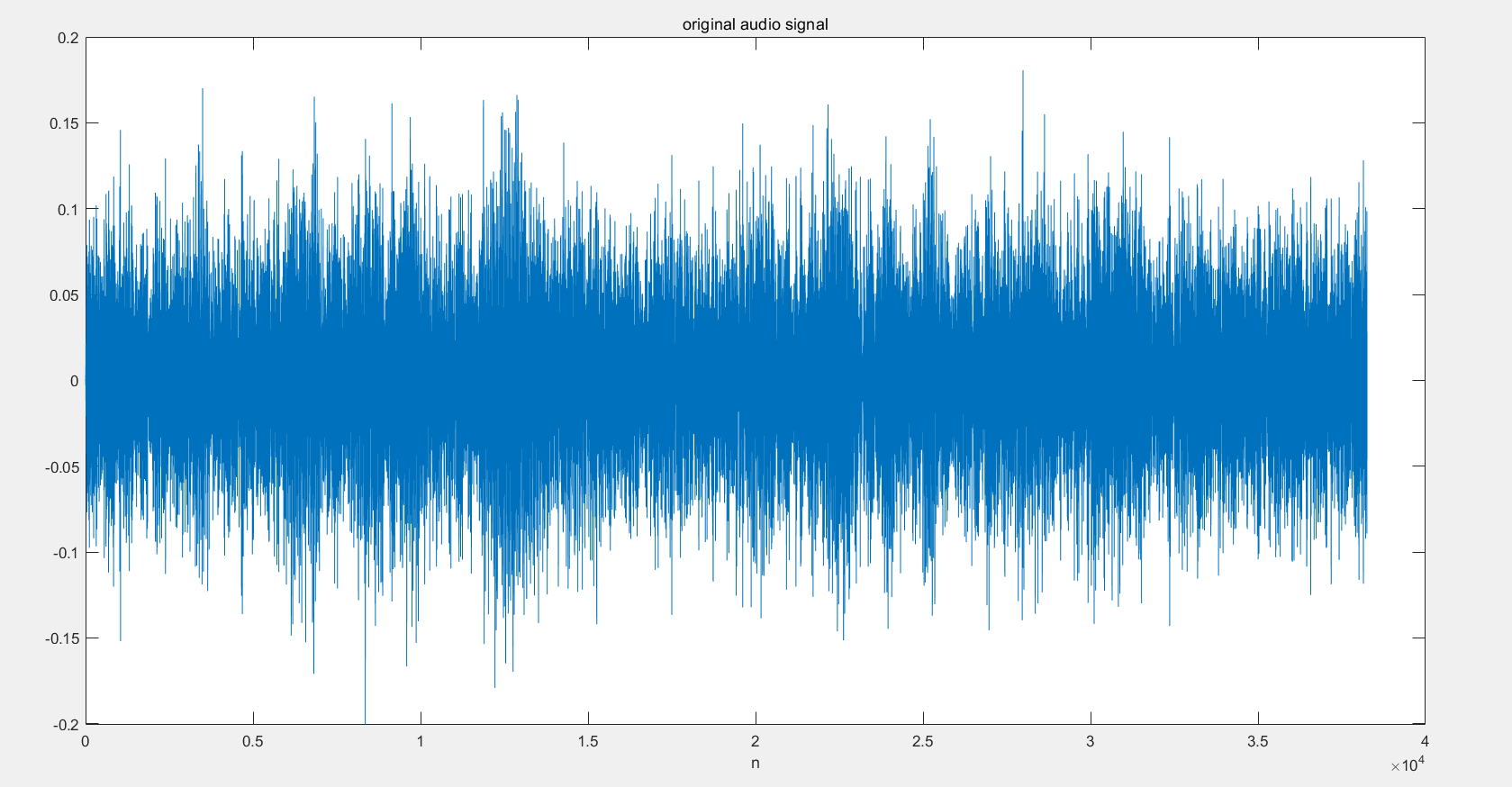


Figure 7.5.3

And the figure 7.5.4 shows the magnitude of the DTFT of the original signal.

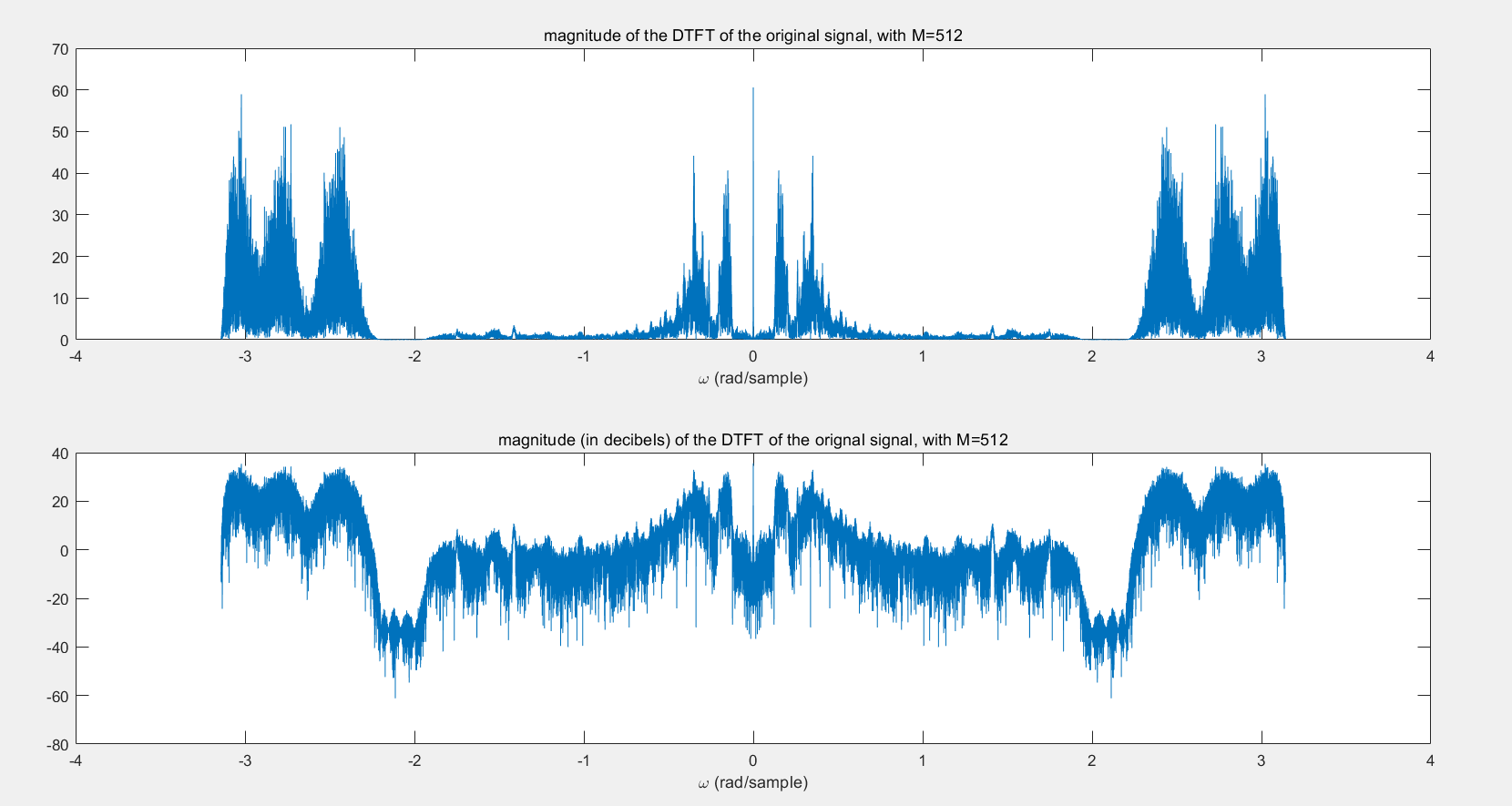


Figure 7.5.4

We can see from it that there are some noise in the high frequency part. And its intensity seems high enough to cover the original signal.

After the signal is filtered by the 21 length filter, the filtered signal is shown in figure 7.5.5

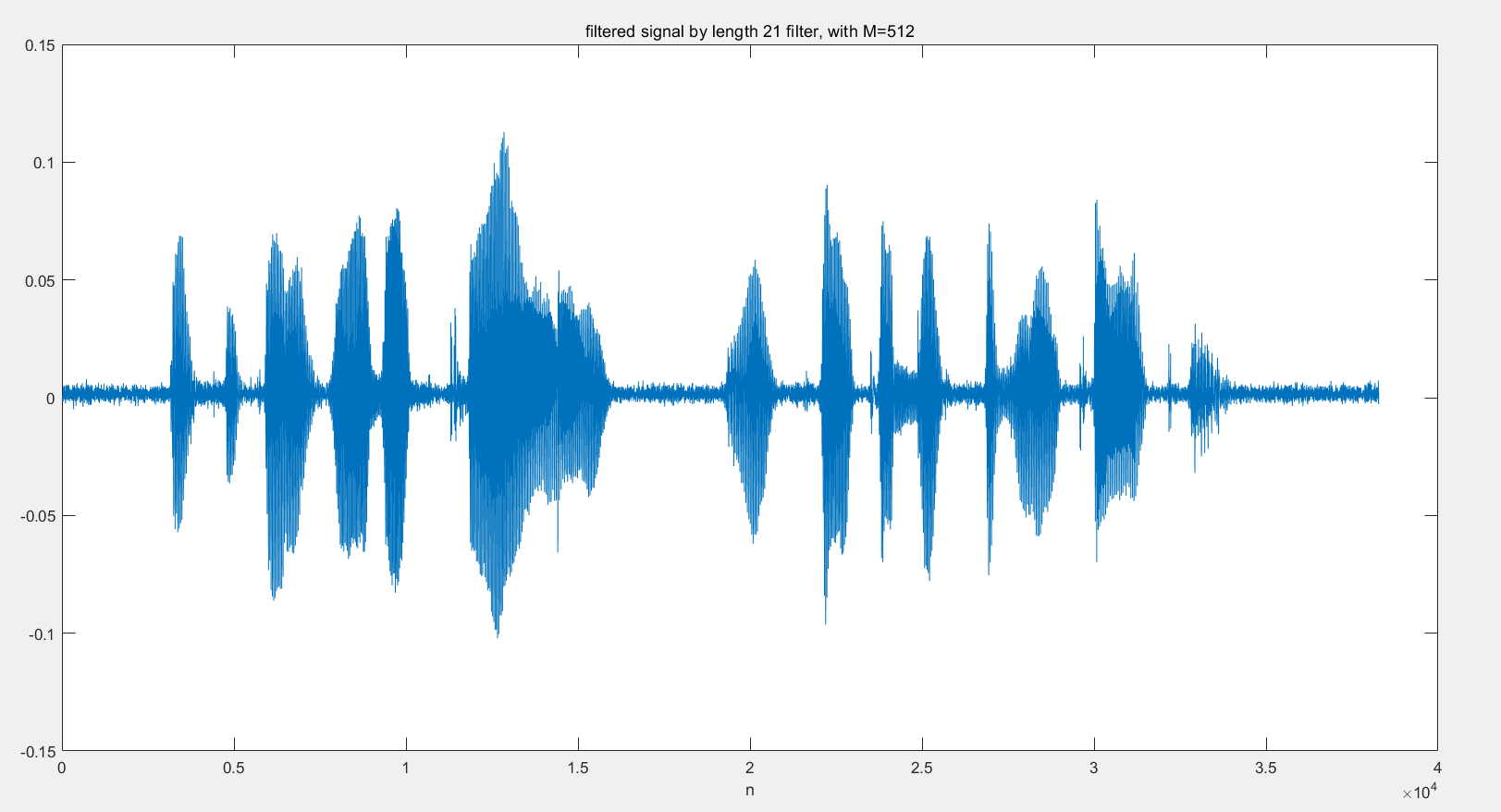


Figure 7.5.5

And the figure 7.5.6 shows the magnitude of the DTFT of the original signal.

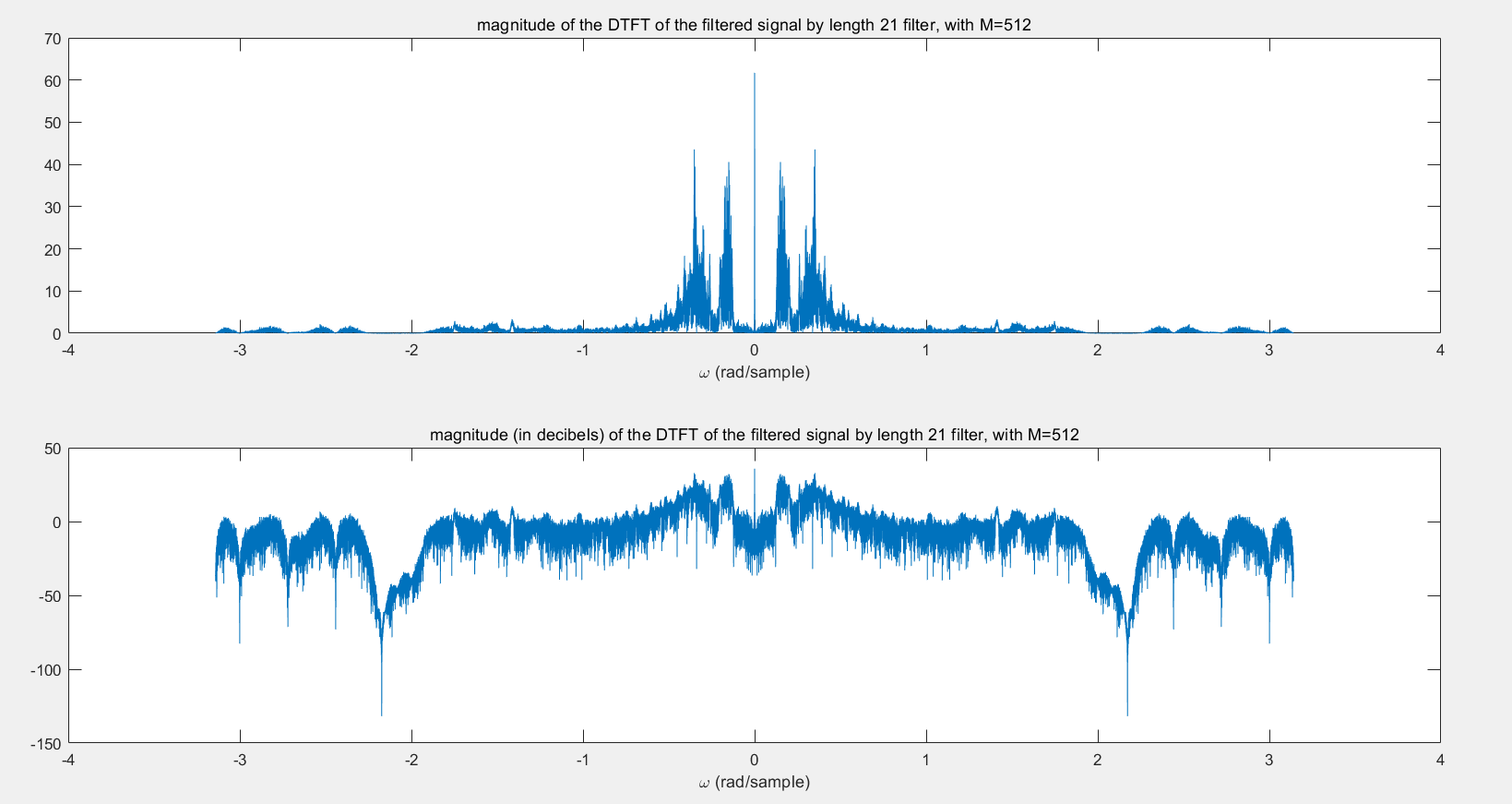


Figure 7.5.6

Similarly, after filtered by the 101 length filter, the filtered signal is shown in the figure 7.5.7

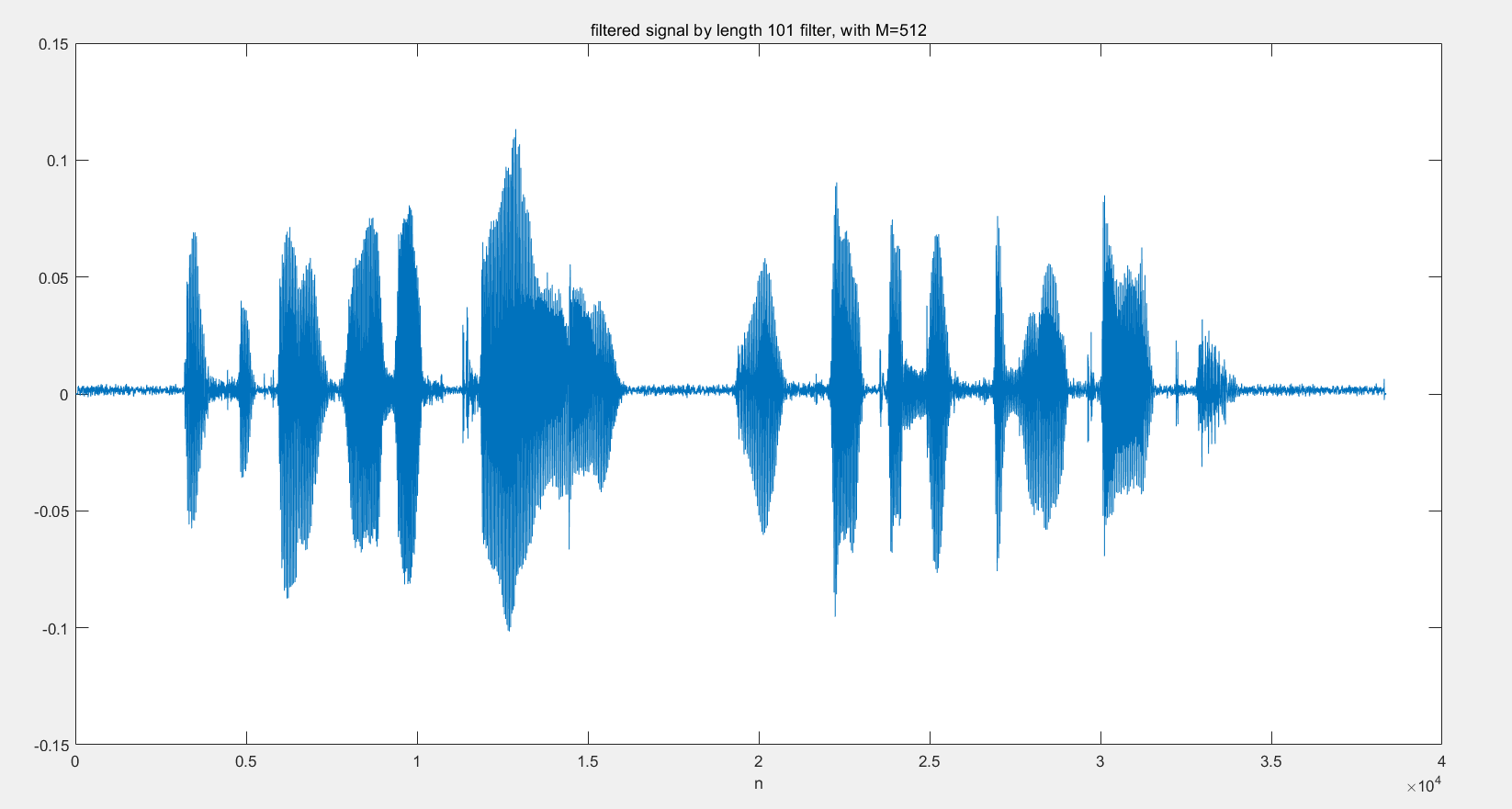


Figure 7.5.7

And the figure 7.5.8 shows the magnitude of the DTFT of the original signal.

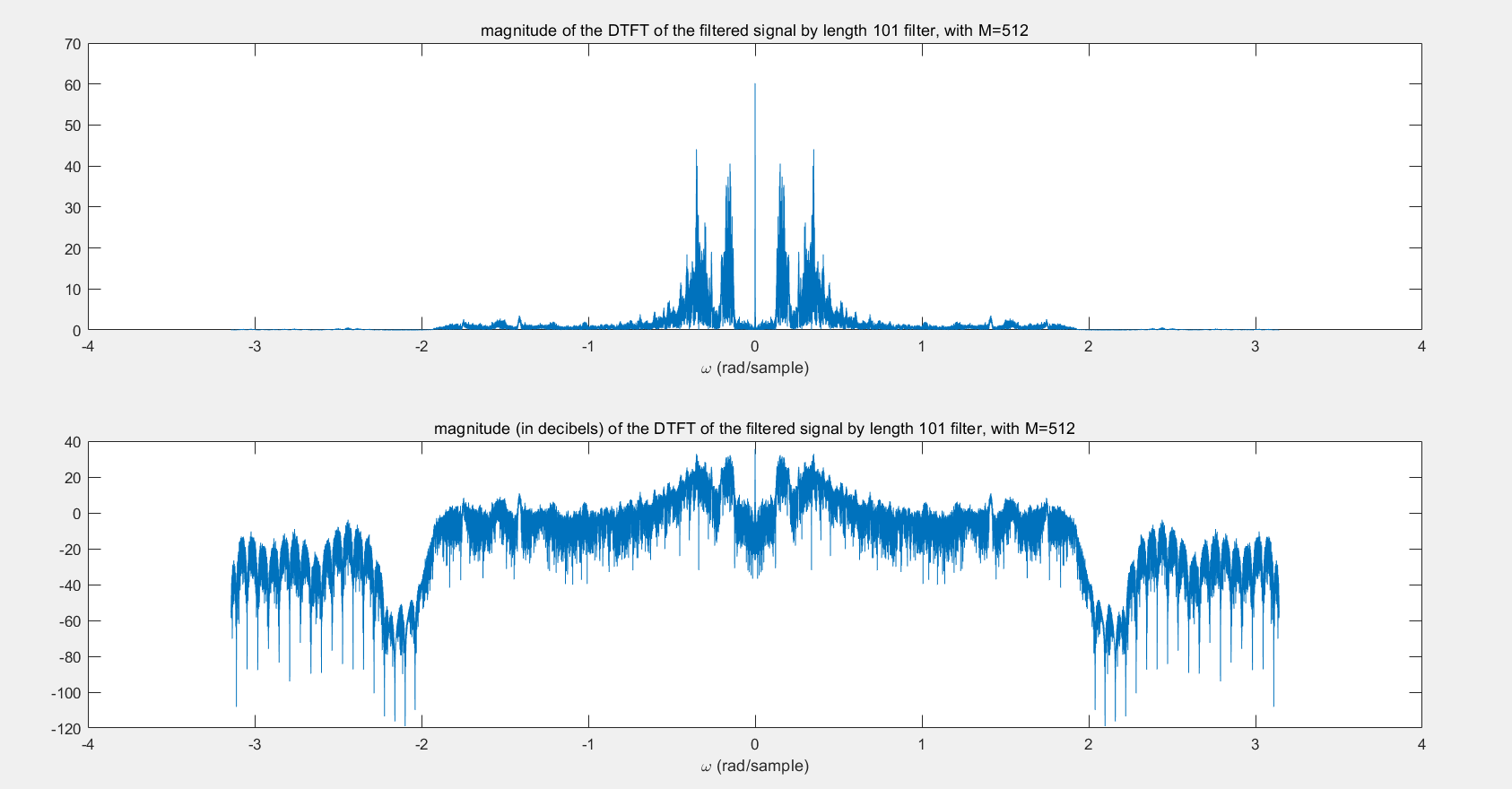


Figure 7.5.8

We can see that both the filter can remove the high frequency noise of the original signal and has little distortion in the low frequency part.

There is some difference between the signals filtered by the two filters

We can see from the time domain plot that the one filtered by length 101 filter has less noise wave.

And from magnitude of the DTFT we can see that the one filtered by length 101 filter has less component of frequencies among the transition band of filter with length 21. (Circled by green circle)

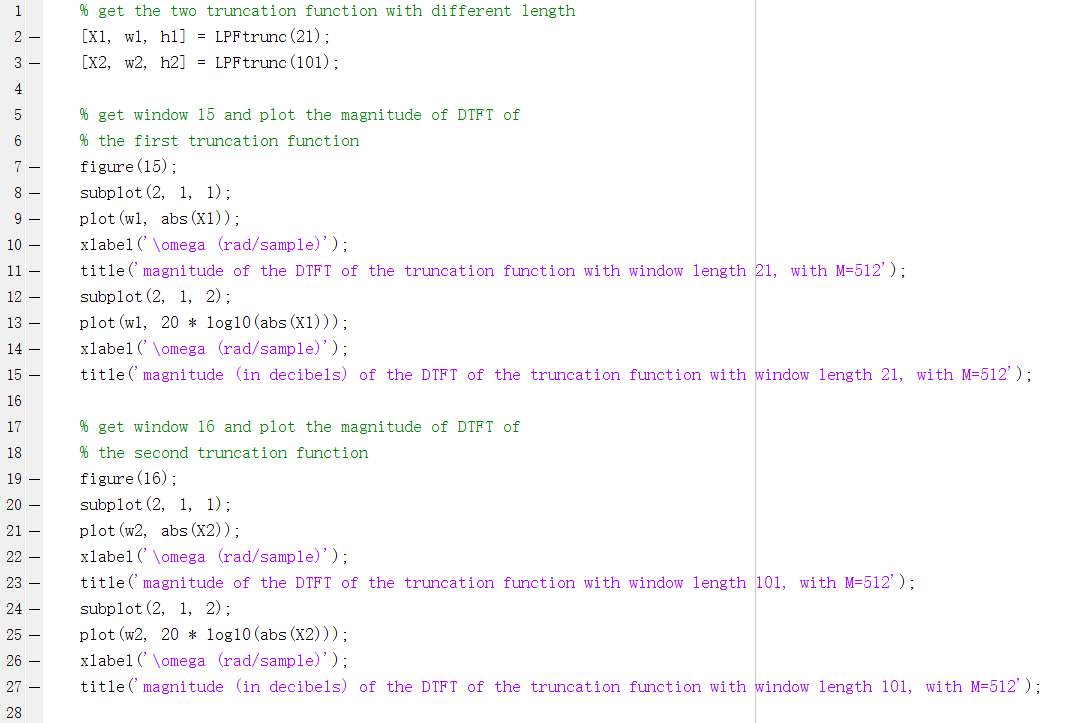
The sound test shows that we can hear the voice from both the filtered signal, but the former one contains more noise.

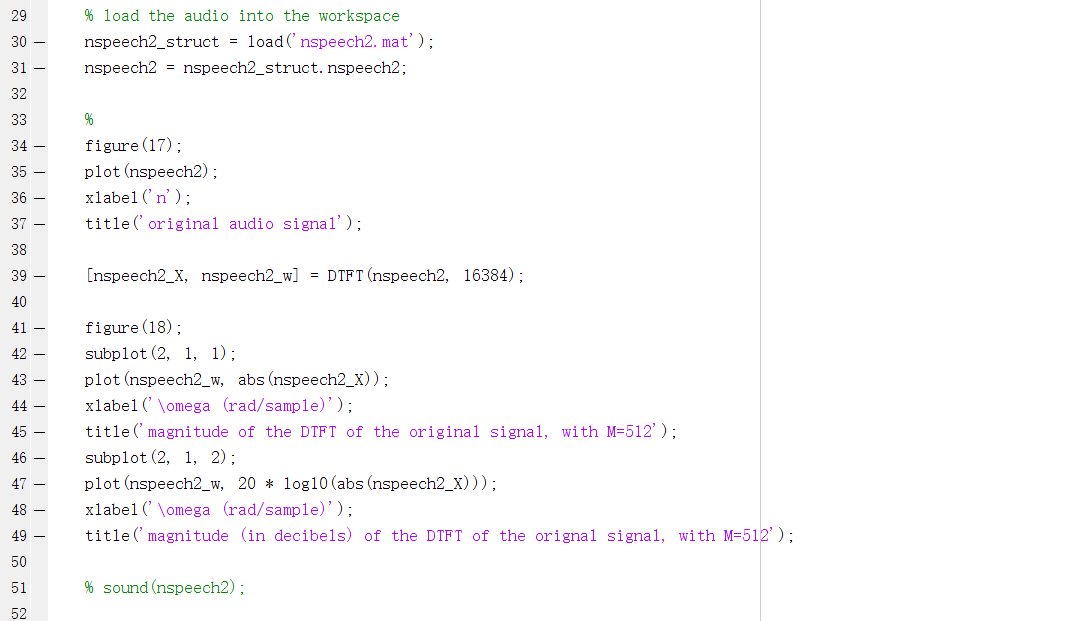
The voice is *“if you can hear this clearly, you filter the signal correctly”*

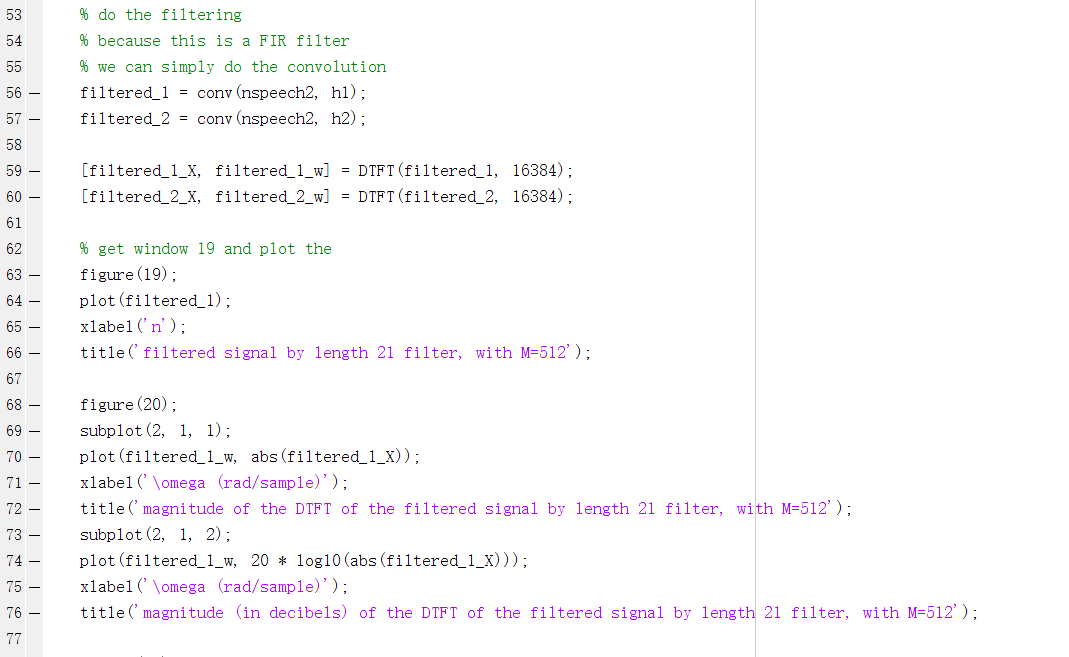
I also tried another length 501 filter, but it still cannot get rid of all the noise, because there still noise in low frequency.

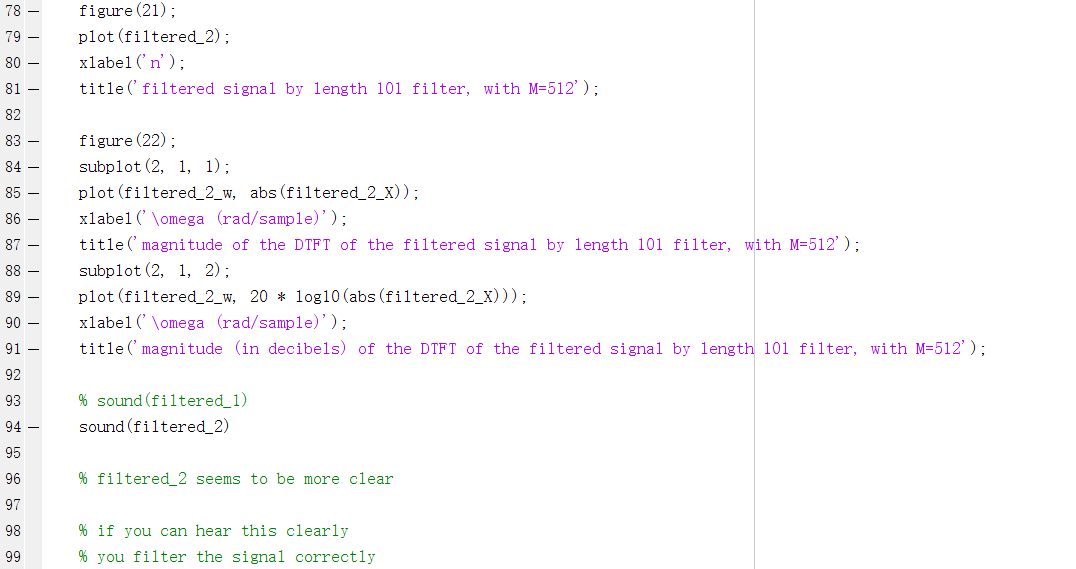
Thus the filter size till have a noticeable effect on the audio quality when the size is not large enough. With the increasing size of the filter size, there is less effect cause by the increasing of the size, because the effect of size mostly is caused by reducing the transition band. (We do not consider the very small size of filter that the ripple takes effects a lot).

The code of this script is shown in the code 7.5.1 (corresponding to the file lab7\_6.m)



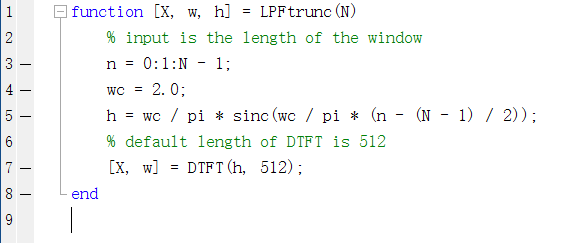






Code 7.51

And the code of the function ‘LPFtrunc’ is shown in the code 7.5.2 (corresponding to the file LPFtrunc.m)



Code 7.5.2

**7.6 Conclusion**

In this laboratory, I have a deeper understanding of how to design a filter by directly from the transfer function and by implementing ideal filter using truncation functions (rectangle function we used here, more in the next lab)).

The first one is easy to implement but it is hard to get the specifications to the filter parameters especially when the order of the filter is large. The second one is easy to set the specification because we only consider these for ideal filters, but how to choose a proper truncation function with proper length is a big task.

Though they are not the filter design we use today, but their idea on how to deal with signals is important.

Additionally I notice the size of DTFT is important in DSP, less length may cause important information loss while dealing with the signal just as I meet in section 7.3.2.