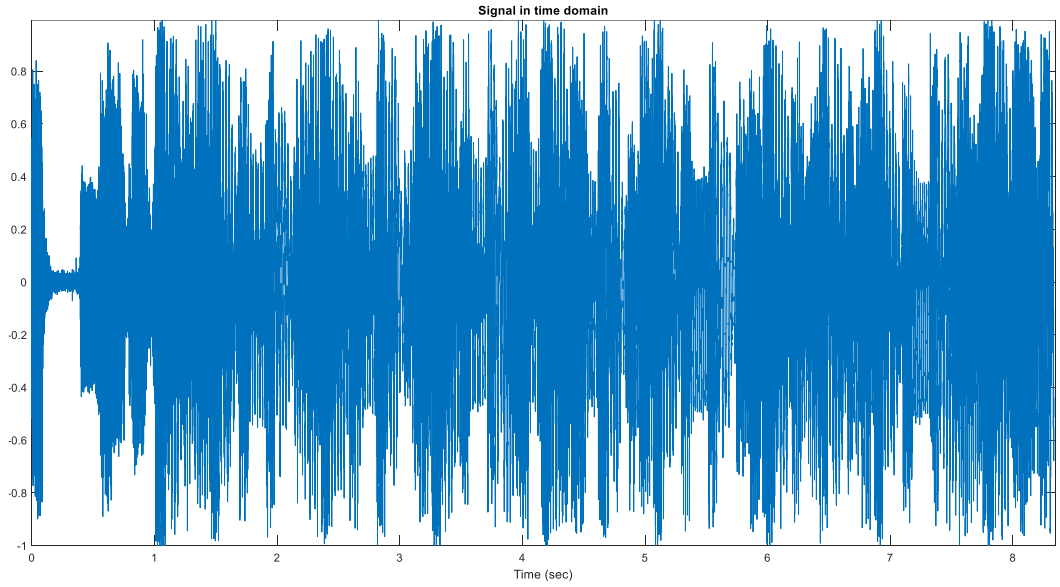
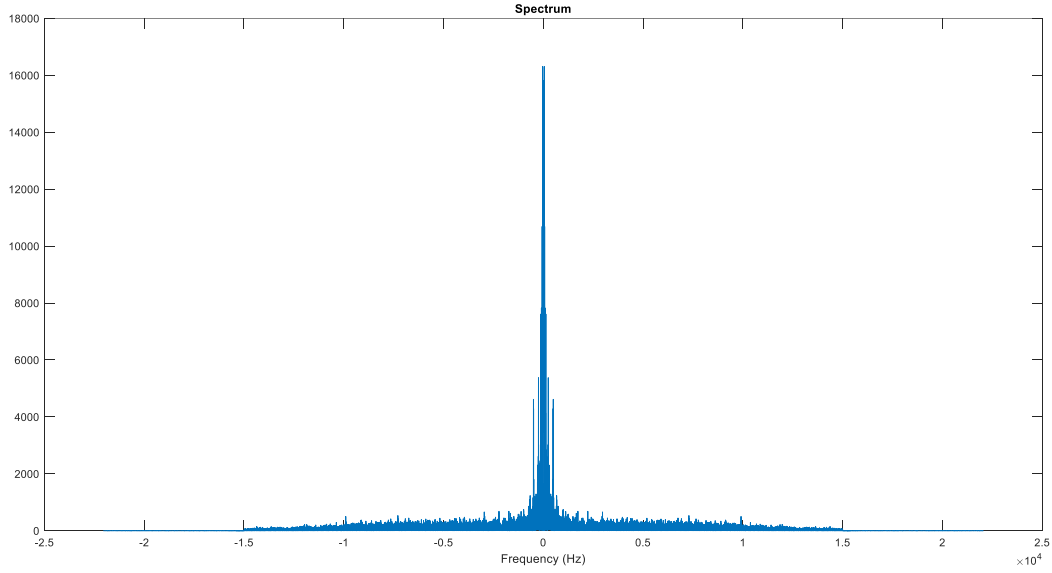
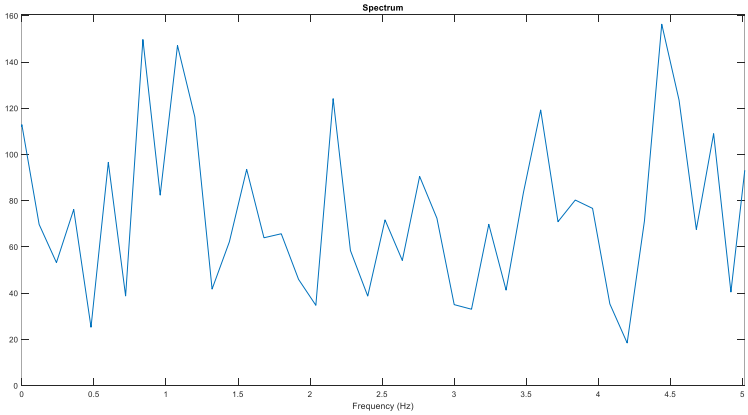
1. (10%) Read the music (sister_8sec.wav) using MATLAB **audioread()**. Plot the signal in time domain and its frequency-domain representation (i.e., magnitude spectrum) using MATLAB **fft()**. Remember to plot with the correct time and frequency axes. Can you find the signals representing drum beats in time or frequency domain? If you can, explain why the signals of drum beats present in such a form.

   Ans:





Listen to the sound, and you can hear 16 drum beats in 8 sec (T = 8/16, fs = 1/T = 2 Hz). Hence, your ears can hear the drum beats in the time domain, but it's difficult to find the 16 similar beats in the above figures.



Zoom in to find the impulse train, and 1/T (T = 8/16, fs = 1/T = 2 Hz) is the beats. However, there are too many peaks in the frequency domain. It may not be easy to see beats in the frequency domain.

2. Filter the music. Apply the designed low-pass filter, high-pass filter or band-stop filter to the music and listen to find the differences.

(a) (20%) Use the MATLAB function **fir1()** to obtain the impulse response **h** of a low-pass (or a high-pass) filter. Change the filter order to 8, 16, 32, 64, 128, and 256, and then briefly describe the difference between their frequency responses (by MATLAB function **freqz()**) and impulse responses (by MATLAB function **stem()**). Plot the frequency responses and impulse responses to find the differences. From your observation, please also tell the idea how the **fir1()** performs the low pass filter design (Hint: compare the ideal low pass filter and the filter designed by **fir1()** or see problem 5.55 in Handwriting HW4). Remember to check whether the designed filter works or not by comparing the signal spectra before and after filtering.

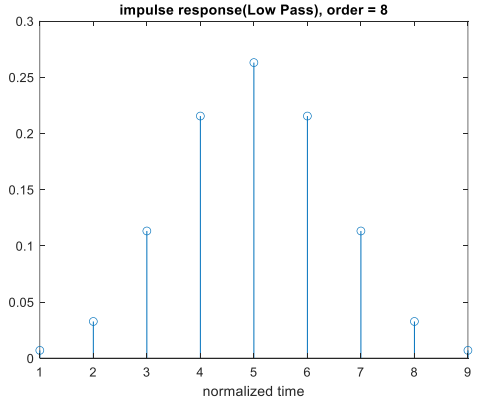
If you'd like to, you also can try band-stop filtering of the music to see if you can get rid of the vocal part and get a karaoke version of the music.

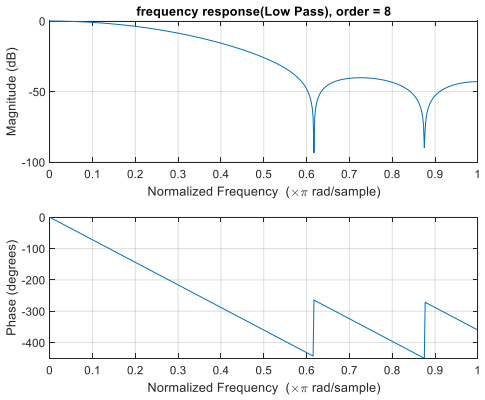
Ans:

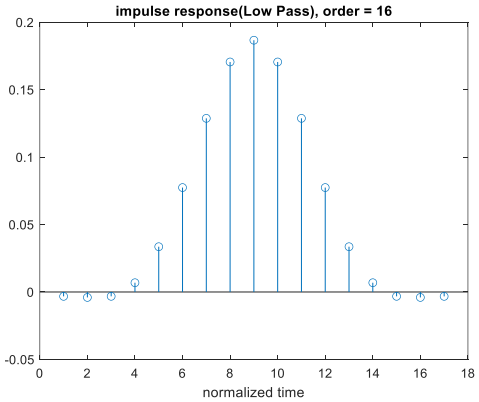
Low-pass filter:

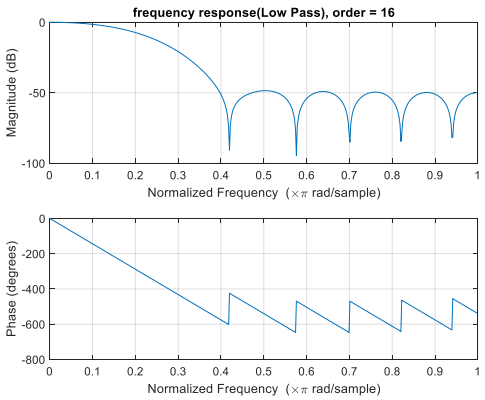
Impulse response h[n] (time domain) of the LPF designed by fir1().

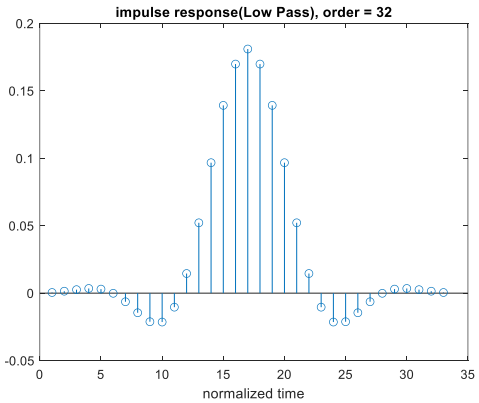
Cut-off frequency = 4000 Hz.

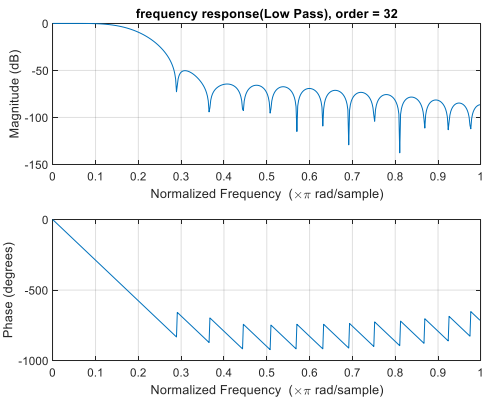

impulse response(Low Pass), order = 8
normalized time

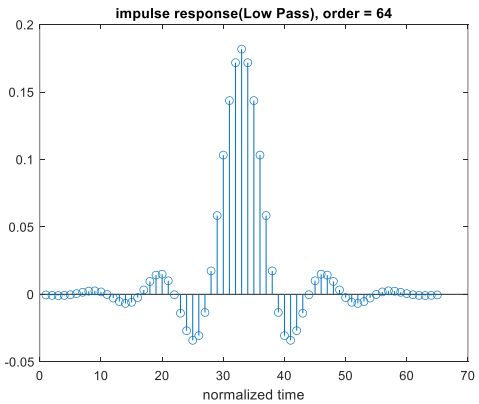

frequency response(Low Pass), order = 8
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)

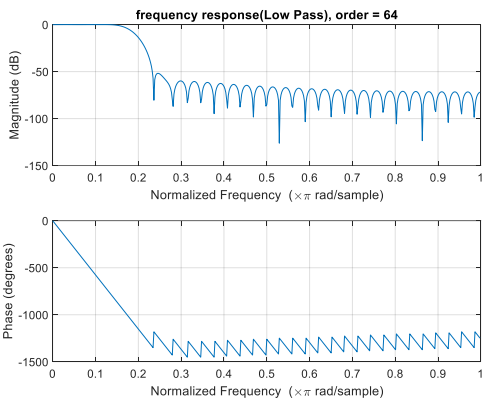

impulse response(Low Pass), order = 16
normalized time

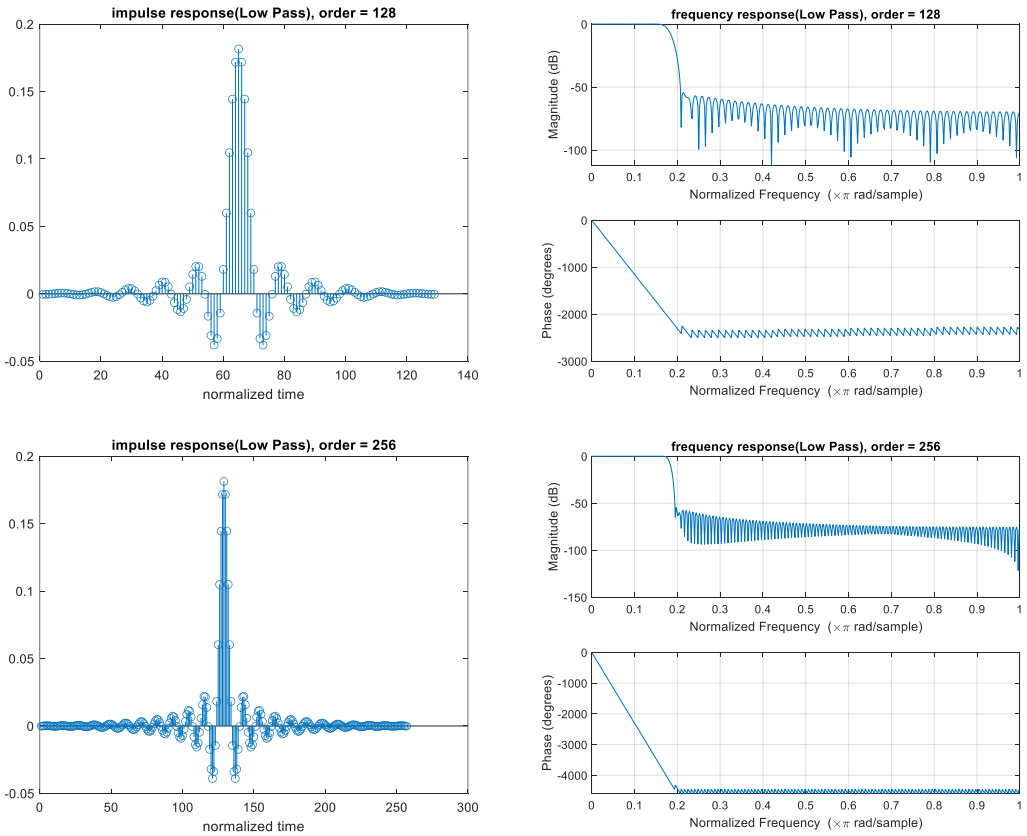

frequency response(Low Pass), order = 16
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


impulse response(Low Pass), order = 32
normalized time


frequency response(Low Pass), order = 32
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


impulse response(Low Pass), order = 64
normalized time


frequency response(Low Pass), order = 64
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)

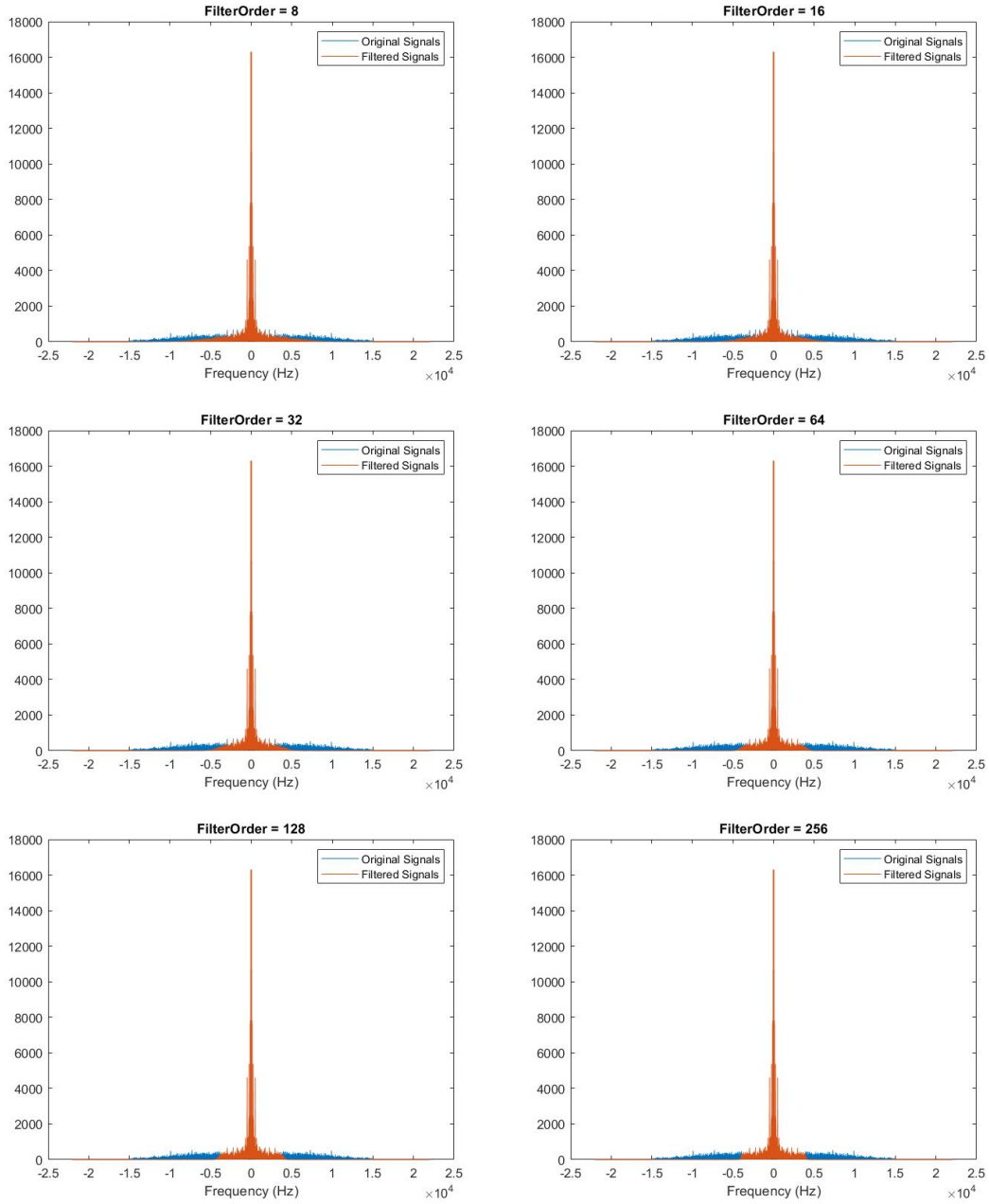


From the results shown above, we can notice that as the filter order increases, the output waveform in the time domain is more similar to the sampled sinc function.

Observing the frequency response, we can notice that the results become more close to an ideal LPF as the order increases. Besides, the magnitude spectrum has a sharper transition band and lower stopband ripples.

The MATLAB function fir1(): From the above observation, the fir1() should design a FIR filter by 'windowing' a sampled sinc function, which is a type of FIR approximation—FIR implementation of an IIR system. From the MATLAB help or doc of fir1(), we can know that by default, fir1() performs windowing with a Hamming window instead of rectangular window taught in the lectures. As the filter order increases, the 'window' size becomes larger, and the filter will behave more close to an ideal frequency selective filter (see also slide 59 to 64, Topic5_TimeAndFreqAnalysisOfSignalsAndSystems_HandWriting0602_2021.pdf).

Compare the signal spectra before and after filtering: it is found that the frequency domain signal in the stopband are suppressed, which will be more clear if the zoom-in spectra are plotted and the attenuation ratios/or rejection ratios in the stopband are very close to the gains in the stopband.



(b) (10%) Listen to your results carefully and answer the following questions. How differently do the low-pass and high-pass versions sound? Does the filter length affect the sound quality of the output? How and why? Note that the FIR filter length is the support length of the impulse response and is equal to the filter order + 1. Note that you had better vary the cut-off frequency of your filter in order to get better feeling of the processed music.

Ans:

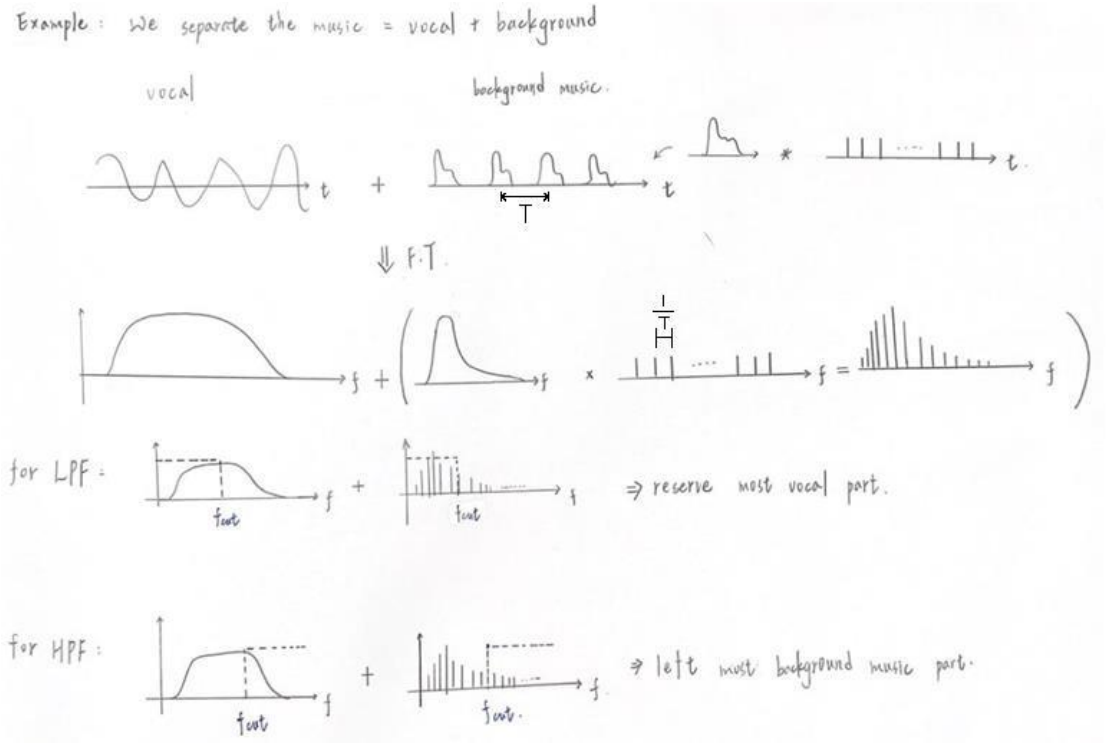
(1) How differently do the low-pass and high-pass versions sound?

The answer depends on the cut-off frequency.

- Low-pass filter version:

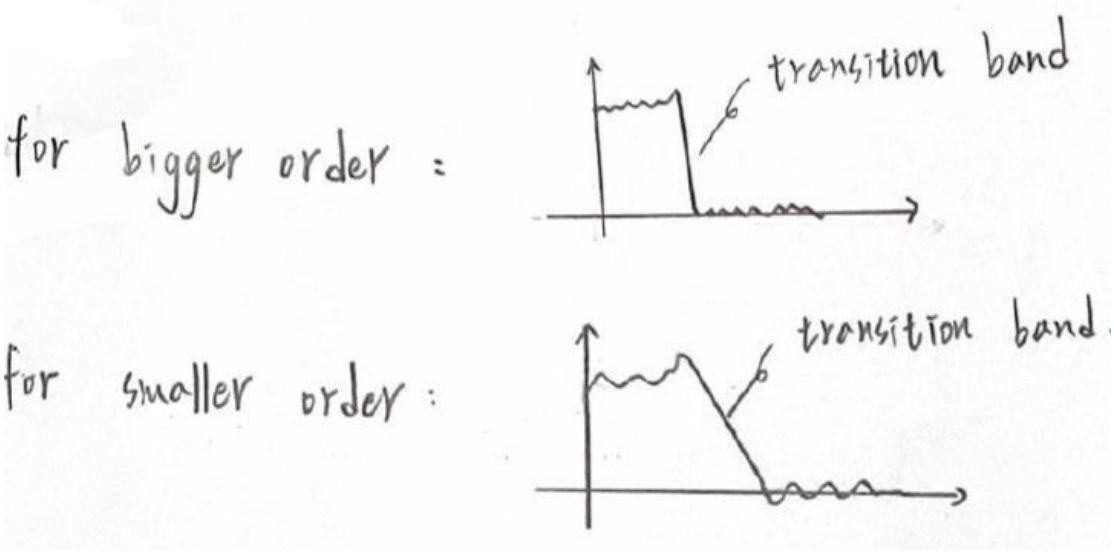
Since the low frequency part of the music (vocal part of music) is reserved by the low-pass filter (cut-off frequency = 4000 Hz), basically you will hear the vocal part more clearly than the instrumental part. If the cut-off frequency is lower than 4000 Hz, the above conclusion may be different.

- High-pass filter version:
  In contrast, using high-pass filter (cut-off frequency = 500 Hz) will reserve the high frequency part of the music, and the vocal part will be suppressed. Hence, you may feel the weaker vocal part and the emphasized instrumental part. If the cut-off frequency is higher than 500 Hz, the above conclusion may be different.



(2) Does the filter length affect the sound quality of the output?
Since the FIR filter length is the support length of the impulse response, which is equal to filter order + 1, you can simply compare the different cases by modifying the order of the low/high-pass filter. From the perspective of math, choosing a larger filter order will make the frequency selective filter become more ideal. With the larger order, the transition band becomes sharper, and the stopband ripple magnitude is lower as illustrated in the figure below.
If the order is small, your ears can hear the closely same music in the time domain. With the larger filter order, the effects of low-pass filter or high-pass filter will be heard apparently because there are fewer spectral

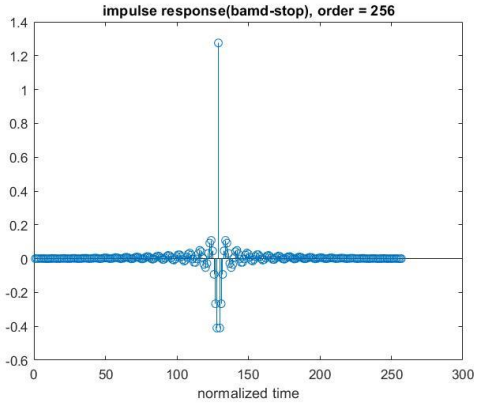
components left in the transition band and pass band.

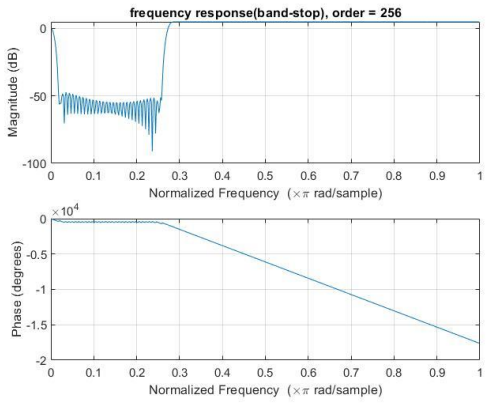


(c) (20%) Use fir1() to design a band-stop filter with which 0.5-kHz to 8-kHz sounds are suppressed. Convolve your filter with the provided audio clip 'sister_8 sec.wav'. Can you hear the vocal part? Adjust the cut-off frequencies of the band-stop filter so that the vocal part you hear, is maximally suppressed while keeping the sound quality of the background music. Comment on how such a processing can provide you a karaoke version of the music.

Ans:

```
% cut-off frequency = [100Hz 6000Hz]
bs = fir1(256, [100/(Fs/2) 6000/(Fs/2)], 'stop');
```





經過一些嘗試之後，以上述規格設計出來的 band-stop filter 雖然沒辦法將 vocal part 完全去掉，但至少有起到抑制的效果，讓 vocal part 不要這麼明顯。使用 band-stop filter 的主要原因是利用 stopband 抑制人聲頻率區間的訊號，但因為人聲頻率區間因人而異，且這個區間內也可能會有樂聲訊號出現，所以即使經過 filter 後，還是有可能得不到音樂的 karaoke version。

因此取的 stopband 不能太廣，要濾掉的低頻太低，bass 聲也會被濾掉；要濾掉的高頻太高，樂器聲也會被濾掉。

3. Re-sample the given ECG signal and make it audible.

(a) (20%) Given a MATLAB data file –ECG.mat where a raw ECG signal (i.e., ECG) and the sampling rate (in Hz, i.e., ECG_Fs) used to acquire the ECG signal are stored, resample the given ECG signal so that the resampled ECG signal has the same sampling rate as that of the music “sister_8sec.wav” in problem2. Note that after re-sampling, the resampled ECG signal should have the same data length as the music “sister_8sec.wav” and the re-sampling should result in no aliasing in frequency domain. Detail your signal processing procedure for the re-sampling along with your codes. If in the procedure, you apply filtering, tell how you design the filter. Note that you can verify your results by using MATLAB resample().

Ans:

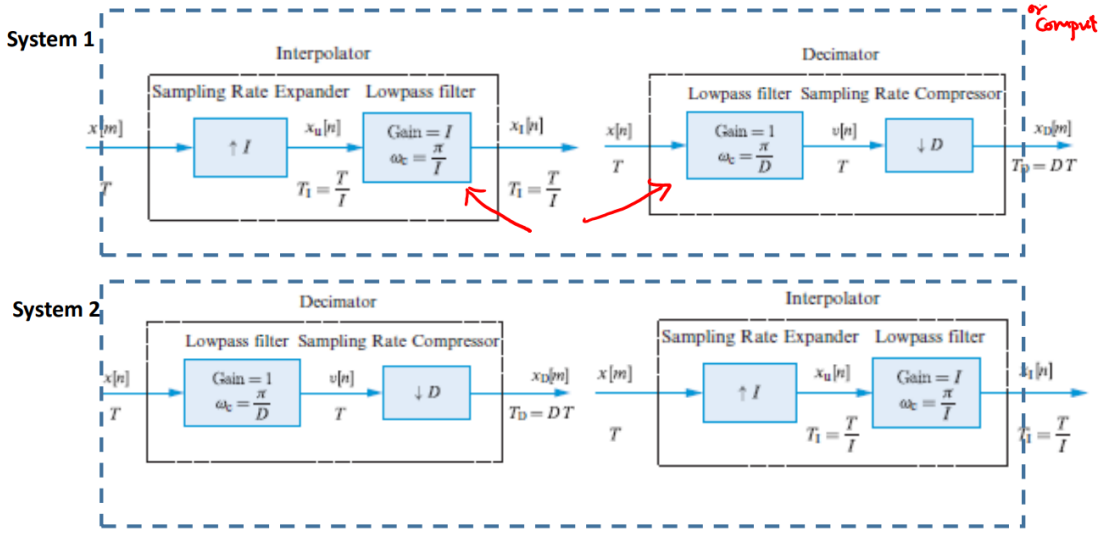


Figure: system block diagram from slide.44 in “Topic4_SamplingAndReconstruction_Part2_HandWriting0526_2021”

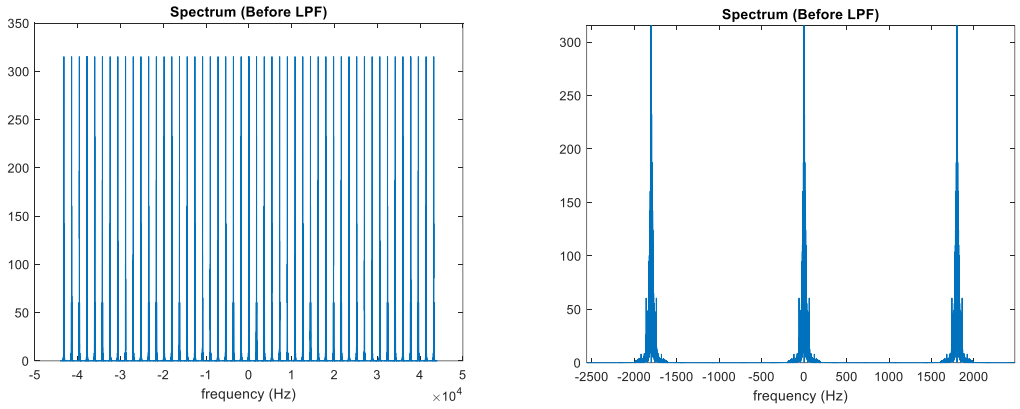
(1)

First, when we need to perform up-sampling(interpolation) and down-sampling(decimation) simultaneously, we will choose to go through system 1, which means we will perform interpolation first then perform decimation to avoid potential aliasing in frequency domain and to preserve more spectral information as much as possible. If we choose to go through system 2, we may have chance to lose some information of signal when we perform lowpass filtering in the decimator. Moreover, the factor of both interpolation and decimation should be integers. In this case, sampling rate

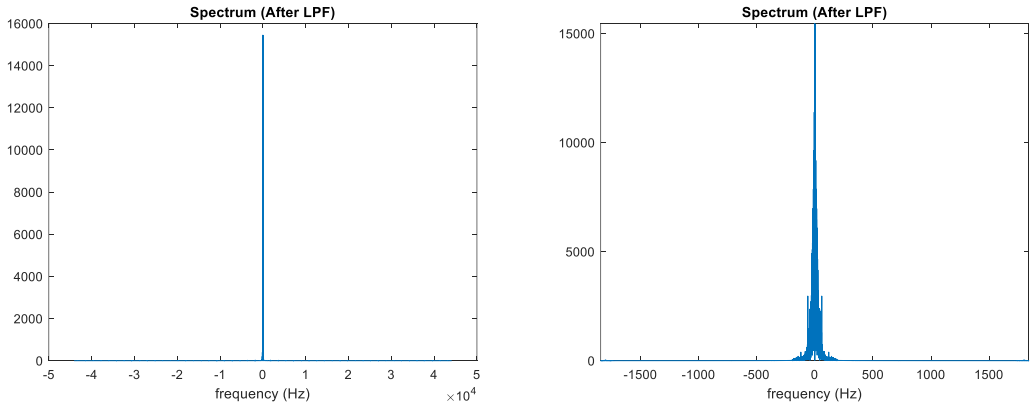
of ECG signal and the music is 1800 Hz and 44100 Hz, respectively. Therefore, I set the factor of interpolation and decimation as 49 and 2, which are co-prime, respectively. In this case, the interpolation and decimation ratios are the lowest; that is, computational complexity for the re-sampling is the lowest.

(2)

As for the LPF, the interpolator and the decimator could share the same LPF, which means we can combine two LPFs into one. The cut-off frequency should be the smallest one between the two, and the gain should be the same as the factor of interpolation. In this case, the smallest cut-off frequency would be $\frac{1}{49}$ cycle/sample, and the gain of the filter should be 49. Then, we could design our LPF with the cut off frequency, the gain, and the function "fir1". In below figures, the order I use is 256.



(left) spectrum of upsampled ECG signal, (right) zoom in



(left) spectrum of filtered ECG signal, (right) zoom in

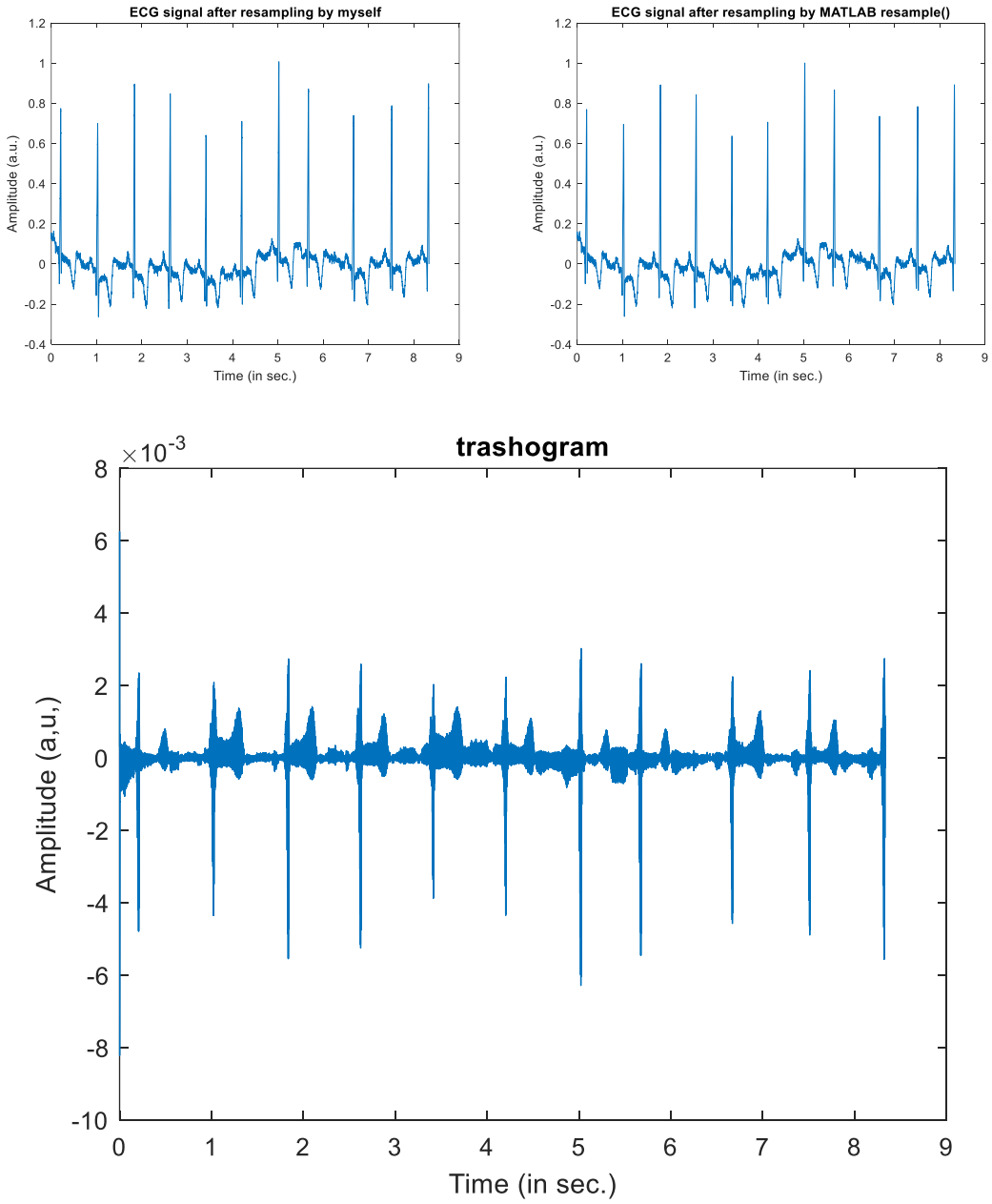


Figure: Difference between my-resample and MATLAB resample() – the difference is small enough, i.e., < 1 x 10^{-3}

(b) (20%) If you simply play the resampled ECG by your computer sound card and speaker (i.e., MATLAB codes – soundsc (ECG_resampled, Fs), you will find out you BARELY hear the ECG. Figure out why you cannot hear the ECG sound and devise a signal processing procedure (i.e., a DT system) to make the processed ECG audible and justify your processing (see the in-class demo of the audible ECG). Remember to output your audible ECG to "AudibleECG.wav".

Ans:

Whether you can hear the original ECG signal depends on the range of

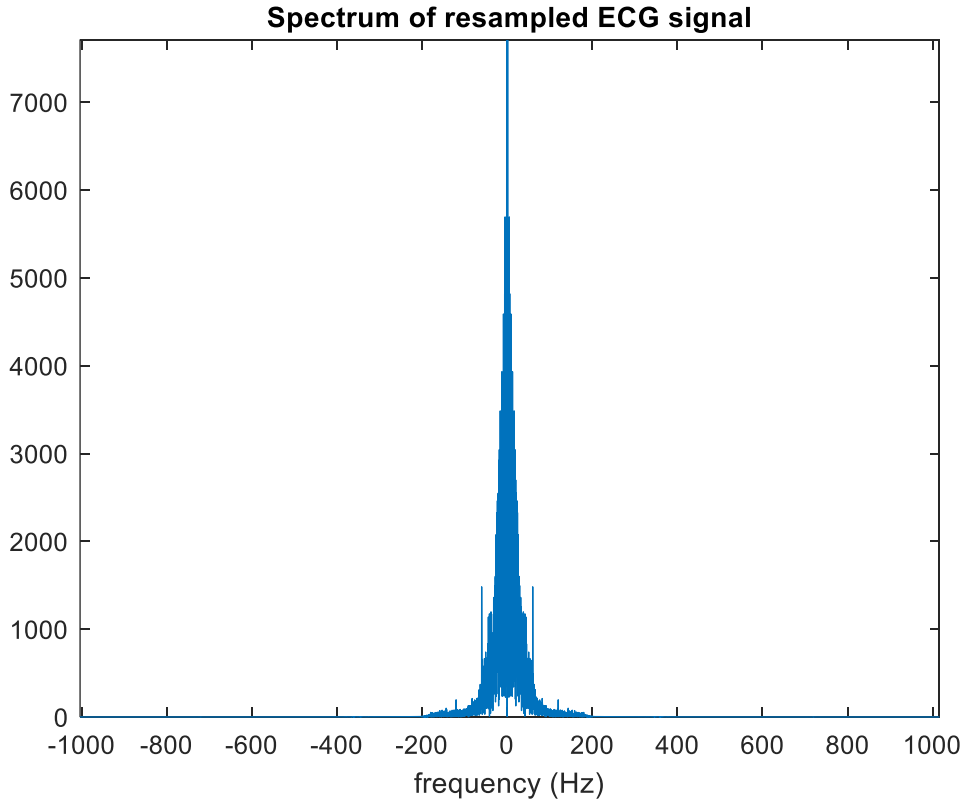
frequency response of your sound card or your speaker. (You can check the provided websites of the speakers and earphones at Computer HW4 assignment page.)

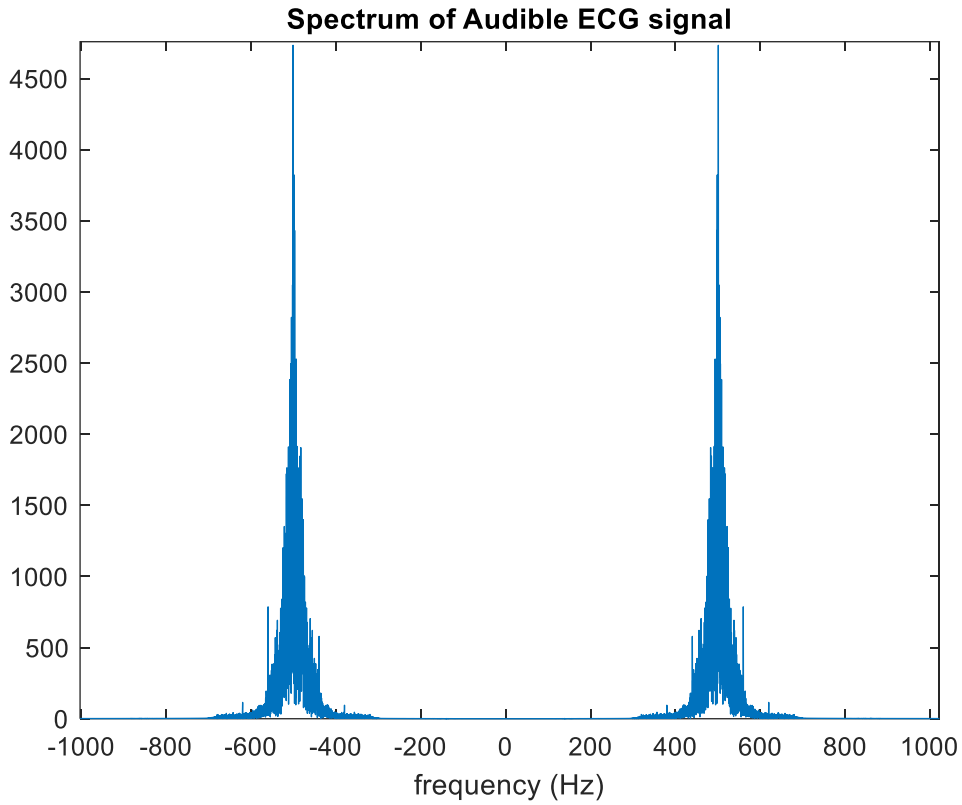
Normally, you CANNOT hear ECG signal via the speaker because the frequency of ECG signal is out of the range of the frequency response of your speaker. However, we can apply signal processing procedure to make the processed ECG signal audible.

In this case, I apply amplitude modulation(multiplying ECG signal with a cosine signal with a carrier frequency (500 Hz or 1 kHz), which means shifting the spectrum of ECG signal to higher frequency in frequency domain to make processed ECG signal audible. Because after frequency shifting, the frequency domain signals are well within the passband of the frequency response of the speaker

```
fc = 500; % Hz
t = (0:length(ECG_dec)-1)/Fs;
AM = cos(2*pi*fc*t)'; % amplitude modulation

AudibleECG = ECG_resampled.*AM;
```





As for why you are asked to perform re-sampling the ECG signal to the sampling rate of the music, it is to ensure that the sampling rate used to play the ECG sound is supported by the DAC the sound card. The original sampling rate of the ECG signal is too low and may not be supported by the DAC of your sound card.