Assignment 1 CS2108 – Sara Djambazovska

Our task for this assignment was to perform demodulation on the input signal, by shifting and filtering, extracting the two original information-bearing music signals. I started the analysis on the given sound, by first reading the provided audio file with the audioread MatLab function. It returned the sampled data - the signal: y, and a sample rate for that data: Fs. In order to play this sound I created an audioplayer object from y and Fs, and passed it as a parameter to the play function. The next step was obtaining the Fast Fourier Transform of the signal, which was easy by using the build in Matlab function fft, and specifying the number of points to be the length of the input signal(even though this would have been the default if we do not specify a length for the fft). After plotting the fft (Figure 1.2) I used it for visually finding f1 and f2, the centre positions of the two music pieces that had been moved from the 0Hz frequency position, by modulation. After inspecting the plot(Figure 1.1) I detected the value, the point **255000**(->249400 + ½*(260600-255000)) that can be used to shift f1 in the frequency domain (equivalent for the frequency of 255000*Fs/N =>**3000Hz** in the time domain), and **680000**(->677500 + ½*(682500-677500)) for f2(**8000Hz**).

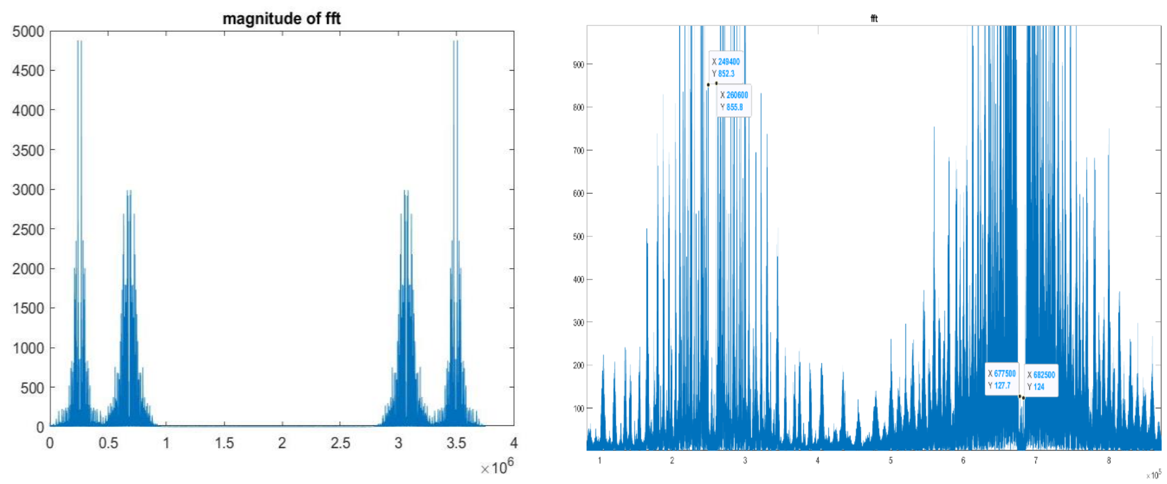

Figure 1: Magnitude of the FFT of y and obtaining the shifting amount from it (right)

In order to demodulate the signal and extract the audio information stored in the first "peak", I had to perform a multiplication with a cosine wave of the frequency for which the signal was shifted. I interpreted the input signal as $y(t) = info1(t) * \cos(2\pi f1t) + info2(t) * \cos(2\pi f2t)$. So by multiplying it with a $\cos(2\pi f1t)$ we would shift the info1(t) function to the 0 Hz frequency, and perform a lowpass to the shifted time-domain signal, to remove everything greater than the max passband frequency for sound 1, thus extracting only info1(t). Shifting in the time domain is equivalent to convolution in the frequency domain, Figure 4 shows the equivalence, neglecting the amplitude difference that can be fixed by scalar multiplication. In order to be able to recover the music "sitting" on f1(3kHz in time domain), I had to shift the signal for the value found, by using circular convolution with an impulse(length of y) that is the fft of the required cosine wave of f1 = 3kHz, which is an impulse having all its energy at two points, namely at f1 and -f1, which scaled for the frequency domain are the points 255000 and -255000. Looking at the output of the magnitude of the fft, we can observe that it is symmetric, since it is real, it is of the form 0 fs/N 2fs/N...(N/2-1)fs/N -(N/2)fs/N -(N/2-1)fs/N ... -fs/N, where N is the number of points in the result of the FFT, if we just shift with a one impulse function, we would lose the symmetry. That is why I created an impulse train having only two ones at f1 and at N – f1 + 1 (the +1 is because there is no zero at the end of the negative part of the fft, it starts backwards from -1fs/N). After doing the circular convolution of this impulse train with Y (fft of y), I obtained the shifted signal, where now music 1 is "sitting" on the 0 frequency (Figure 2 left).

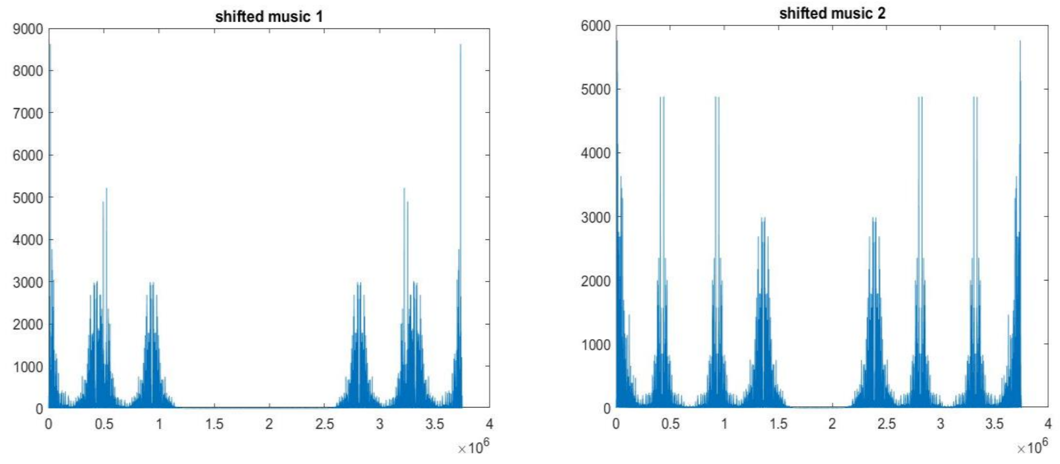

Figure 2: The result after shifting the original signal by f1(left) and by f2(right)

Subsequently, I had to filter only this piece of music, by using the lowpass function on the time-domain shifted signal. I wanted to get rid of high frequency noise component, so I used a low pass filter to throw away higher frequencies. In order to get the correct passband frequency I used the fft plot again but this time using the real frequency as the x-axis(by scaling it times Fs/N), and noted the width of the first piece, starting from the centre to one of its edges. Namely, the parameters to the lowpass function are ifft(of the shifted fft), the passband frequency obtained by manual inspection, and the sampling rate which is the same as the one used to sample the original sound. Figure 3 shows my result upon doing the lowpass filtering. I managed to obtain a fairly clear sound and I saved it as music1_recovered.mp4 by using audiowrite. Subsequently, I did the same for the second piece of music, only this time it was placed on f2, which I discovered to be the point 680000 in the frequency domain, created an impulse train with only two 1s at f2 and N-f2+1, did the circular convolution with the original signal and after manual inspection revealed the fmax (2643Hz) passband frequency which I used in the lowpass filter to obtain the second piece of recovered music.

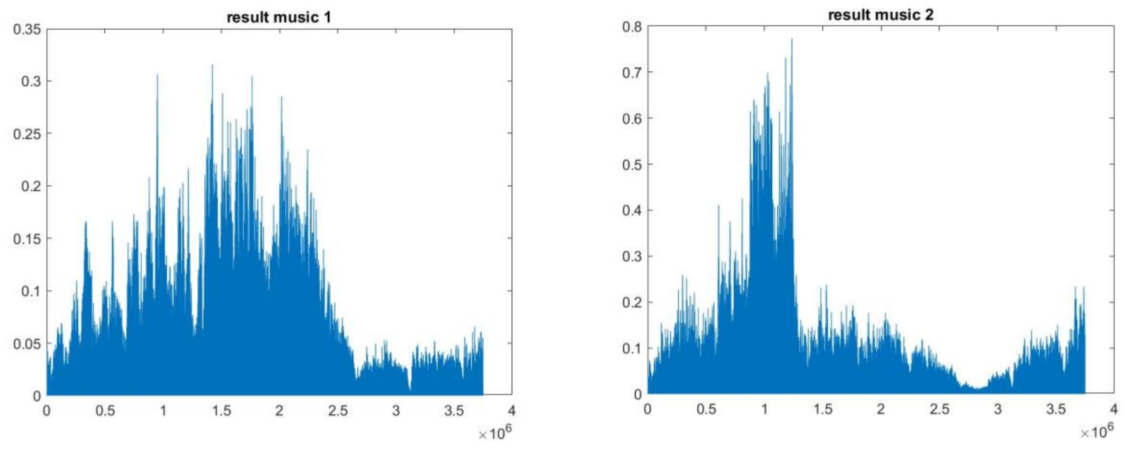

Figure 3: final result after low pass filtering, music 1(left), music 2(right)

In order to obtain the magnitude and phase of the saved music audios, I read the two signals and used the abs and angle functions on the signals to obtain their magnitude and phase accordingly. To synthesize a new piece of music whose fft has the magnitude of music 1 and a phase of music 2, I used Euler's formula where the fft of the mixed signal1 = Magnitude1 multiplied (pointwise) with $e^{i*Phase2}$. An the fft of the mixed signal2 = Magnitude2 multiplied (pointwise) with $e^{i*Phase1}$ accordingly.

Answers to the proposed questions:

After saving the two signals as newMusic1.mp4 and newMusic2.mp4 with audiowrite, I played the two audio files and realized that the first one(newMusic1) has a big similarity to the recovered music 2, and almost no connection with recovered music 1, and the second mixed sound sounds a lot like the first recovered music, and has almost no connection with recovered music 2, even though they have the same magnitude.

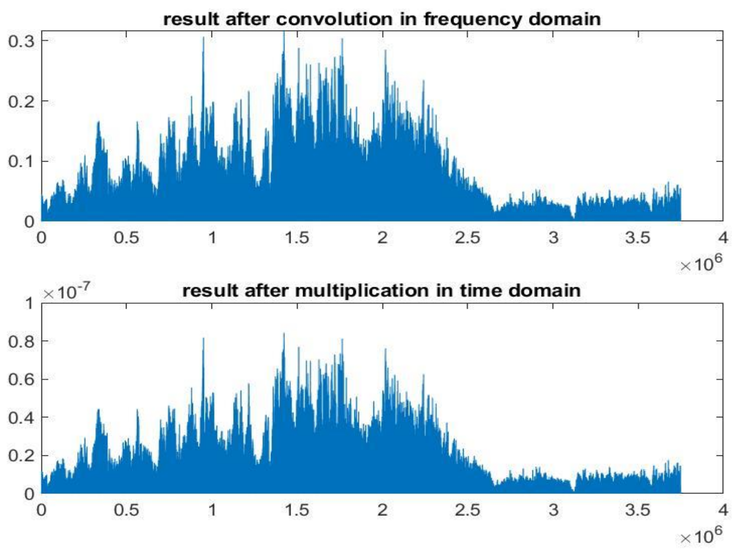

Figure 4: Showing the equivalent end result for music 1 (neglecting the amplitude) after using cconv in the frequency domain or pointwise multiplication in the time domain

This led me to the conclusion that the phase component carries more crucial information for the sound than the magnitude component of the fft. Meaning, by only using the same phase and an arbitrary magnitude we are able to still recognize the original sound, however that is not the case with the magnitude, it does not keep enough information to be able to reconstruct partially the signal given only its fft magnitude. The magnitude of a signal determines the relative presence of a sinusoid $e^{-i\omega t}$ in y(t). Since y is real then the magnitude of the fft of y, Y, is even, meaning abs(Y(f)) == abs(Y(-f)), the fft magnitudes are the same. It follows that if y(t) is real (audio signal) then y(-t) would be playing the same signal backwards, meaning a different signal, but both would have the same magnitude of their fft, however their fft phases would be different! The phase of a signal gives information about the relation between the different frequency components making up that sound, their ordering. It determines how the sinusoids line up relative to each other to form y(t). Meaning, the phase component of the fft of the sound is crucial, containing far more information than the magnitude component and should be preserved when filtering the sound! However, we realize that the new mixed sound does not sound exactly like the original recovered sound, meaning that even if we use the same phase, we still need to use the same magnitude component as well to be able to completely reconstruct the signal.