# Digital Signal Processing Project

Aurora Zabot, 1206742

A.Y. 2018/2019

## Introduction

In the project assignment, it was requested to recast an audio signal from the form:

$$y(nT) = (x_1(nT) + A_1)cos(2\pi f_1 nT) + (x_2(nT) + A_2)cos(2\pi f_2 nT) \quad (1)$$

where $x_1(nT)$ and $x_2(nT)$ were two real audio information signals with bandwidth [20, 8000] Hz, while $f_1$ and $f_2$ were the frequencies of two sinusoidal carriers with their respective amplitudes $A_1$ and $A_2$. In particular, it was asked to find the two carrier frequencies ($f_1$, $f_2$) with their amplitudes and isolate them through a very narrow bandwidth filter in order to demodulate $x_1(nT)$ and $x_2(nT)$.

## 1 Theory

At the very beginning of the analysis, I would like to explain the main concept behind the demodulation of the given signal. First of all, demodulating means extract the original information-bearing signal from a carrier wave. Looking at the structure of $y(nT)$ before filtering, I can see two kind of components: one is related to the real signal, $x_i(nT)$, while the second one regards a sinusoidal carrier $A_i cos(2\pi f_i nT)$ of a certain amplitude and frequency (with i = 1, 2). In order to take out the real information signals, I need to select properly with a very narrowband band-pass filter the two undesired frequencies and then multiplying the modulated signal by the previously extracted carriers; what it's obtained in this way must be also multiplied by the gain $\frac{2}{A_i}$. After this computation, I just have the information signals $x_i(nT)$ and I need to filter them in order to maintain frequency components into the interval [20, 8000] Hz; to do this, I'll use another band-pass filter obtained cascading a low-pass and an high-pass notch filter. Summing up all those components, the obtained $y(nT)$ would be a clean audio signal.

## 2 Carriers manipulation

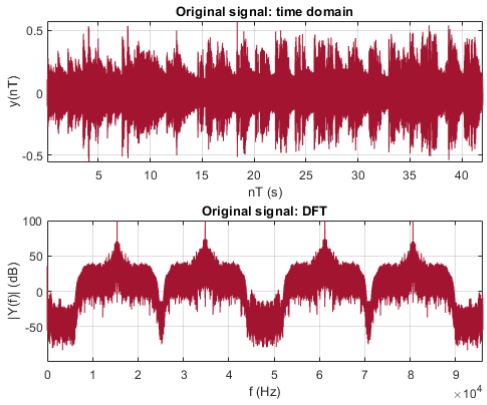



Figure 1: Spectrum of input signal in time domain and its DFT

The very first request of the project is to plot the spectrum of the given input (Figure (1)) and find, through an appropriate implementation, the two carrier frequencies and their amplitudes.
In order to do that, I implemented a function to discover the two peaks into the spectrum, making a comparison in the absolute value of the DFT of the input and using the function *find* (otherwise I would use the MATLAB function *findpeaks*, that gives the same results).
The obtained outcomes were:

- $f_1 = 15'400.02Hz$ with $A_1 \simeq 0.0500$
- $f_2 = 34'800.02Hz$ with $A_2 \simeq 0.0500$

As a confirmation of the results, it can be noted that:

- $f_2 - f_1 = 19'400Hz \geq 17'000Hz$
- $10'000Hz \leq f_1 < f_2 \leq 38'000Hz$

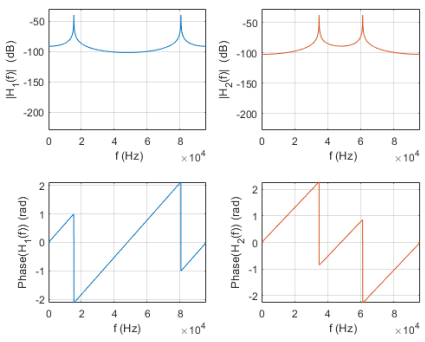
Now we had to move on dimensioning the two filters; I decided to use a second order IIR filter with $M = 0, N = 2$:

$$H(z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{2}$$

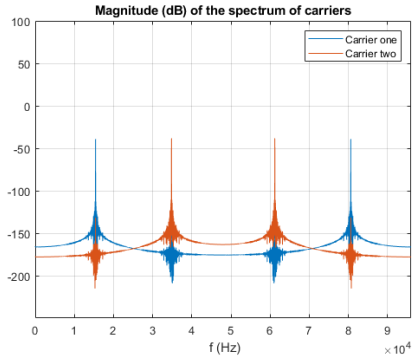
I set the parameters $a_1, a_2, b_0$ as we've done during the course:

- $a_1 = 2rcos(2\pi \frac{f_i}{F_s})$ with i = 1, 2
- $a_2 = -r^2$
- $b_0 = 2(1 - r)sin(2\pi \frac{f_i}{F_s})$ with i = 1, 2

The two filters can be visualized through Figure (2a). For completeness, at the very end of the process, I extracted the spectrum of the two carriers (Figure (2b)).




(a) Filters used to extract carriers




(b) Spectrum of the carriers

## 3 Demodulation and filtering

To demodulate the two information signals, as it was suggested in the assignment, I multiplied the input by the previous extracted carriers and by the gain. I need to maintain just the frequencies in the interval [20, 8000] Hz using a band-pass filter. In order to do this, I designed a cascade of a low-pass filter (maintaining just the components in the range [0, 8000] Hz) and an high-pass filter (as before, maintaining just the [0, 20] Hz components).
For the first one I used the $ellip$ command in MATLAB, while for the second one I just implemented an IIR second order notch filter.

## 3.1 High-pass filter: specifications

For the highpass, I used the notch filter we've seen in the course with $M = N = 2$:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3}$$

and I set the parameters as in Table (1):

| Parameter | Value |
|---|---|
| $\theta_0$ | $0$ |
| $a_1$ | $-2r_{notch}cos(\theta_0) = -2 * r_{notch}$ |
| $a_2$ | $r^2_{notch}$ |
| $b_0$ | $-2cos(\theta_0) = -2$ |
| $b_1$ | $1$ |

Table 1: Review of high-pass parameters

remembering I'd used it in the interval [0, 20] Hz, so that $f_{3db,notch} = 20Hz$. The plot can be displayed in Figure (3).

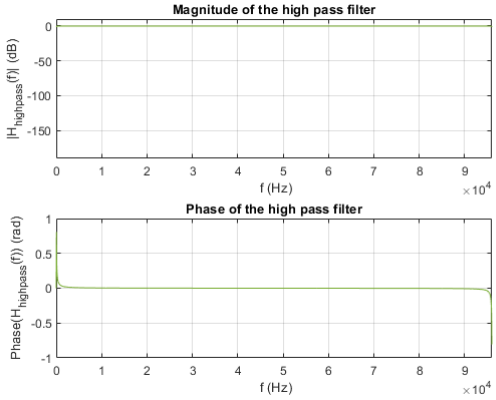



Figure 3: High-pass Notch filter

## 3.2 Low-pass filter: specifications

In this case, I used the command $ellip$ in MATLAB, that it's described in the documentation as:

$$[b, a] = ellip(n, Rp, Rs, Wp, ftype) \tag{4}$$

so, I set the parameters as follow in Table (2):

| Parameter | Description | Value |
|---|---|---|
| *n* | filter order | 15 |
| *Rp* | passband ripple | 0.1 dB |
| *Rn* | stopband attenuation | 100 dB |
| *Wp* | passband edge frequency | 8000 Hz |
| *ftype* | type of filter | 'low' |

Table 2: Review of low-pass parameters

With those parameters, there's a really small transition band and a quite regular trend in the passband. The plot of the frequency and the phase can be visualized in Figure (4).

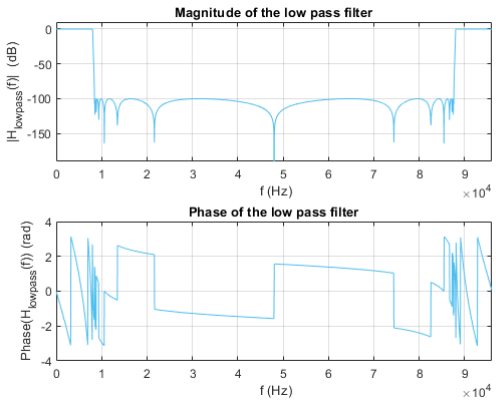



Figure 4: Lowpass filter

At the end of the computation, the final band-pass filter could be obtained as a cascade of the low-pass and the high-pass filters (Figure (5)). As before, it could be noted that as the band-pass filter has slightly high order, the transition band is quite small.

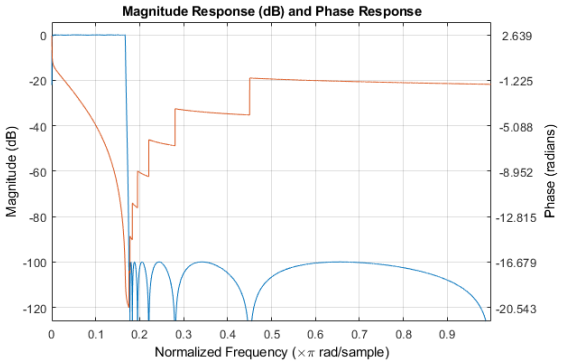



Figure 5: Cascade of the two filters

# 4 Demodulated signal

As requested, I've plotted the demodulated signals $X_1(f)$, $X_2(f)$ (figure 6) and the total output in time and DFT domain. $y(nT)$ was obtained multiplying the input signal by the pass-band filter, the two carriers and the respective gains (Figure 7a); $Y(f)$ is, of course, the DFT of the previous computation (Figure 7b). In the code I used two kinds of equivalent implementation to generate $y(nT)$: the main difference can be found in the realization of the cascade of the low-pass and high-pass; in the first one, I just used the command *dfilt.cascade* of MATLAB and then, using *filter*, I filtered the obtained $H(z)$ with the demodulated signals. In the second one, I manually implemented the cascade of the two filters. Both give, of course, the same result.

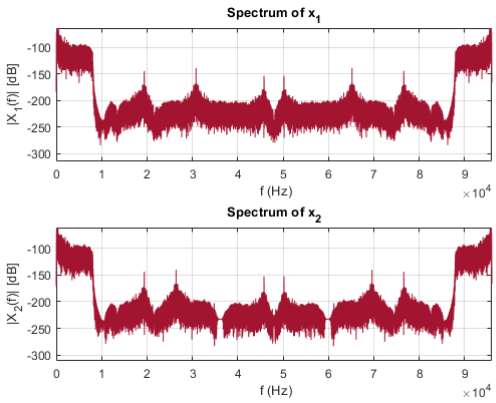
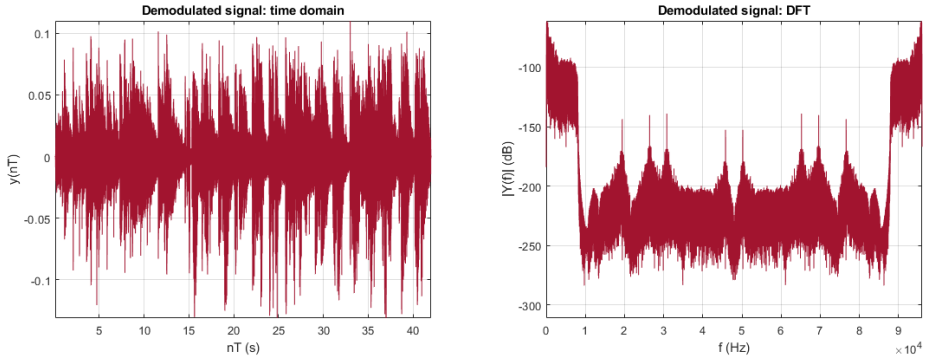



Figure 6: Spectrum of the demodulated signals




(a) Demodulated signal in time

(b) Demodulated signal in frequency domain

Just to make a comparison between the initial signal and the final one, I've also plotted the two signals overlapping in the time domain (Figure 8).

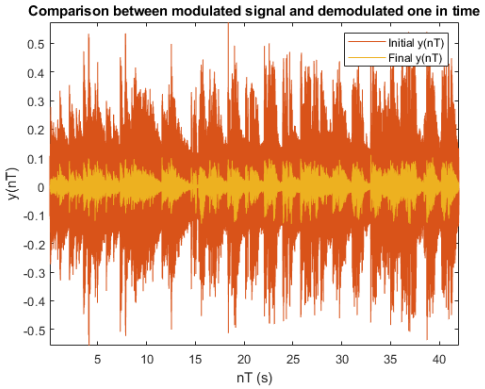



Figure 8: Initial y(nT) vs final y(nT)