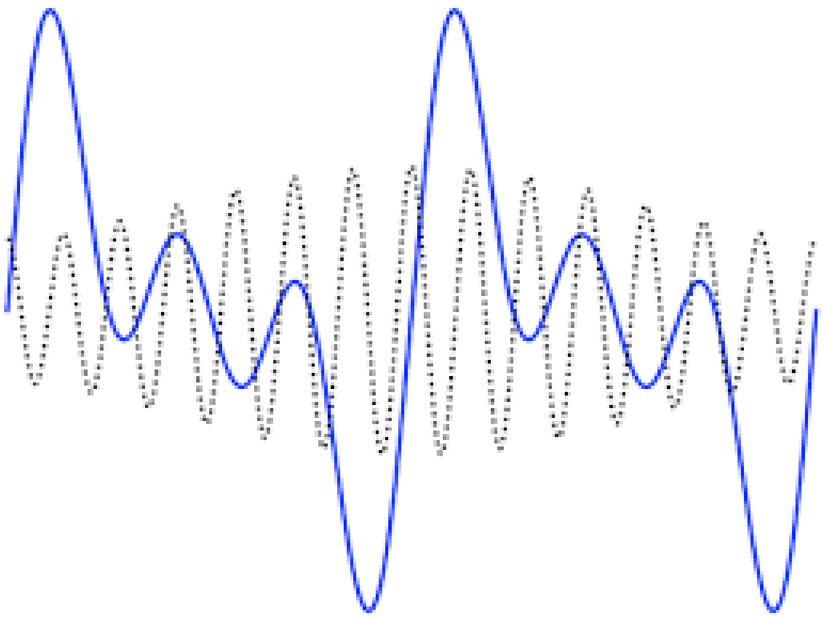
The University of Franche-Comté

SP

Signal Processing

Signal proccessing : lab work 1



Chiari Even
Legrand Maxime
M1 Compuphys 2022

Supervisor : Josipa Madunic

october 24^{th} 2022

Contents

1 Introduction

The aim of this labwork is to recreate a signal with the Fourier transform. With the knowledge that we have acquired in class we can understand the different steps of the digitization of a signal:
- First we have to sample the signal and choose our sampling rate.
- In second time we use the fast Fourier transform to describe our signal in frequency domain.
- Finally we rebuild the signal with the main information of the signal and after we will compare the original sound and digitized signal for different parameter.

2 Questions

2.1 Signal sampling visualization (1-4)

(1)

First of all one defines the discretized interval in order to evaluate the signal. This information is contained in the variable *time* of length *47936*. In other words, *time* contains all the samples.

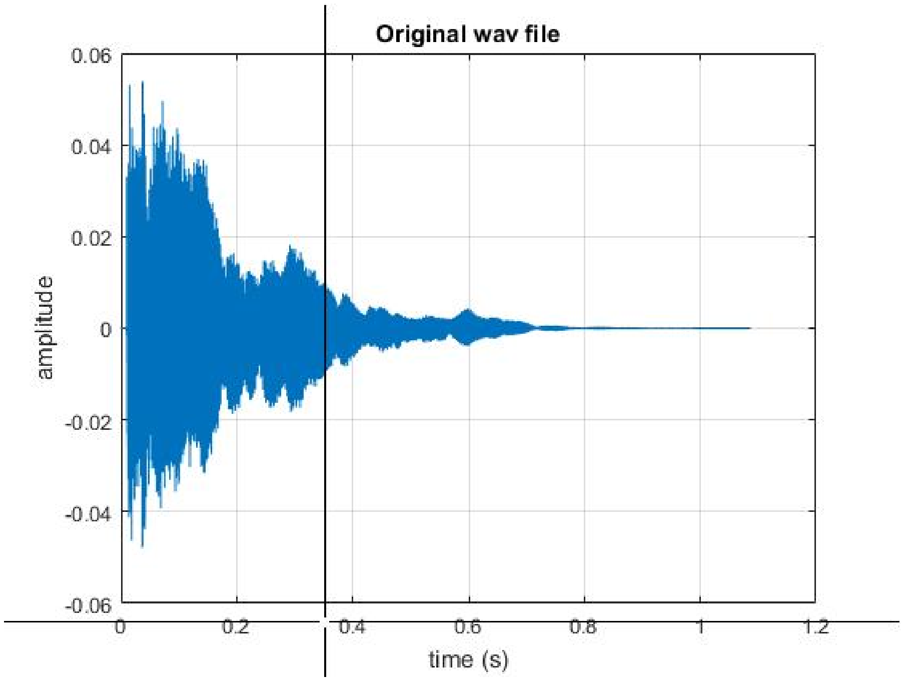
(2)

If one wants to get the step duration, it can simply be obtained through this line of code :

```
time = (0:Nb-1)/FS;
```

(3)

This plot shows the signal evaluated on *47936* points.



(4)

The *ginput* function allows us to select a range of data on the graph. It returns the selected values on the *x axis* and the corresponding ones on the *y axis*.

A code line to extract the length of a cycle could be :

```
CycleDuration = tt(2) - tt(1);
```

2.2 Fast Fourier transform (5-13)

(5)

This code part shows the Power Density of the signal (in dB scale).

(6)

The *fftshift* function puts the lowest frequencies (zeros) at the center of the plot.

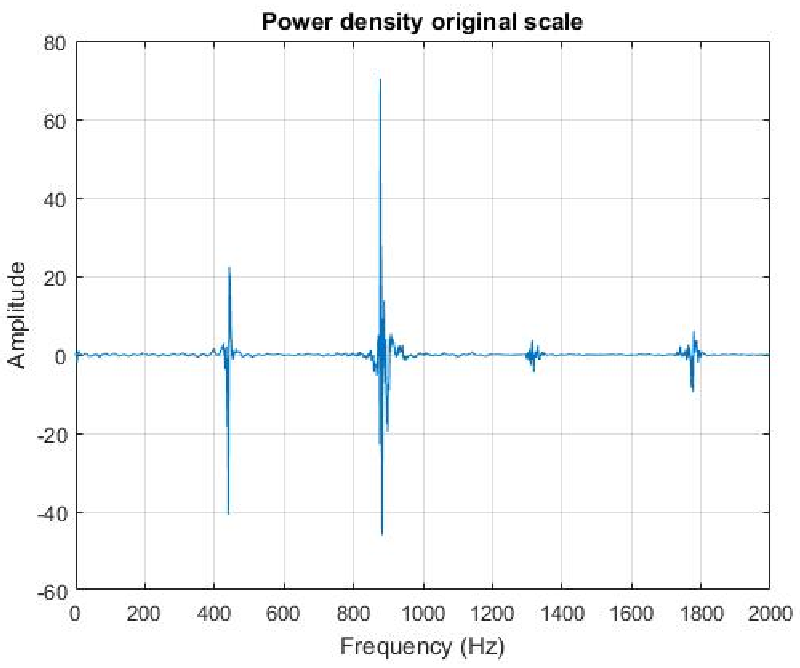
(7)

The *xlim* function sets bounds on the *x axis*, in our case we limit the *x axis* in the range $[0; 2000]$.

It is used here for readability, if we don't put this limit then the interesting part of the spectrum will be compressed near 0.

(8)

Below you will find the spectrum of the considered signal without the dB scale :

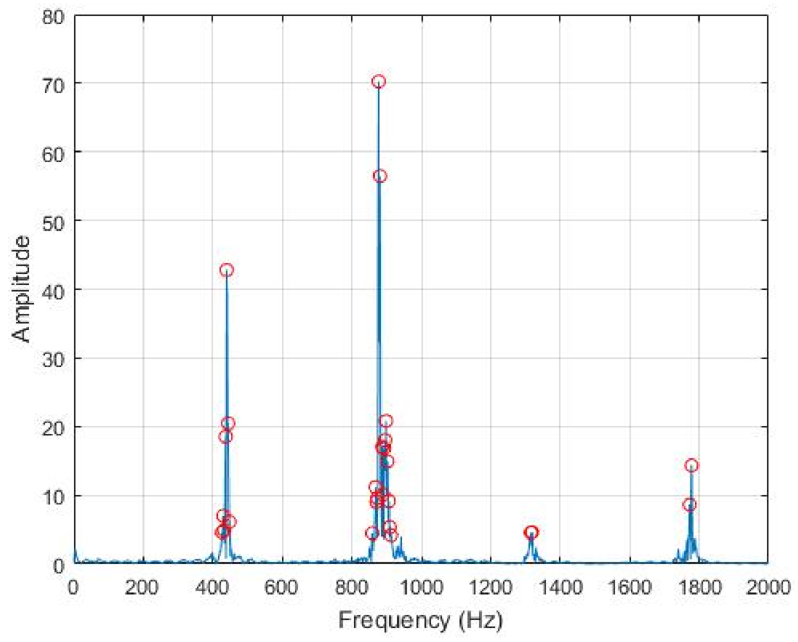


(9)

We can see the frequency of signal in blue and the location of resonant frequencies (red circles) plot with this function :

```
figure(3)
plot(frequency,abs(spectrum))
hold on
plot(frequency(locs),abs(spectrum(locs)),'or')
hold off
grid on
xlim([0 2000])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
```

We obtain the following plot :



(10)

While having changed *NbO* at 10, we see that the positive and negative peaks are a mirror of each other.

(11)

The function *Hold on/off* is used to keep the windows graph open, without this function we would've obtained two graph, one with red circle and an other with the blue signal.

(12)

In order to understand the function *findpeakfunction* help is very useful, with this we can compute :
-the maximum of this function.
-the minimum of this function.
-other parameter to use this function.
-understand how this function work.

(13)

The line :

```
freq_resonance = frequency(locs);
```

The list *frequency* contains the frequencies corresponding to the location indexes *locs.*

2.3 Signal reconstruction (14-20)

(13bis)

The following line corresponds to the response of a damped oscillator :

```
    synth_signal = synth_signal + spectrum(locs(i)) * exp(j*2*pi*freq_resonance(
        i)*time).*exp(-time/tau);
```

With :

- $spectrum(locs(i))$ the frequency amplitude.
- $exp(2j\pi freq_resonance(i) * time$ the oscillation term.
- $exp(-time/tau)$ the damping term.

(14)

There is a loop over i because we use i oscillators to modelise the resonances.

(15)

There are *NbO* damped oscillators responses summed.

(16)

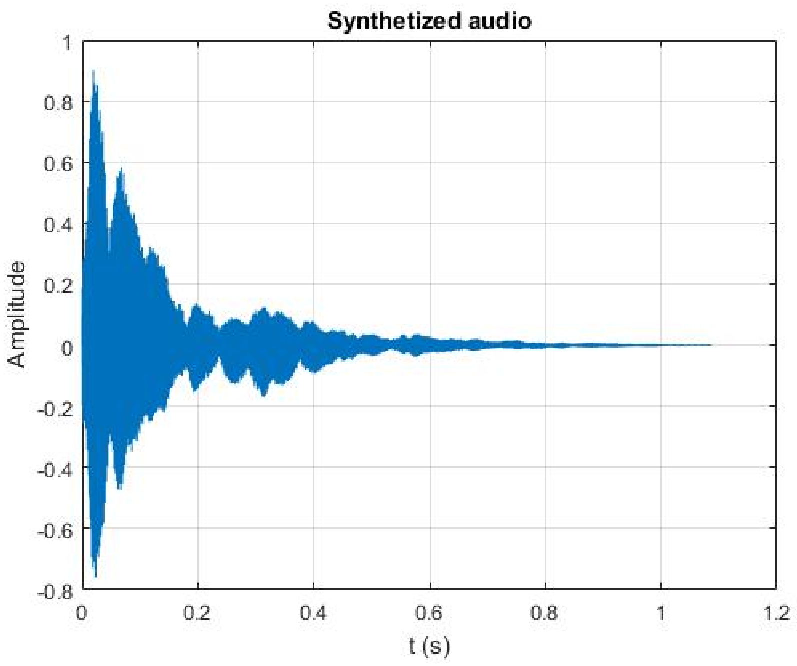
The assumption used is that the damping time τ is equal to 0.2 seconds.

(17)

We only keep the real part because the imaginary one has no real-life meaning.

(20)

The reconstructed signal below :

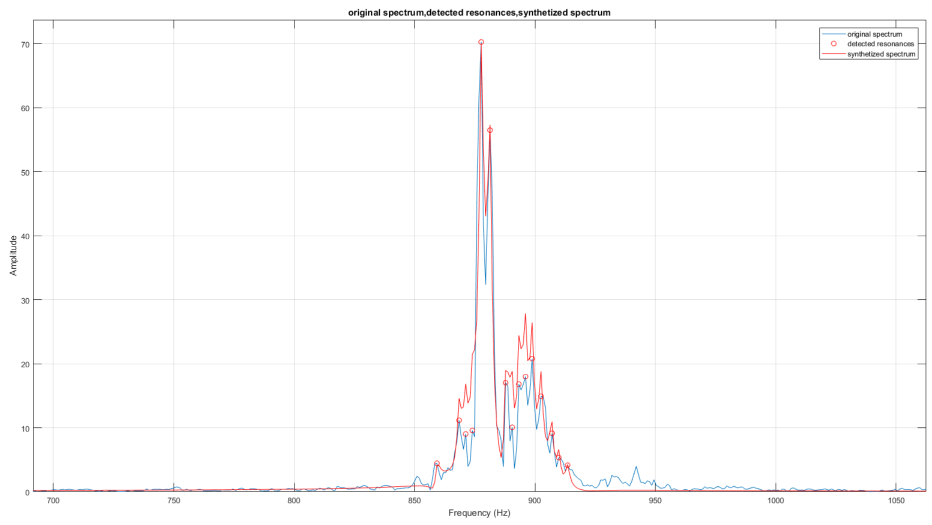


is smoothed compared to the original one.

2.4 Comparison of spectra (21-24)

(21 22)

By zooming on the spectrum, one can see some imprecisions regarding the peaks of the reconstructed signal. It can be improved in a way by rising up the amount of oscillators NbO and the damping time τ.



(23)

The higher NbO is, the more accurate will be the reconstructed sound compared to the inital one.

(24)

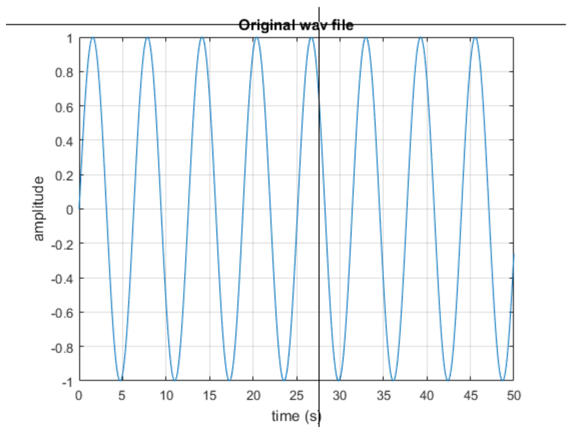
It doesn't work because we are only using the real part of the Fourier transform so we loose the information about the delay between two sounds.

(25) We create a signal with this code:

```
clear
close all
clc
FS = 4000;
x=linspace(0,50,FS);
signal = sin(x);
Nb = length(signal);
time = (0:Nb-1)/FS;
step = (0:Nb-1)/time;

figure(1)
plot(x,signal)
grid on
xlabel('time (s)')
ylabel('amplitude')
title('Original wav file')
[tt,aa] = ginput(2);
CycleDuration = tt(2) - tt(1);

df = FS/Nb;
frequency = (-Nb/2:Nb/2-1)*df;
spectrum = fft(signal);
spectrum = fftshift(spectrum);
spectrumdB = 20*log10(abs(spectrum).^2);

figure(2)
plot(frequency,spectrum)
grid on
xlim([0 2000])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
```

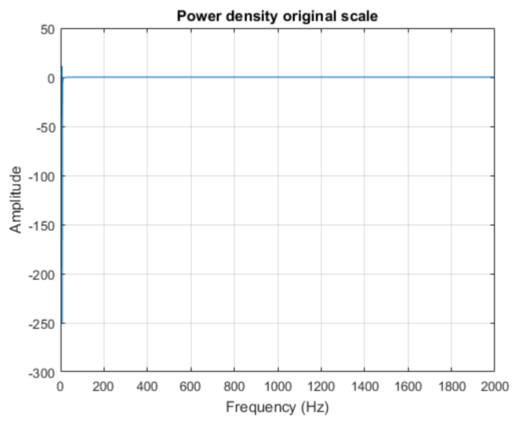
```
title('Power density original scale')
```

We can plot this signal:



This is a sinus function but don't have the good parameters but we can see the sinus sweeping in a range of frequency.

and its Fourier Transform :



We can see one resonance peak because the signal is purely numerical and we have chosen a high frequency step.