ENG5027: Digital Signal Processing

SCHOOL OF ENGINEERING

Assignment I. Fourier Transform

Author	GUI
Anton Saikia	2426828S
Gabriel Galeote Checa	2413232C

UNIVERSITY OF GLASGOW

GLASGOW, OCTOBER 2019

Contents

1 Objectives

In this assignment, it was required to make a vocal audio recording and process it. The audio recorded had to be an uncompressed WAV file and also have a minimum sampling rate of 44kHz. The audio signal was then transformed into frequency domain in python, and the voice quality was improved enhanced by manipulating the voice harmonic amplitudes using the Fast Fourier Transform command.

2 Methodology

The audio was recorded using a standard microphone from the Digital Signal Processing laboratory and was taken in a slightly noisy environment. The audio sample recorded consisted of several vowels and consonants. Therefore, the frequency response will be covering a greater range of frequencies with more peaks present in the frequency spectrum. Therefore, there will not just be one fundamental frequency but a range of fundamental frequencies for different combinations of vowels and consonants.

The first step in order to start the analysis and processing of the voice is to show the time domain representation of the signal in linear axis. The sound recording taken in wav 44 KHz and 16 bits is loaded into python using the library *scipy* using the *wav.read()* command. In Fig. 1 the voice recording is represented in time domain. Here, it can be seen the peaks of the voice recording corresponding to the pitches of the words and syllables.

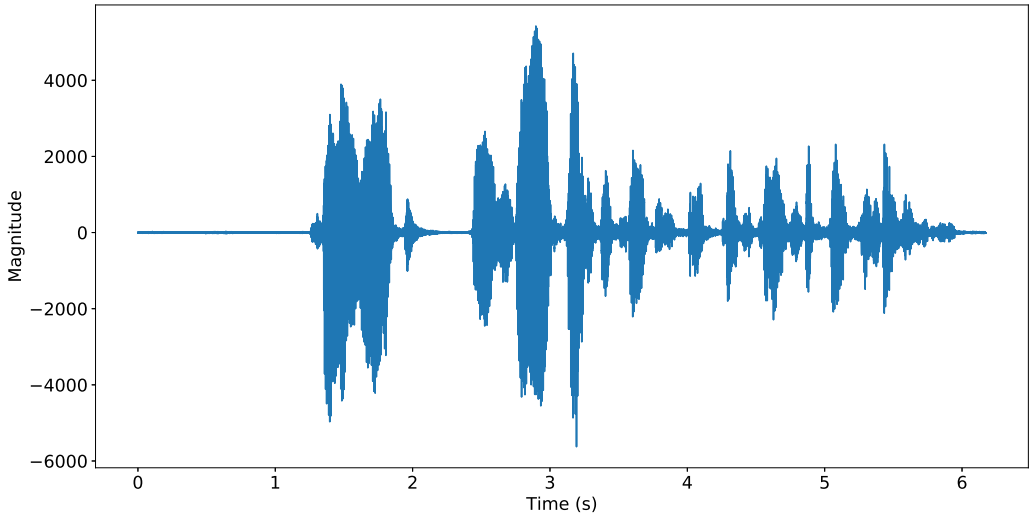

Figure 1: Time domain representation of the sound recording. In *Y-axis* it is represented the values of the input signal that depends on the sensitivity of the microphone and in the *X-axis* the time scale that is sampled in 44KHz.

As it can be seen from Fig. 1, the signal in time domain shows the sound profile

obtained from the entire recording in approximately 6 seconds. The distribution of peaks along the time axis and their dependency on each word and vocal intensity can be seen. It is important to understand that the profile is irregular because some sounds are stronger than others and that it is affected by the vocal anatomy of the speaker as well as the distance from the microphone among other factors. From the time domain representation we can get a general idea of the frequency spectrum of the signal and a brief analysis of the cleanness of the signal as well as the possible frequency anomalies in it. For example, in Fig. 2 it can be observed in the zoomed section of the non-speaking part of the signal that there are both high and low frequency noises. After the 6 seconds in the time axis, the signal should ideally not show any frequency but there is a low-frequency noise followed by a high-frequency noise that must be considered as just noise and not as data of any interest.

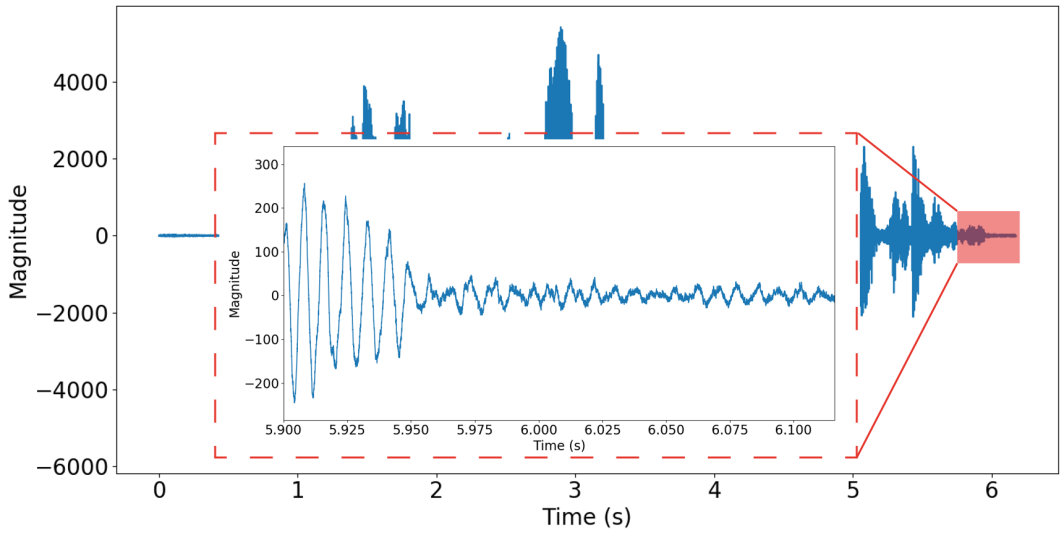

Figure 2: Zoom in to a non speaking section of the time domain representation of the sound recording. High frequency noise can be seend following the signal and low frequency noise after second 6.

The next step is to calculate the Fourier transform of the signal to analyse the frequency spectrum. This will provide a representation in frequency domain of all the frequencies of the recorded audio. Using the FFT (Fast Fourier Transform) command, the audio signal can be transformed into frequency domain and plotted in logarithmic scale. Using this frequency spectrum, the fundamental and harmonic frequencies of the signal can be analysed and manipulated for improvement. As described earlier, there will not be just one fundamental frequency but a range of several fundamental frequencies corresponding to different sounds of the recorded voice. The general convention is to consider any major peaks before 1 KHz as fundamental frequencies. The frequency spectrum is shown in Fig. 3.

In the frequency spectrum it can be observed that the most well defined peaks are between 100 Hz to 1 KHz and thus, this is the frequency range that is considered to have

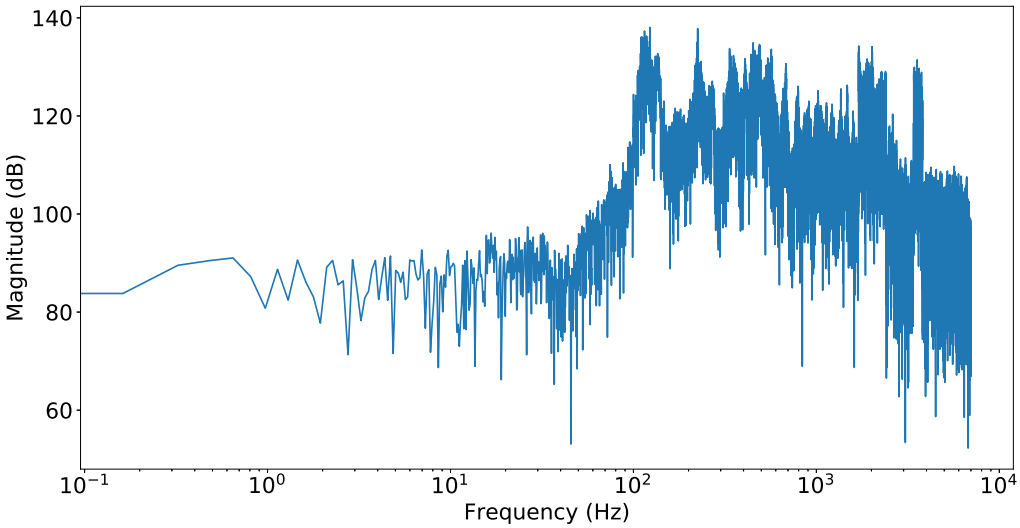

Figure 3: Frequency Spectrum with logarithmic axis.

all the fundamental frequencies. In addition, according to some studies, male voice is normally considered between 80 or 100 to 800 or 1000 Hz (Heylen et al., 2002; Svec and Granqvist, 2010).Therefore, it is logical that the frequency range from 100 to 1000 Hz could be the range of fundamental frequencies for the voice recorded. In Table 1 it is summarised all the peaks that were considered as fundamental frequencies.

Peak	Frequency band (Hz)
1	95 - 150
2	200 - 270
3	315 - 380
4	630 - 720

Table 1: Most representative peaks in the fundamental frequencies band between 100 to 1 KHz.

Due to the periodicity of the frequency response of the signal, there are infinite copies of the fundamental frequency as multiples of real numbers. The formula that describes this harmonic behaviour is the presented in Eq. 1.

$$\text{Harmonic(n)} = n \cdot FundamentalFrequency \tag{1}$$

Where n is a natural number that multiplied by the fundamental frequency gives the harmonic frequency band of all the frequencies of the voice recorded (Smith et al., 1997).

The voice recording of this work is the sentence: ”Hello world, this is a voice recording for digital signal processing”. As explained earlier, the fundamental frequency concept must be understood as a range of frequencies. For example, the vowel ”o” has a lower fundamental frequency than the vowel ”i” and consonant ”s”. However, as there are

many consonants and vowels, equation 1 is not suitable for this type of audio signals.For preliminary analysis, the frequencies from 100Hz to 1KHz were isolated and reproduced. Normally, it is considered that this frequency range comprises all the male fundamental frequencies. The voice in this frequency range appeared to be deeper and having more bass than the original recording.

The next step was to find the band for harmonics in the 1 - 2.5 KHz range. This frequency band was isolated for analysis by removing the rest of frequencies. On hearing this frequency band, the audio appeared to be slightly higher pitched as well as some noise. To make the sound clearer, various smaller frequency bands were taken between the 1 - 2.5 KHz range to narrow down the range and obtain a band where the voice was loudest and the noise was minimum. By process of elimination, the 1.7 – 2.4 KHz range appeared to be the optimal range.

Subsequently, the entire frequency range was taken and the harmonic band between 1.7 - 2.4 KHz was amplified by different factors from 2 to 10. At $a_1 = 2$ (amplification factor), the audio sounded clearer than the original recording, however the voice sounded quieter. With the increase of a_1, the audio began to sound louder and slightly clearer while it also sounded relatively softer than the original recording. However, from $a_1 = 8$ onwards, the voice sounded progressively shrill. Hence, an amplification factor of 8 was used for this harmonic band.

A similar process was used to determine the harmonic band in the 3 - 4 KHz range. The isolated vocals in this frequency range sounded even more high pitched. After thorough analysis, the clearest sounding frequency range for this harmonic band was found to be between 3.4 - 3.8 KHz and the corresponding amplification factor was as 8. The frequency response of the entire audio signal was then taken and the harmonic bands mentioned above are separately amplified by factors of 8. Furthermore, to reduce high frequency noise all frequencies above 7 KHz were removed. Finally, the modified audio signal is reconstructed into time domain and exported as 16 bit WAV .

3 Results and Discussion

After listening to the the new audio sample, it was observed a moderate improvement in the vocal quality and enunciation with almost negligible noise in the background. The audio also sound more balanced and equalised.

It is important to note that since the recorded audio sample consisted of a combination of many vowels and consonants, it is very likely that across the frequency spectrum, where there are harmonics of the various vocal sounds, there are also some sort of noises present at that same frequency. So, whenever the amplitudes of any frequency band are increased, the noise present at those frequencies will also be amplified. The frequency bands amplified in this assignment were all taken such that they appeared to have the minimum audible noise in their respective frequency ranges and such that the resultant audio sounded better than the original recording.

For further analysis, the spectrogram of the signal pre and post processing were shown in order to see the distribution of frequencies in time. In the spectrogram, it can be seen the noises in the high frequency (after 15 KHz) and the distribution of the frequencies depending on the word spoken. Besides, it can be also observed the effect of the processing as an amplification of the frequency bands of 1.7 - 2.4 KHz and 3.4 - 3.8 KHz.

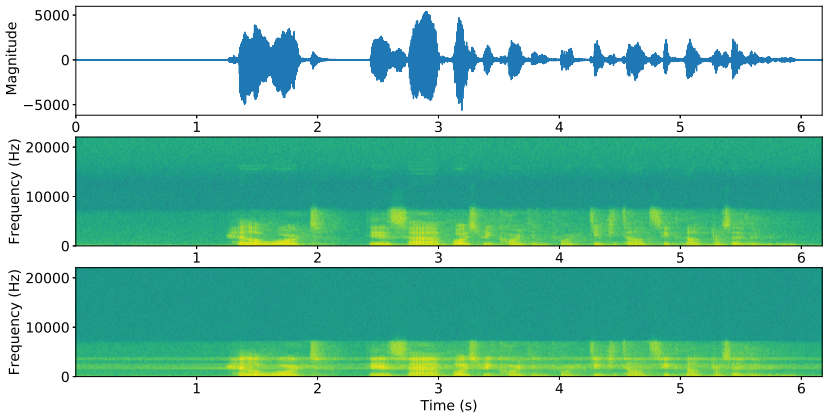

Figure 4: time domain representation of the signal with the spectrogram of both, original and processed signals from top to bottom.

4 Bibliography

Heylen, L., Wuyts, F., Mertens, F., De Bodt, M., and Van de Heyning, P. (2002). Normative voice range profiles of male and female professional voice users. *Journal of voice*, 16(1):1–7.

Smith, S. W. et al. (1997). The scientist and engineer's guide to digital signal processing.

Svec, J. G. and Granqvist, S. (2010). Guidelines for selecting microphones for human voice production research. *American Journal of Speech-Language Pathology*.

Python Code

```
#   Code for the Assignment 1 of Digital Signal Processing
#   Authors: Gabriel Galeote-Checa & Anton Saikia

import matplotlib.pylab as plt
import numpy as np
import scipy.io.wavfile as wav
import wave
plt.rcParams.update({'font.size': 16})

fs, soundwave = wav.read ('original.wav') # Read sound signal

# Representation of Time Domain
t = np.linspace(0, len(soundwave)/fs, len(soundwave)) # time vector

# ---- Figure 1 ------
plt.figure(1)
plt.plot(t, soundwave)
plt.xlabel('Time (s)')
plt.ylabel('Magnitude')

# ---- Frequency Domain Calculations ----
fftSoundWave = np.fft.fft(soundwave)
f = np.linspace(0, fs, len(fftSoundWave)) # Full spectrum frequency range

fftSoundwaveHalf = fftSoundWave[:len(fftSoundWave) // 2] # REMOVE the half of
    the mirrored spectrum
fHalf = np.linspace(0, fs / 2, len(fftSoundwaveHalf)) # Half spectrum
    frequency range

# ---- Figure 2 ----
# FFT signal in linear and logarithmic axis
plt.figure(2)
plt.plot(fHalf, 20*np.log10(abs(fftSoundwaveHalf)))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude (dB)')
plt.xscale('log')

# ---- Voice Processing ----
"""
Description:
```

```
The processing consists of two parts: amplification of the harmonic between
    300 and 355 Hz and attenuation of the
harmonic between 400 and 460 Hz. The reason is to give same importance
    (magnitude range) to the first 4 harmonics
of the signal

We would need to convert from frequency to the exact sample of the array
    where we have to start operating. Therefore,
the conversion needs to be parsed to int obtaining the position of the fft
    signal where we apply the operations.

"""

fLow = 7000 # Hz
fHigh = fs/2 # Hz
fAmplifyL1 = 1700 # Hz
fAmplifyH1 = 2400 # Hz
fAmplifyL2 = 3400 # Hz
fAmplifyH2 = 3800 # Hz
a1 = 8 # amplification 1st harmonic
a2 = 8 # amplification 2nd harmonic

# Conversion from frequency to sample
fL = int((fLow / fs) * len(fftSoundWave))
fH = int((fHigh / fs) * len(fftSoundWave))

fAL1 = int((fAmplifyL1 / fs) * len(fftSoundWave))
fAH1 = int((fAmplifyH1 / fs) * len(fftSoundWave))

fAL2 = int((fAmplifyL2 / fs) * len(fftSoundWave))
fAH2 = int((fAmplifyH2 / fs) * len(fftSoundWave))

FSW = fftSoundWave # pass fftSoundWave to other variable to not overlap the
    original signal

# Operations

FSW[fL:fH] = FSW[fL:fH] * 0
FSW[len(FSW) - fH:len(FSW) - fL] = FSW[len(FSW) - fH:len(FSW) - fL] * 0

FSW[fAL1:fAH1] = FSW[fAL1:fAH1] * a1
FSW[len(FSW) - fAH1:len(FSW) - fAL1] = FSW[len(FSW) - fAH1:len(FSW) - fAL1] *
    a1
```

```
FSW[fAL2:fAH2] = FSW[fAL2:fAH2] * a2
FSW[len(FSW) - fAH2:len(FSW) - fAL2] = FSW[len(FSW) - fAH2:len(FSW) - fAL2] *
    a2

# Reconstruction of the signal
timeFilteredSoundWave = np.fft.ifft(FSW)
timeFilteredSoundWave = np.real(timeFilteredSoundWave)

# ---- Figure 3 ----
# Represents the result signal in linear and logarithmic axis.
plt.figure(3)
plt.plot(fHalf, 20*np.log10(abs(FSW[:len(FSW) // 2])))
plt.xscale('log')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude (dB)')

# ---- Save Processed Sound Recording ----
f = wave.open(r"improved.wav", "wb")
# 1 channel with a sample width of 2 and frame rate of 2*fs which is standard
f.setnchannels(1)
f.setsampwidth(2)
f.setframerate(2*fs)
timeFilteredSoundWave=timeFilteredSoundWave.astype(int)
f.writeframes(timeFilteredSoundWave.tostring())

# --- EXTRA ----
# Here, the spectogram is presented to see the variation of frequencies with
    respect to the time domain signal.
# This representation is very useful to analyse the profile of the sound and
    the frequencies distribution
# along the time. Thus, frequency over time is represented.
plt.figure(4)
plt.subplot(311)
plt.plot(t, soundwave)
plt.xlabel('Time (s)')
plt.xlim([0, len(t)/fs])
plt.ylabel(' Magnitude')
plt.subplot(312)
powerSpectrum, frequenciesFound, time, imageAxis = plt.specgram(soundwave,
    Fs=fs)
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.subplot(313)
powerSpectrum, frequenciesFound, time, imageAxis =
```

```
    plt.specgram(timeFilteredSoundWave, Fs=fs)
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')

plt.show()
```