POLITECNICO DI TORINO

Electronic and Communications Engineering

Assignment Report 6 - Intro to Digital Audio Processing

Applied Signal Processing Laboratory

01TUMLP

Academic Year 2019/20

Gloria DAL SANTO
s245220

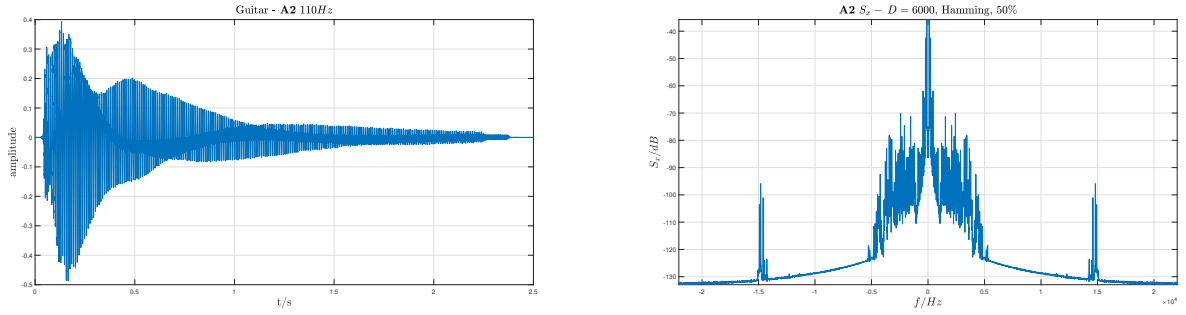
Exercise 1

The first signal to be analyzed is a record of the A2 note played on a guitar (Figure 1.a). The information of the signal are reported in Table 1.

	Description
Duration	2.5 s
File Format	WAV
Sample Rate f_s	44100 Hz
Total Samples N	110250
Channels	2
Bits per Sample	16

Table 1: Audio file "A2_guitar.wav" information

To estimate the spectrum of the signal applying the Welch periodogram I used the function pwelch with segment length equal to $D = 6000$ samples, a frequency resolution of $7.35Hz$ and with 50% overlap. The estimated spectrum is shown in Figure 1.b where it can be noticed that the main components are centered around $\pm 110Hz$, the frequency associated to the note played by the guitar. It can be seen that there are also some components at higher frequencies around $15kHz$.



(a) *Time domain plot*

(b) *Welch periodogram*

Figure 1: Exercise 1 - *Guitar A2 note signal*

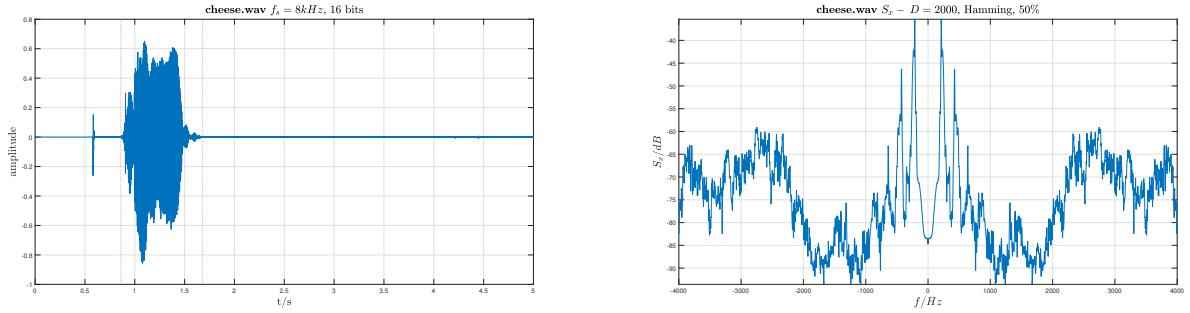
To find the pitch I first computed the normalized autocorrelation with xcorr function, and then I evaluate the distance between the zero lag and the first peak, found with function findpeaks. Finally, I computed the inverse of the lag distance and multiplied it by f_s to get the corresponding value in frequency unit:

$$f_1 = 109.43Hz.$$

In theory, the $A2$ note should correspond to a frequency of $110Hz$, however the sound produced by the guitar appears to be slightly detuned. This error could be related to the dynamic and to the duration of the note played or simply to an unavoidable tuning error of the string played.
For the second part of this exercise I first created an audiorecorder object of my voice saying the word "cheese" and I wrote it into a .wav file with the audiowrite function (Figure 2.a). The information of the recorded audio are reported in Table 2.
I repeated the procedure to estimate the spectrum used for the first signal modifying only the number of samples of each segment, which now is $D = 2000$ leading to a frequency resolution of $4Hz$.
In Figure 2.b it can be seen that there is a fundamental component around $215Hz$ and some distinguishable harmonics at higher frequencies but the are also other frequency components along the entire frequency axis.

	Description
Duration	5 s
File Format	WAV
Sample Rate f_s	8000 Hz
Total Samples N	40000
Channels	1
Bits per Sample	16

Table 2: Audio file "cheese.wav" information



(a) *Time domain plot* (b) *Welch periodogram*

Figure 2: Exercise 1 - *recorded "cheese"*

To estimate the pitch I first cut the audio file to keep only the part containing the voice. Figure 3 shows the signal cut where the first part related to the affricative sound is followed by the quasi-periodic sound at around $1s$. Repeating the procedure to find the pitch using xcorr and findpeaks functions, I obtained the following estimated value:

$$f_2 = 216.22Hz. \tag{1}$$

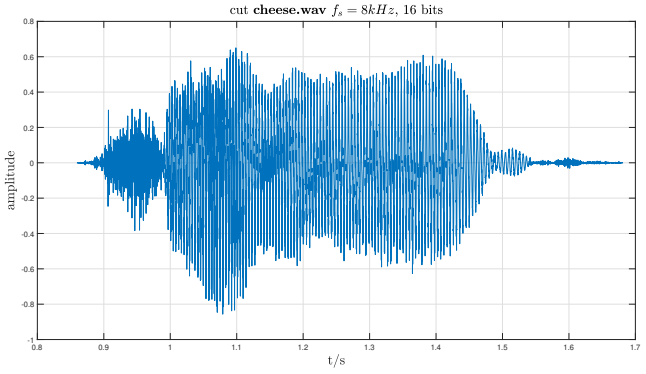


Figure 3: Exercise 1 - *recorded "cheese"*

Exercise 2

The signal to be analyzed in this exercise is a record of the A major scale played on a guitar (Figure 4). The information of the audio file are reported in Table 3.

	Description
Duration	10 s
File Format	WAV
Sample Rate f_s	44100 Hz
Total Samples N	441000
Channels	1
Bits per Sample	16

Table 3: Audio file "A_major_scale.wav" information

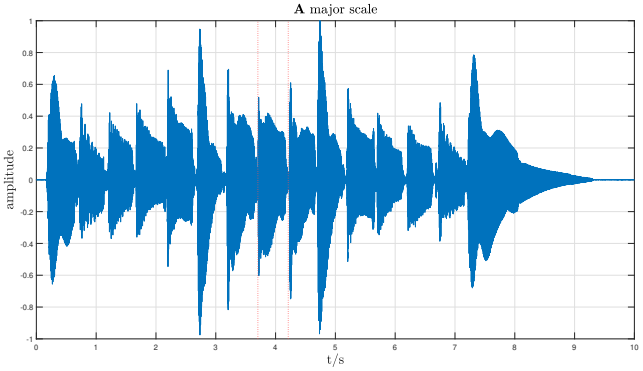
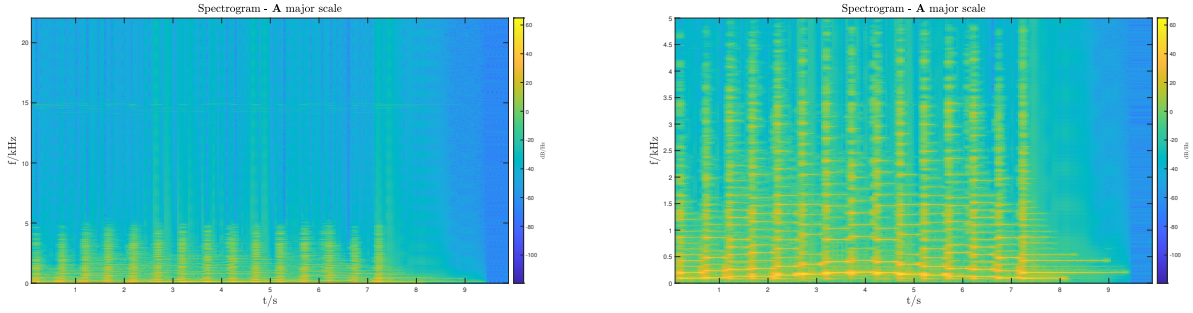


Figure 4: Exercise 2 - *A_major_scale.wav audio signal*

Figures 5.a and 5.b show the spectrogram of the signal in two different frequency ranges, evaluated using the spectrogram function with a Hamming window of size $M = 10001$ samples, 50% overlap and 8000 points FFTs.



(a) *full range spectrogram* (b) $0 - 5kHz$ *range spectrogram*

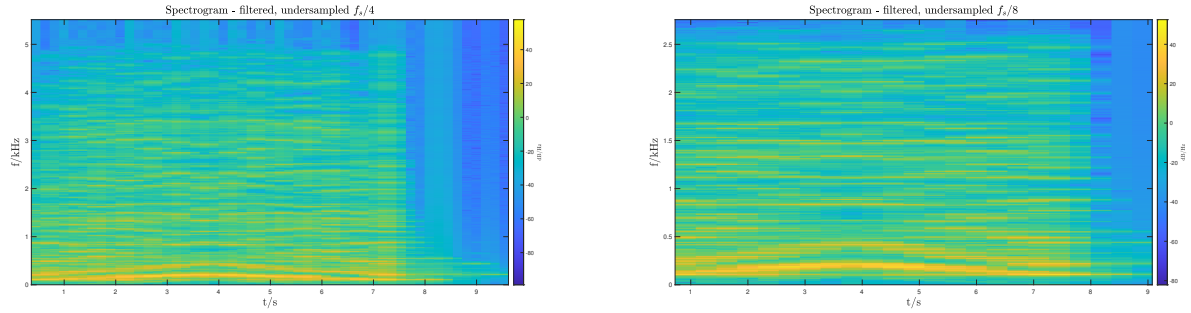
Figure 5: Exercise 2 - *Spectrogram of "A_major_scale.wav" with* $f_s = 44100Hz$

I cut the signal in the interval that goes from $t_0 = 3.7s$ to $t_1 = 4.2s$ to select only the higher note and I evaluated its corresponding frequency applying the function findpeaks in the same way I did for the first exercise, obtaining the following value:

$$f_{A2} = 222.72Hz.$$

The requirements for the low-pass filter to be applied on the whole signal were: a cutoff frequency $f_c = 5kHz$, a transition band of $B_T = 500Hz$ and a minimum stop band attenuation $A_{s_{min}} = 50dB$. To meet these requirements I choose a Hann window and to compute its impulse response and to filter

the signal I used the functions fir1 and filter.
Figure 6.a depicts the spectrogram obtained from the filtered signal, undersampled with a sampling frequency of $f_{s1} = f_s/4 = 11025Hz$. From Figure 5.a it can be seen that almost all the frequency components of the original signal were below $5kHz$ and, as a consequence, they were not distorted by the low-pass filter. Moreover, the new sampling frequency is above $2 \cdot f_c$, which ensures negligible aliasing effect. As a result, the audio signal sounds virtually identical to the original one.
Repeating these last steps, I filtered the original signal with a low pass filter similar to the previous one apart from the cutoff frequency, which now is $f_c = 2.5kHz$. With this new value of the cutoff frequency, the effect of the filter is no more negligible since it distorts the spectrum in a range dense of harmonics. Indeed, the signal sounds muffled, less rich in higher harmonics even though the played notes are still clearly distinguishable. Figure 6.b depicts the spectrogram obtained from the filtered signal, undersampled with a sampling frequency of $f_{s2} = f_s/8 = 5512.5Hz$.



(a) $f_{s1} = f_s/4 = 11025Hz$, $f_c = 5kHz$

(b) $f_{s2} = f_s/8 = 5512.5Hz$, $f_c = 2.5kHz$

Figure 6: Exercise 2 - *Spectrogram of "A_major_scale.wav" filtered*

Exercise 3

In this exercise it was asked to write a custom function that computes the spectrogram of the input vector x. The syntax of the function is the following:

[S, t, f]=my_spectrogram(x,M,L,N,fs,plot_flag)

The way I implemented the function follows the instructions given in the assignment description.
To check that the custom function worked properly, I compare it to Matlab function spectrogram with the .wav file ”A_major_chord" used in Exercise 2 as input signal and with window size $M = 10001$ samples, number of overlapping samples $L = 8000$ and a number of FFT points $N = 16384$. Figure 7 shows the spectrogram obtained by the custom function, that is identical to the one obtained with the Matlab function.

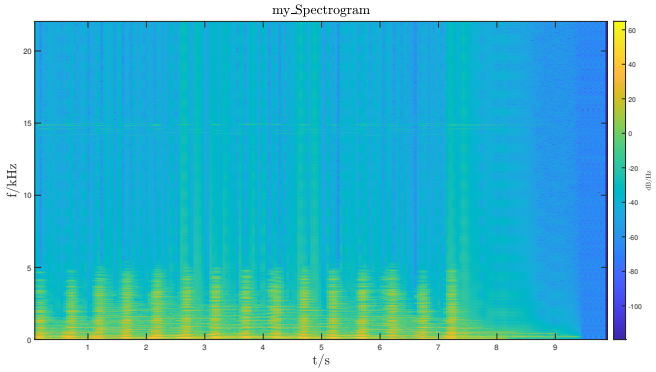


Figure 7: Exercise 3 - *Spectrogram of A_major_chord*

Exercise 4

In this exercise it was asked to write a custom function that generates by means of additive synthesis a tone whose partials are defined by the input parameters. The syntax of the function is the following:

[tone, t]=fnote(a,f,p,T,fs)

where tone is the output signal of duration T sampled at sampling frequency fs.
To test the function I generated a sequence of tones, each of length $2s$, which synthesize the vowels in the order "A-E-I-O-U". To set the frequency vector f and the amplitude vector a I referred to the proposed table made by Helmholtz choosing the frequency in eq.(1) as fundamental. The obtained signal does not sound exactly as a natural voice, as it was expected from the fact that it is generated with a small range of harmonics and without any kind of envelope.
Then I synthesize the sound of a bell using again the custom function. Figure 8 shows the obtained signal which was multiplied by an exponential envelope of the form $e^{-t/0.9}$. The frequencies of the partials refers to the following multiples of the fundamental frequency $f_0 = 440Hz$:

$$\text{f} = [1\ 2\ 2.4\ 3\ 4.5\ 5.33\ 6]$$

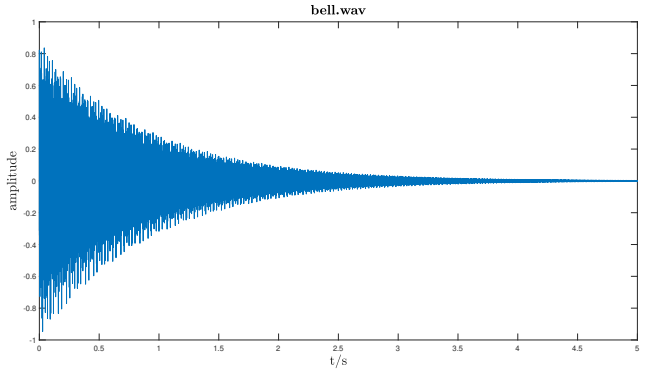


Figure 8: Exercise 4 - *Bell sound generated by additive synthesis*

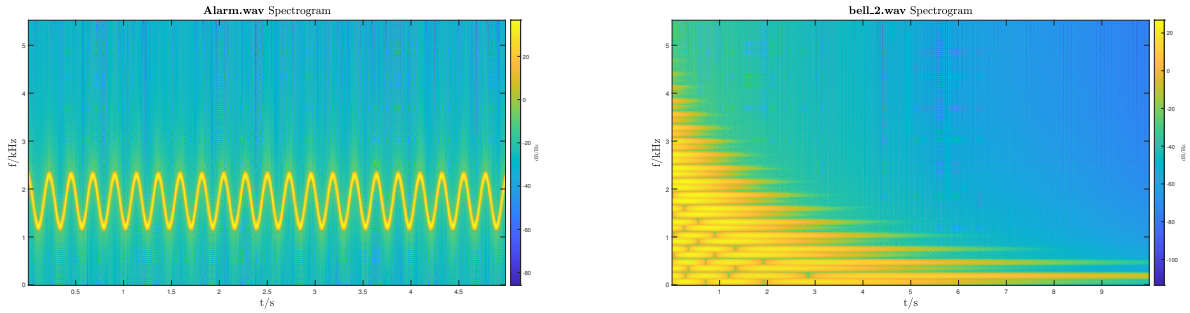
Exercise 5

To synthesize the sound of an alarm and that of a bell using the Frequency Modulation (FM) synthesis, I sampled at $f_s = 11025Hz$ two different signals defined as follows:

$$y_1(t) = Acos[2\pi f_c t + I_0 sin(2\pi f_m t)] \tag{2}$$

$$y_2(t) = Aa(t)cos[2\pi f_c t + I_0 a(t) sin(2\pi f_m t)] \tag{3}$$

The signal in eq.(2) corresponds to the alarm sound and its spectrogram is illustrated in Figure 9.a. From the figure it can be seen that the carrier frequency $f_c = 1760Hz$ is modulated by the modulator term $I_0 sin(2\pi f_m t)$, whose frequency is $f_m = 4.4Hz$, so that it changes periodically around $1760Hz$ in a sinusoidal pattern.
Figure 9.b shows a more complex spectrogram which is related to the bell sound described in eq.(3), where the carrier freqnuency is $f_c = 200Hz$ and the modulator frequency is $f_m = 280Hz$. Both the amplitude of $y_2(t)$ and the amplitude of the modulator signal $I_0 sin(2\pi f_m t)$ are multiplied by the exponential envelope function $a(t) = e^{-t/2}$. The effect of $a(t)$ applied to the modulator is that of attenuating exponentially all the higher harmonics as it can be seen in the spectrogram.



(a) *Alarm sound* (b) *Bell sound*

Figure 9: Exercise 5 - *FM synthesis*

Exercise 6

In this last exercise it was asked to generate a chord progression applying the additive synthesis to create each chord and the subtractive synthesis to generate each single note. To this purpose, I defined three local functions whose syntaxes are as follows:

[y,t]=chord_gen(fs,T,note,oct,min) (4)

[y,t]=note_gen(f0,fs,T) (5)

h=filter_imp(f0,fs) (6)

The function in eq.(5) is the one responsible for the note generation. The input parameters are the fundamental frequency f_0, the sampling frequency f_s and time duration of the note T. The starting signal is a Pulse Width Modulation (PWM) signal generated from a sawtooth, with fundamental frequency f_0, which is then compared to a sinusoidal signal at frequency $f_0/240$ that operates as low frequency oscillator. The subtractive synthesis is achieved by filtering the PWM signal with a FIR filter at cutoff frequency $f_c = f_0 \cdot 17/2$, transition band $B_T = 15f_0$ and which applies the Bartlett window. For the filtering operation I defined a separate function, whose syntax is the one in eq.(6), and I used it in note_gen function. Then in chord_gen function I created a cell array containing all the multiplicative factors in terms of semitone distance of each note from the A4 note

at $440Hz$. Each time the function is called, it will check through the cell array to find the frequency of the fundamental corresponding to the note given in the note argument and it generates it calling the function note_gen. Function chord_gen allows also to generate the flat (♭) or sharp (♯) pitch modification and it distinguishes between minor and major chords using the input argument min. If min is set to 1 then the triad will be a minor one, otherwise it will be always a major. Finally, the function chord_gen generates also the third note, according to the argument min, and the fifth note, and it outputs the chord as a sum of the three notes.
Each chord of a chord progression is then multiplied by the ADSR envelope function depicted in Figure 10 and then they are concatenated to create a single vector so that to save the chord progression as a single .wav file.

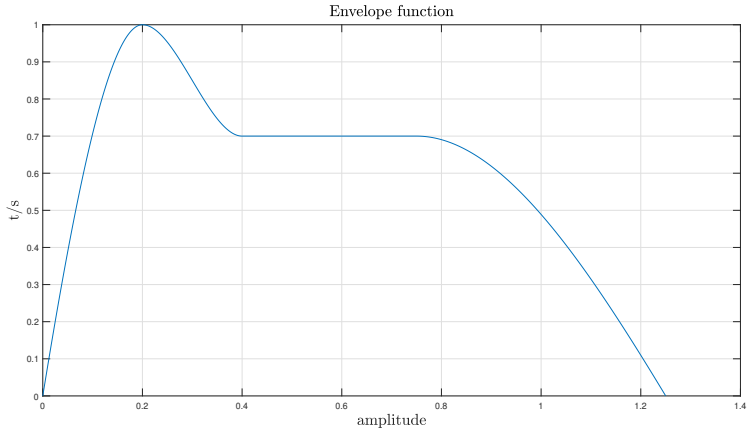


Figure 10: Exercise 6 - *ADSR envelope*