Département Informatique
Formation Initiale

Musical transcription with Least Mean Square algorithm

Intership in Hachinohe Kosen in Japan

Pahoua YANG

2012

Tutors :
M. Lebegue, M. Hirakawa, M.Kudoh

Acknowledgements

First of all, I would like to thank the IUT A's teachers of the informatics department to give me there knowledge during these two years of my formation in Lille1. Then, I am also thankful for the Japanese government and teachers who give the opportunity for foreigners students to make their intership in one of their Kosen.

Besides, I would like to thank especially all my tutors of this intership. To begin, I thank Mister Lebegue, my tutor from France and responsible of international relation, who gives me the chance to make my intership in Japan. Then, Mister Hirakawa, the responsible of international relation in Japan, who find me a place in Hachinohe kosen, even if the foreigner's dormitory was full. Moreover, Mister Kudoh, my intership project tutor, who gives me his time to explain the project for my intership in Hachinohe kosen, and finally, Mister Hosakawa, who explains us everything about the schedules, the classrooms and all the events in the Kosen.

At last but not least, my thanks are also for Mister Rigal and Miss Morita, my Japanese teachers in France and all my Japanese teachers who spent their times to teach me the language and the culture of Japan. Then, about Japanese culture, I would like to thank the Ikebana club where I learnt a lot thing about the art of flower arrangement. And finally, I am very glad to thank all the students of the E5 class in Hachinohe Kosen and foreigner students in the dormitory who make my intership enjoyable thanks to their friendship.

Abstract

For my last semester of my “DUT Informatique de gestion” in the University of Lille1, I made an intership during three month. The goal of this intership is to evaluate the informatics skills that I earned during my schooling.

I decided to do this intership in Japan because it was an opportunity to have an experience in this country where the great culture and way of life is different of ours.
My intership subject “Musical transcription with LMS algorithm” is one of the specialities of my tutor Mister Kudoh. This project’s goal is to make a program which, when it analyses music, it can tell us whose tones were played. The part of analysis is based on Fourier’s equation “LMS algorithm”. This algorithm analyses all the amplitudes which this sound is composed: thanks to the played sound’s amplitude and frequency, it estimates the real played sound value and its errors due to interferences.

The programming of this project is normally in C language. However, my tutor let me the choice of the programming language. So, I decided to make it in JAVA language in order to use the skills that I learnt during my education. This choice was really a risk because, the programming language that is teaching in this Kosen in the C language. Therefore, I knew that I will work in autonomy. Albeit, I faced many problems during this project, I completed it in time. Thereby, this project was a success.

This intership in Japan, allow me to learn how to work in autonomy close to a group, to develop my sense of analysing encountered problems and find the best solution. At last but not least, this intership was benefited for me: I learnt many things about the organization and observation that we have to have, when we are doing a project like this, respecting the dateline and rules.

Résumé

Durant mon quatrième semestre de mon DUT Informatique de gestion à l'Université de Lille1, j'ai effectué un stage de fin d'études d'une durée de trois mois. Le but de ce stage est d'évaluer les compétences informatiques que j'ai acquises durant ma formation.

J'ai décidée de faire ce stage au Japon car c'était une opportunité d'avoir une expérience dans ce pays dont la riche culture et la façon de vivre est différente de la nôtre.
Mon projet de stage « Musical transcription with LMS algorithm » est une spécialité de mon tuteur Monsieur Kudoh. Le but de ce projet et d'effectuer un programme qui, lorsqu'il analyse une musique, ressort les notes jouées correspondes. La partie analyse du projet est basé sur les équations de Fourrier : « LMS algorithm ». Cet algorithme analyse l'amplitude d'un son, estime la valeur du son réellement joué et son erreur dû aux interférences. Ceci se fait grâce à la l'amplitude du son joué et de sa fréquence.

La programmation du projet s'effectue normalement en langage C. Cependant, mon tuteur me laissa le choix du langage de programmation. J'ai donc décidé de l'effectuer en langage JAVA afin d'utiliser les compétences que j'ai acquises durant ma scolarité.
Ce choix était un très grand risque car le langage de programmation enseigné dans le Kosen est le langage C. De ce fait, je savais que je travaillerai dans une grande autonomie. Bien que j'aie rencontré des problèmes durant ce projet, je l'ai mené à termes et dans les délais. De ce fait, ce projet est une réussite.

Ce stage au Japon m'a donc permis d'apprendre comment travailler en autonomie auprès d'un groupe, de développer mon sens de l'analyse des problèmes rencontrés et trouver les solutions les plus appropriées. Enfin, ce stage a été bénéfique pour mon apprentissage de l'organisation et d'observation afin d'effectuer un tel projet en respectant les délais et les règles demandés.

まとめ

私はリール大学の 2 年生の最後の学期に、3 ヶ月間、このインターンシップをした。目標は、インターンシップを通して、大学で習った情報スキルを評価することであった。

私は日本でインターンシップをすることに決めた理由は、フランスの文化と大きく異なるこの国の習慣や、人々の暮らし方を経験する機会だったからである。

私のインターンシップのプロジェクト名は「Musical Transcription with LMS Algorithm」で、研究室の工藤先生の 1 つの専門である。プロジェクトの目的は音楽をプログラムで分析し、どんな音調かがわかるためである。LMS アルゴリズム分析の一部はフーリエ方程式に基づいている。プログラムは音楽の振幅と周波数を利用し、それのレアルサウンドとエラーを推定する。

プロジェクトは通常 C 言語を使うはずだが、先生は私に好きな言語を使っていいといった。それで、私は大学で学んだスキルを使用するため、JAVA 言語でそれを作ることにした。この選択は本当に危険だった。なぜなら、高専で教えている言語は C 言語であったからだ。したがって私は自律で作業することを知っていた。プロジェクトをやっている時、多くの問題に発生したけれども、時間内に完了することができた。それでプロジェクトは成功できた。

日本でのインターンシップを通して、グループで考え、自治で働くことと、起きた問題を分析し、解決する能力を身に着けることができた。最後に、このインターンシップからたくさんことを習ったといえる。たとえば、計画を立てることと、必要な情報を調べることと、また、このようなプロジェクトをする時、日付や規則を守ることなどである。

Table of contents

Introduction

During my last semester of my "DUT Informatique de Gestion"[1] in the "Université Lille1", I have to make intership to complete my education, to carry out a project in order to improve my computer skill and finally to graduate.
I have chosen to make this intership in Japan, because it was a good opportunity to discover a new culture, a new way of life and a new education system. That was the reason why I learnt Japanese during my two year in the IUTA[2] in Lille to improve and make easier the relationship.
Then, I had the chance to make my intership in the north of Japan, in Hachinohe kosen.

For my internship in Hachinohe Kosen in Japan, some subjects of projects were proposed to me. I have chosen 'the musical transcription' project because I thought that it was very interesting to associate music and computing. Instead of programming, 'the musical transcription with LMS algorithm' is based on Fourier's mathematical equations[3] that I had to use along this project.

How can I make this software that can recognise the notes of an input music and transcript it?

This project is normally programmed in the language C[4] but my project tutor, Mister Kudoh, let me the choice between the C language and JAVA language[5]. I decided to make it in JAVA even if I know that I will work in autonomy because the most used programming language in this Kosen is the language C. That is the reason why, this project is divided in several parts to make it easier to do. These parts are themselves divided in smaller steps that I will present you.

First of all, I will make a description of the environment of my intership, and then I will describe you how I make my program and finally the skills that I earn during this intership.

[1] DUT Informatique de gestion (Glossary page 57)
[2] IUT (Glossary page 57)
[3] Fourier's equation : look at page14 "Programming of the project"
[4] C language : programming language (see in Glossary page 57)
[5] JAVA language : programming language (see in Glossary page 57)

Presentation of the intership

I. Hachinohe Kosen

First, I wanted to come in Japan because I love Japanese culture, way of life and I wanted to see the difference between Japanese's education and French's one. Thus, I have chosen Japan for my school intership.

1. Hachinohe

Hachinohe is the largest city located in the flatlands on the east coast of Aomori prefecture in the Tôhoku region of Japan.

This city is facing the Pacific Ocean with a port which having one of the largest volumes of landed fish in Japan. However, since its designation as a new industrial city in 1964, Hachinohe has developed a large coastal industrial belt with a diverse range of chemical, steel, cement and fertilizer products.

The symbol of Hachinohe is the Yawata-uma, a wooden horse with gold saddle markings and a decorative plume attached to its head. The art of Yawata-uma figurines is a regional art form and popular memory.

Thanks to Japanese teachers and students, I had the chance to visited many famous places of Hachinohe and Aomori prefecture.
Like in Aomori prefecture, I had visited Hirosaki Park which is the best place to do the hanami, and then I went to Towada Lake, after 9km working along the Oirase-kekyu, the river engendered by Towada Lake. But also, I went on the top of the famous mountain to go to a typical onsen, the Japanese public hot spring bath and many other places.

2. Kosen

A Kosen[6] is a college that gather a High school and the begging of the university. If we try to translate a kosen in French, a kosen will be a "Lycée" and three years in University so, a "Licence".

Student inter in the kosen at 15-yers-old in the first grade. Then, during five years they learn many things about the speciality that they had chosen when they arrived in the kosen. After that, when they arrive in the fifth grade class, they have the choice to work or go in the advanced class during two more years. There, they will improve their skills thanks to a project that they have to do with a lot of experiments. These two more year, allow these students to continue their education at University or to work.

If we translate the kosen learning system in French[7], the first grade is our first year in the "Lycée" thus, the class that name "Seconde". Then, the fifth grade class is corresponding in our first year in University. Finally, the second year in advance class is our third year in University which the equivalent is the graduation of the "Licence".

[6] Annex 1 page30 : Japanese kosen course system
[7] Annexes 2 to 4 pages 31-32 : Translation of Japanese kosen into French school system

Each kosen have its specialities. Hachinohe kosen specialities are “Electricity and Computing sciences”, “Mechanics”, “Chemistry” and finally “Civil Engineering”. To recognize the speciality of a student, they have on their uniform a pin with the speciality letter: “E”, “M”, “C” or “Z” for the “Civil Engineering” class.

To be more integrated in the kosen, Mister Hirakawa attribute us a fifth grade class according to our intership project. Another student from the IUTA of Lille1 and I were in the E5 class.
With this class, we participated at the annual sport festival where each speciality is in competition to have the trophy of the kosen during the school year[8].

3. Laboratory

During the fifth grade and advanced course, the students have to carry out a researched project to graduate. This project depends on their speciality and on what they want to study. So, they choose the theme of their researches which correspond on of the teachers specialities. According to their choice, they are in the teacher’s speciality laboratory.

I am in the laboratory[9] with five students of E5 class and one in the second advanced course who came in France last September. Students call this laboratory Kenken because our teacher first kanji of his family name can be read “ken” and the first kanji for a laboratory in Japanese is “ken”.
This laboratory has all the instruments that we need to do our researching: oscillator, generator, and other instruments. There is also a special box without sound inside. This box is used for the “echo canceller” project.
For my intership, I choose to program in my own laptop with the software eclipse[10] that I have already used during my “DUT Informatique”.

Thus, I have a good working environment to do make intership in Hachinohe kosen. Now, I will present you in intership project.

II. Intership Project

My Intership project “Musical transcription with LMS algorithm” is one of my tutor’s specialities. This project is based on Fourier’s equations that determinate the amplitudes of a sound. I have to use these equations to analyse music and make the corresponding partition.

1. Simple version

a. Fourier’s first equation

The generation of the input signal is based on Fourier’s equation:

$$x(t) = a * cos(\omega * t) + b * sin(\omega * t) + \emptyset(t)$$

With a and b two numbers that I have chosen. These numbers represented the amplitude of the signal. We also have $\emptyset(t)$ the noise which will be represented, at first, by a random number between$[-0.5; 0.5]$. Then, there is ω the variant amplitude found thanks to this equation with f and fs the frequencies that I have already defined:

$$\omega = 2\pi * \frac{f}{fs}$$

[8] See the annex 5 page 33 : E5 class during the annual sports festival
[9] See the annex 6 page 33 : Kenken Laboratory
[10] Eclipse software (Glossary page 57)

b. Simple Least Mean Square

I use the method of Least Mean Square to estimate the signal with this equation where $a'(0)$ and $b'(0)$ are initialised at 0 at $t = 0$:

$$\boxed{x'(t) = a'(t) * cos(\omega * t) + b'(t) * sin(\omega * t)}$$

c. The error

After that, I have to calculate the error between $x(t)$ and $x'(t)$:

$$\boxed{e(t) = x(t) - x'(t)}$$

d. The updates

Finally, I update a' and b' using the error $e(t)$ and μ a small positive constant that we defined following these equations:

$$\boxed{a' = a' + \mu * e(t) * cos(\omega * t)}$$
$$\boxed{b' = b' + \mu * e(t) * sin(\omega * t)}$$

Normally, a' will converge in a and b' in b.

With these equations correspond of my first part of programming. With them, I can estimate the signal of the playing sound. However, in the second part, I have to use improved equations based on this one to have analyse that are corresponding like a real sound.

2. Improved version

In the improved version, the update change, that is the reason why, the analysis are more preferment.

$$\boxed{a' = (1 + \gamma) * a' - (\gamma * a'') + \mu * e(t) * cos(\omega * t)}$$
$$\boxed{b' = (1 + \gamma) * b' - (\gamma * b'') + \mu * e(t) * sin(\omega * t)}$$

We can see that a'' and b'' appear. These two values are corresponding in the second update. So, we have to update them too.

$$\boxed{a'' = a'} \quad \boxed{a' = a}$$
$$\boxed{b'' = b'} \quad \boxed{b' = b}$$

This improved version is my second part of my project. That is the reason that I have to use these two versions of Fourier's equation to estimate the values of the sound signal and generate the partition.
Now I will explain you how I make my program is made of and how I make it.

Programming of the project

As I told before, I decided to make this project in language JAVA, that is the reason why, I have to adapt my programme to make it simple and clearly for a better comprehension of an outside person. First of all, I only use predetermined value in the simple and improved version because I only want to know the values of perfect sound. And then, I use reals values of a simple music that I made.

I. Simple LMS

This part of my project is to define the problem, make a program which can resolve it with simple values. This program is making by a loop which calculates Simple LMS equation to generate a graph. This graph represents the amplitude of the signal a and b by t the time. This graph is composed of the input signal, the estimate signal, the error and finally the updates. To make it, we need an excel file so I have to export my data in the format CSV which can be used in excel software and then make the graph that we need.

1. Encountered problems

Before making my first step program, I used to search a solution because there are some differences between French norms and Japanese ones and between the language C and JAVA. That is the reason why I created a type of model source code[11] which I divided in two part of programming.

a. Exportation into a CSV file

It is difficult to export an array in java to a CSV file because there is not a terminal function as a in the C language. That is the reason why, I had to add a new library into my sources to export my array into a CSV file thanks to this library methods.

b. French norms and Japanese norms

In French norms, the symbol that separates the integer part and the decimal part of a number is a comma whereas in Japan, as in American, they use a dot as the symbol. Thus, I have to create a function which changes the dot of these values to a comma.

```
/**
 * change the dot which seperate the interger part and the decimal part of a
 * number that you give to a comma
 */
public String dotToComma(String value) {
    changement = "";
    for (int i = 0; i < value.length(); i++) {
        if (value.charAt(i) != '.') {
            changement += "" + value.charAt(i);
        } else {
            changement += "" + ',';
        }
    }
    return changement;
}
```

[11] Look at the source code “Model of source code for the exportation in a CSV file in Java” annex 25 page 41

2. Generate fundamentals values

Thanks to the model that I have made before, to make my first step program, I only have to apply a simple sinusoidal equation[12] and make the graph[13] with the values.
This is the sinusoidal equation that I used in my program with $a = 1$:

$$\boxed{x(t) = a * sin(\omega * t)}$$

This graph represents the fundamentals values of a sound. These values will be used after to determinate whose sound is played.

3. Simple LMS without noise

For this program, I only use Fournier's equation without the noise at first. The chosen values for the frequencies are $f = 100$ and $fs = 1000$. Then, for the amplitude I choose to make different values for a and b to see the convergence of a' and b' . That is the reason why $a = 3$ and $b = 2$.

```
// calcul
x = a * Math.cos(omega * i) + b * Math.sin(omega * i);
xh = ha * Math.cos(omega * i) + hb * Math.sin(omega * i);
e = x - xh;
ha = ha + mu * e * Math.cos(omega * i);
hb = hb + mu * e * Math.sin(omega * i);
```

As we can see in the program[14], these equations are used in a loop. This loop has the length of the number of iteration. Then, because, I put in my array the value just after calculate, I do not need a lot of variables.

The result in the corresponding graph[15] is really incredible because we can see that if the estimate signal looks like the input signal, the error is smaller. Besides, a' and b' are really converging into a and b.

4. Simple LMS with a random noise

Thanks to the latest program, I only have to add a random noise between $[-0.5; 0.5]$ to complete the whole program of this part of my project. These random values refer to the real error that we have in the real music sound, because a perfect sound does not exist.

```
// calcul
fai = Math.random() - 0.5;
x = a * Math.cos(omega * i) + b * Math.sin(omega * i) + fai;
xh = ha * Math.cos(omega * i) + hb * Math.sin(omega * i);
e = x - xh;
ha = ha + mu * e * Math.cos(omega * i);
hb = hb + mu * e * Math.sin(omega * i);
```

[12] Look at the source code "Source code for the generation of the fundamental values" annex 26 page 42
[13] Annex 7 page 34: Fundamental values
[14] Look at the source code "Simple LMS without random noise" annex 27 page 43
[15] Annex 8 page 34: Simple LMS without random noise

In Java language, the method random return value between [0;1], the problem is that we need a value between [-0.5;0.5]. So, I reduce the value of φ using a subtraction.

The goal of this program is to see 'a real signal' with errors like in real life[16]. So, if we compare the graph that we made before with the graph of this program, we can see that the principals seem to be the same: if the estimate signal is quite the same as the input signal, the error does not really existed.

5. Optimisation of μ the small positive constant

As I explained before, μ is a constant value that I had chosen by myself, the goal of this optimised program is to show us that with different μ we do not have the same result. To make it, I use an array of μ values that we will calculate in the principal loop.

In the corresponded graph of this program[17], we can see that the biggest μ is, the faster is the convergence of a' and b' to a and b.
With this optimisation, I decided that in my future programs, I will choose $\mu = 0.01$.

```
// calcul
fai = Math.random() - 0.5;
x = a * Math.cos(omega * i) + b * Math.sin(omega * i) + fai;

for (int j = 0; j < xh.length; j++) {
    xh[j] = ha[j] * Math.cos(omega * i) + hb[j] * Math.sin(omega * i);
    e[j] = x - xh[j];
    ha[j] += mu[j] * e[j] * Math.cos(omega * i);
    hb[j] += mu[j] * e[j] * Math.sin(omega * i);
}
```

This simple version is useful to represent the perfect value of a sound. But our goal is to analyse real value and deduce what is the sound that is playing at this instant.
For that, we need the new version of this program which can reach the real value and not only tend toward it.

[16] Annex 9 page 34: Simple LMS with random noise
[17] Annex 10 page 34 : Optimisation of μ with Simple LMS

II. Improved LMS

As I had explained in the first part of this report, my second part of programming is to improve the analysis with the improved LMS. The goal of this part is to reach the real value and analyse what sound is played on this time.

1. Improved LMS with predetermines values

In this program, I apply the equation of the improved LMS that I describe before, besides, I use predetermines values. We can see in the result graph[18] that the values don't converge to predetermine values. Sometimes, the estimates values and the real value are quite the same. We can conclude that, in this improved version, the performance is better because we can have the real values.

```
//calcul
fai = Math.random() - 0.5;
x = a * Math.cos(omega * i) + b * Math.sin(omega * i) + fai;
xh = ha * Math.cos(omega * i) + hb * Math.sin(omega * i);
e = x - xh;

ha = (1.0+gamma)*iha - gamma*iha2 + mu * e * Math.cos(omega * i);
hb = (1.0+gamma)*ihb - gamma*ihb2 + mu * e * Math.sin(omega * i);

iha2 = iha;
iha = ha;
ihb2 = ihb;
ihb = hb;
```

Until now, I only use one frequency. In real live, each sound have its own frequency. That is the reason why, I have to adapt my program for multi-frequencies values.

2. Improved LMS with three different frequency: do, re and mi.

My tutor gave me the table of music tones frequencies[19] which are depending on the played octave. Now, I will only use the octave5 sound because these values are the nearest of the fundamental values.

Before programming, I have to analyse the way of doing that. The problem is that if I change the frequency f, I also have to change the values of ω because ω depends on f. After analyse it, I found a solution to make this program using an array of values for f and ω that I initialise in the beginning of the program[20].

After that, I have to make a loop in the equation of x and x' because they are depending on ω. This loop must be the same as the number of f values because, the goal of this project is to export this method of analysis not only for one octave but for all the frequencies of sound.

[18] Annex 11 page 35: Improved LMS with simple values
[19] Annex 13 page 36 : Table of music frequencies
[20] Look at the source code "Improved LMS with multi-frequency" annex 28 pages 44-45

```
// calcul of multifrequencies
for (int j = 0; j < f.length; j++) {
    x += a * Math.cos(omega[j] * i) + b * Math.sin(omega[j] * i) + fai;
    xh += ha * Math.cos(omega[j] * i) + hb * Math.sin(omega[j] * i);
    e = x - xh;

    ha = (1.0 + gamma) * iha - gamma * iha2 + mu * e * Math.cos(omega[j] * i);
    hb = (1.0 + gamma) * ihb - gamma * ihb2 + mu * e * Math.sin(omega[j] * i);

    iha2 = iha;
    iha = ha;

    ihb2 = ihb;
    ihb = hb;
}
```

We can see in the result graph[21] that the performance of analysing the values is better and faster. We can observe that we reach the predetermine values even if there is some error due to the interference of the multi-frequency.

This is the biggest part of programming, after that, I have only to analyse the particularity of the different sounds, and do it for the whole octave 5. That is what I will explain you in the next part.

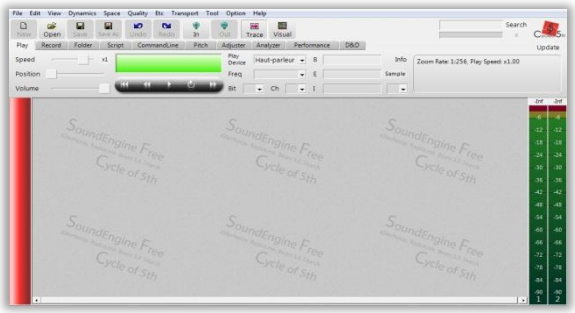
[21] Annex 12 page 35: Improved LMS and multi-frequency

III. Analysis of "Doremi" wave file

To analyse real values, I have to import a music file which has to have these characteristics: it has to be a wave file in mono sound that the length of each frame is composed of 16bits. Normally, in my basic program I chose a reading frequency of $fs = 8000$ but, I can change it for the type of sound. Now, I will only use 8Hz music sound.

1. FLStudio10 and SoundEngine

FLStudio10 is a software that I use to make the different music that I will use latter. I choose this software because we can make our own music in the octave that we want and with the instrument that we want. But, the most important is that we can directly export the music into a wave file.
Even if FLStudio10 export in wave file, this file is in stereo and we need mono file.
Thus, I use the free software SoundEngine where we can change the frequency and the number of channel of music.



2. Importations of a wave file in Java

This part of my project is the most important part. I have to find a solution to import a wave file in Java. This part was really difficult because, without this importation, I could not test my program and analyse real music.

First of all, wave file are composed in two parts that I have to analyse: the header and the data named samples. In the header I can find all the information that I need to know if this is the type of sound that my program can analyse thus, mono file in 8Hz where each data have a length of 16Bits. We can also find in the header the number of data that there are in this file. After read the header, I have to analyse the data in this wave file. Normally one data is 8bits, but in my program I analyse 16bits data. That is the reason why when I will analyse the data, I have to concatenate the data by two else my result will be false.

To make my program[22], I find in the JavaDoc[23] some Java classes which can read audio files and keep all the information that I need. With these classes, I create two methods which gave me all the data that I need in a double array.

[22] See source code "Importation of wave files in JAVA" annex 29 pages 46-47
[23] JavaDoc : Java Documentation (Glossary page 57)

```java
/**
 * return data as a byte array
 */
private static byte[] readByte(String filename) {
    byte[] data = null;
    AudioInputStream ais = null;
    try {
        // try to read from file
        File file = new File(filename);
        if (file.exists()) {
            ais = AudioSystem.getAudioInputStream(file);
            data = new byte[ais.available()];
            ais.read(data);
        }
        // try to read from URL
        else {
            URL url = StdAudio_CreationLectureSon.class.getResource(filename);
            ais = AudioSystem.getAudioInputStream(url);
            data = new byte[ais.available()];
            ais.read(data);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
        throw new RuntimeException("Could not read " + filename);
    }
    return data;
}
```

On the one hand, I create a method named “readByte” that can read and put in a byte array all the data from the filename passed in parameter.

First, I declare I File type with the name of our file that we give in parameter. Then, I initialise the AudioInputStream with the file that I create and try to read this file thanks to the method “available()” from the AudioInputStream class. This method read the data until the return value is the value of the end of the file -1 as in C language. Finally, I put the reading value in the byte array and return it.

But, if the name of the file that we give in parameter is not a name but the absolute link to go the file or an URL, I also can read this file thanks to the method “getResource()”.
I surround this action with a “try” and a “catch” which is a Java test: if there is an exception in the “try”, the program does not bloc, it pass in the “catch” and display the error.

```java
/**
 * Read audio samples from a file (in .wav or .au format) and return them as
 * a double array with values between -1.0 and +1.0.
 */
public static double[] read(String filename) {
    byte[] data = readByte(filename);
    int N = data.length;
    double[] d = new double[N / 2];
    for (int i = 0; i < N / 2; i++) {
        d[i] = ((short) (((data[2 * i + 1] & 0xFF) << 8) + (data[2 * i] & 0xFF))) / ((double) MAX_16_BIT);
    }
    return d;
}
```

On the other hand, I create a method name “read” which concatenated the values of the array from “readbyte” method, two by two. I use this concatenation to allow the LMS algorithm.
My music data’s length is 16bits but, I only have a byte array and a byte is 8bits. So, I need a concatenation to have the sample of the amplitude to use the LMS algorithm.

In case, I use this two methods to import a simple music with only the tones “do”, “re” and “mi”. Then we can see in the graph[24] that the result seems like the data we can see in software like SoundEngine.

[24] Annex 14 page 37 : Importation of “Doremi”

3. Simple LMS with real values of "doremi" wave file

Even if I only have to apply the simple version of Fourier's equation using the values of the wave file, I need to analyse the situation to combine two of my program[25]: Simple LMS and Import of a wave file.

To combine these to program, I have to change the estimate value x by the real value in the wave file that I have already extracted before.
I also have to change the frequency f because in these real values I have the frequency of the sound "do", "re" and "mi". That is the reason why I have to use the multi-frequency version of the Improved LMS program and adapt it to the Simple version.
I have to calculate the frequency of these three sounds on the same time to have the real values of the interference of the over sounds of this music.

We can see in this program graph[26] that the values are all positive because we calculate the positive square root of the estimate value.

Now, I can use this program to make the improved LMS algorithm that have the best estimation values as we have already seen.

4. Improved LMS with real values of "doremi" wave file

As I have already done the simple version of the LMS equation, to make the improved version is quite easy because I only have to change the update.

After the generation of the graph[27], if we look at the difference between the two latest programs and this one, we can see that first of all, this one also has positive value. Then, the Simple LMS graph seems like more a curve graph that an audio graph. Finally, with the Improved LMS, the getting values are higher than the simple one.

We can say one more time that the improved version is more efficient than the simple version so, for the rest of the analysis, I will only use the improved version.
The analysis for "do", "re" and "mi" sounds are a success, now I have to export this program for the entire octave 5.

[25] Look at the source code "simple LMS with Doremi music" annex 30 pages 48-49
[26] Annex 15 page 37 : simple LMS with "Doremi" music
[27] Annex 16 page 37 : Improved LMS with "Doremi" music

IV. Analysis of octave 5 and chord

The latter program was only for “do”, “re” and “mi” tones. Now my goal is to extend this last program to the entire octave 5.

1. Improved LMS with all the octave 5 in wave file

To extend my latest program into the whole octave five, I change my array frequency. Then I optimised the program[28] because I saw that it is always to the same calculations I always do but with different value and array. So, I make a method that return value of the estimate values thanks to the real data in the wave file.

```
/**
 * calcul the amplitude
 */
public double amplitudeValue(int iterationNum, double x, double[][] ha, double[][] hb, int soundNum) {
    xh = 0.0;
    for (int j = 0; j < f.length; j++) {
        xh += ha[0][j] * Math.cos(omega[j] * iterationNum) + hb[0][j] * Math.sin(omega[j] * iterationNum);
        e = x - xh;
        ha[0][j] = (1.0 + gamma) * ha[1][j] - gamma * ha[2][j] + mu * e * Math.cos(omega[j] * iterationNum);
        hb[0][j] = (1.0 + gamma) * hb[1][j] - gamma * hb[2][j] + mu * e * Math.sin(omega[j] * iterationNum);

        ha[2][j] = ha[1][j];
        ha[1][j] = ha[0][j];
        hb[2][j] = hb[1][j];
        hb[1][j] = hb[0][j];
    }
    return (Math.sqrt(ha[0][soundNum] * ha[0][soundNum] + hb[0][soundNum] * hb[0][soundNum]));
}
```

Finally, I have to display the values like a partition, thus, I add for each estimate values a determine value to make it easier to see and to understand.

```
// writting the values
for (int i = 0; i < sampleValue.length; i++) {
    String[] temp = new String[] { "" + values[0][i], "" + values[1][i], "" + values[2][i],
            "" + values[3][i], "" + values[4][i], "" + values[5][i], "" + values[6][i] };

    // change the dot to the comma and write values in the file
    for (int k = 0; k < temp.length; k++) {
        csvOutput.write((dotToComma(temp[k])) + "");
    }

    // end of the line of the file
    csvOutput.endRecord();
}
```

We can see on the represented graph[29] that when a tone is played, the highest error has the value of 0.04. We this information, we can determinate a threshold for the playing sound: if the estimate value is higher than the threshold, this sound is played.
Besides, we can see that when a played sound finish, the last values of amplitude are around 0.001 but, I take a marge of 0.005 because I won’t have the interference of the over sound.

[28] Look at the source code “Improved LMS with octave 5” annex 31 pages 50-51
[29] Annex 16 page 38 : Improved LMS with the octave5

```
// check amplitude values if each note is played are not
for (int j = 0; j < played.length; j++) {
    // check if the value is more than 0.04 when the sound begins and less than 0.005 when it finishes
    if (((values[j][i] > 0.04) && !played[j]) || (played[j] && (values[j][i] > 0.005))) {
        played[j] = true;
    } else {
        played[j] = false;
    }
}
```

Music is not only composed on simple sound. That is the reason why, the next step of my project is to have the value of chord sound.

2. Improved LMS with chord and adjacent sound error

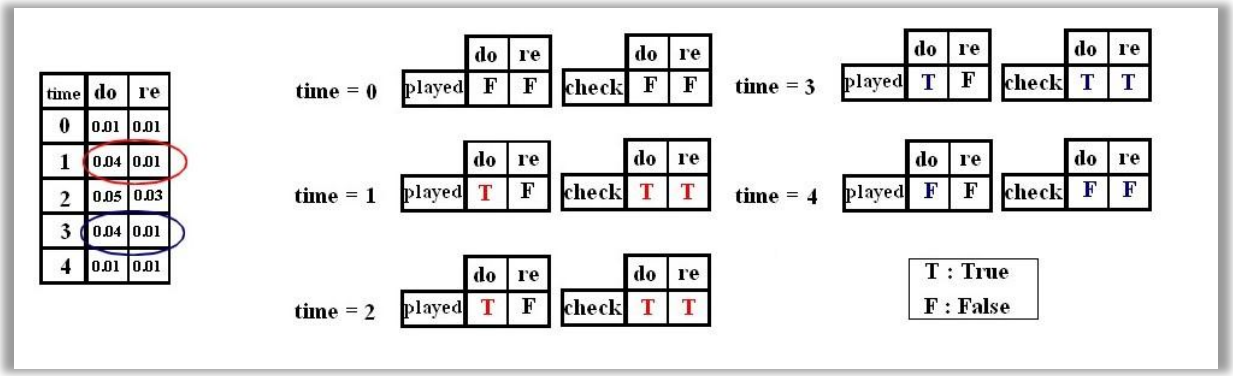
The goal of this part is to find a way to appear the chord sound and only the adjacent note error, the rest must be at 0.
This is meant that if “do” is played, I have to see the error of “re” only. Then, if “re” is played, the shown errors are “do” and “mi”. The difficulty of this part is to find a way to appear chord and error. For example, if “do” and “la” is played, I have to show in the graph the played sound and its adjacent sound thus, the graph is constitute of “do”, “re”, “sol”, “la” and “si”. The other sounds have to be at 0 even if they have a value. The value that they have is the estimate error of this sound.

To remedy it, I use two Boolean arrays: one is for the played sound named “played” and the other is check and know the adjacent tone.

```
// check amplitude values if each note is played are not
for (int j = 0; j < played.length; j++) {
    // check if the value is more than 0.04 when the sound begins and less than 0.005 when it finishes
    if (((values[j][i] > 0.04) && !played[j]) || (played[j] && (values[j][i] > 0.005))) {
        for (int k = -1; k < 2; k++) {
            try {
                checked[j + k] = true;
            } catch (ArrayIndexOutOfBoundsException e) {
            }
        }
        played[j] = true;
    } else {
        if (played[j]) {
            for (int k = -1; k < 2; k++) {
                try {
                    checked[j + k] = false;
                } catch (ArrayIndexOutOfBoundsException e) {
                }
            }
            played[j] = false;
        }
    }
}
```

We can see here the two values that I choose to make the thresholds are 0.04 for the beginning of the sound and 0.005 for the end thanks to the analysis I made with the latest program.

Now, to explain the function of these tests easily, I will make an example we only ‘do’ tones with its adjacent error “re”. I supposed for this example that the threshold for the beginning of a sound is equal are more than 0.04 and the threshold for the end of a sound is equal are less than 0.005.



To start, at time = 0, “do” is not played and wasn’t played before. So, nothing change, played[0] is still at false and it is the same thing for “re” at time = 0.
Then, at time = 1, “do” is played so checked[] turns to true for the adjacent tones and played[0] also turns to true. Besides, “do” is still play at times 2 and 3.
Finally, when “do” is not played at time = 4 bur, it was played at time = 3 so, checked[] turns to false for the adjacent tones and played[0] also turns to false.

```
// make the amplitude not played at 0
for (int j = 0; j < checked.length; j++) {
    if (!checked[j]) {
        values[j][i] = 0.0;
    }
}
```

Then I only have to set the values which are not true in the array “checked” at 0.0 and finally display it.

Now we can see in the graph[30] that with this system, the adjacent tones errors appear.
In this graph there is a mistake about the sound “si”. When “si” is played, we also see a “played do sound” with its “re” error wherase “do” is not played at this moment.

After many analyses, my tutor Mister Kudoh discovered that the spectrum of the sound “si” that I create with the software FLStudio, is the same as do’s spectrum. That is the reason why, when “si” is played, my program detect the sound “do” and “si”.

About the graph with chord[31], we can see that this method with the two Boolean arrays is a success. So, if we don’t consider this error due to the software FLStudio, this part of my project is complete.

[30] Annex 18 page 38: Improved LMS with the adjacent tones errors on the octave 5
[31] Annex 20 page 39: Improved LMS with chord on the octave 5

3. Analyses of a wave file without error

With the last program, if I won't see the adjacent errors, I have to delete the "checked" array. The Boolean methods will still be right to calculate and display the played sound.

```
// check amplitude values if each note is played are not
for (int j = 0; j < played.length; j++) {
    // check if the value is more than 0.04 when the sound begins and less than 0.01 when it finishes
    if (((values[j][i] > 0.04) && !played[j]) || (played[j] && (values[j][i] > 0.005))) {
        played[j] = true;
    } else {
        played[j] = false;
    }
}

// make the amplitude not played at 0
for (int j = 0; j < played.length; j++) {
    if (!played[j]) {
        values[j][i] = 0.0;
    }
}
```

So, the difference between this graph[32] and the two others is that we only see the played sound whereas the others with the interference of the other sound.

This program closed my intership project. Whenever I finished, I always have my problem with the apparition of "do" when "si" is played. That is the next part that I will explain you.

V. Last version

1. New estimation equation to delete "do" error from "si" tone

After analysing the problem of the apparition of "do" values when "si" is played, my tutor found a solution to remedy it.
When I estimate the value, I have to use the adjacent tones of the played sound. For example, if "do" is played, I have to estimate its values thanks to "re" frequency and the frequency of "si" in octave 4. Then, my update will be change because of the tones from the octave 5 outside.

```
/**
 * calcul the amplitude with the ajacents tones
 */
public double amplitudeValue(int iterationNum, double x, int soundNum) {
    xh = 0.0;
    for(int i = 0; i<3; i++){
        xh += ha[soundNum][i]*Math.cos(omega[soundNum+(i-1)]*iterationNum)
                + hb[soundNum][i]*Math.sin(omega[soundNum+(i-1)]*iterationNum);
        e = x - xh;
        ha[soundNum][i] += mu * e * Math.cos(omega[soundNum + (i-1)]*iterationNum);
        hb[soundNum][i] += mu * e * Math.sin(omega[soundNum + (i-1)]*iterationNum);
    }
    return (Math.sqrt(ha[soundNum][1] * ha[soundNum][1] + hb[soundNum][1] * hb[soundNum][1]));
}
```

[32] See the annex 19 page 38 : Improved LMS without the adjacent tones errors on the octave 5

I decided to reduce the number of variables into one biggest array. This is an explication of how this calculation is done with this example.

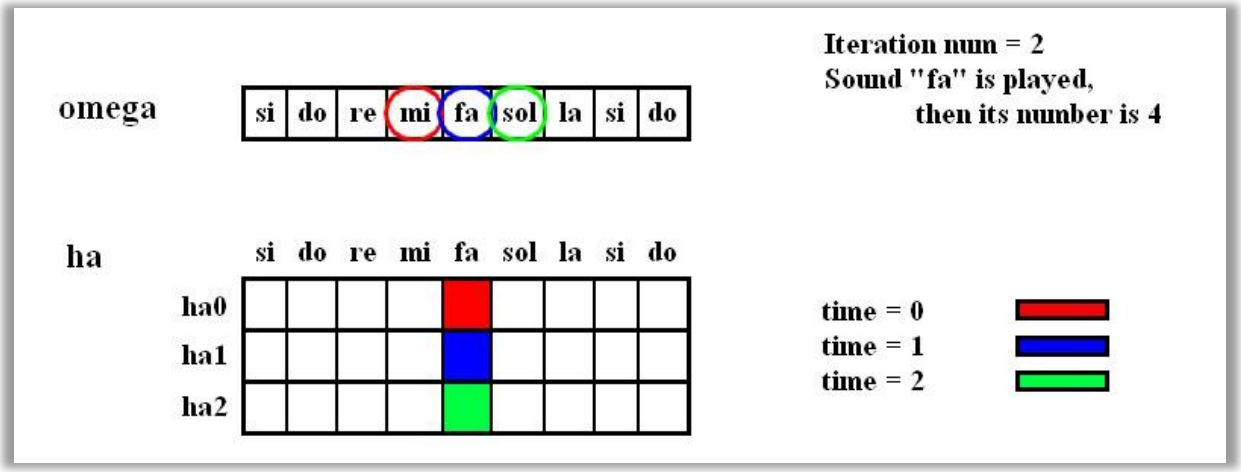
First, I reinitialise xh at 0. This initialisation is very important because xh is a global variable[33], so it may have a value before. Then if I take step by step my program with the iteration number 2 and the sound "fa" that is played, so $soundNum = 4$. At time 0, so when $i = 0$:

$$xh \mathrel{+}= ha[4][i] * \cos(omega[4 + (i - 1)] * iteration$$

It means that, for xh we will add the value for the array ha column 4 line i so, line 0, and multiply by cos of the value from omega array line $4 + (i - 1) = 4 + (0 - 1) = 4 - 1 = 3$, then $omega[mi]$.
After calculate the error e thanks to xh, the update will be:

$$ha[4][i] \mathrel{+}= mu * e * \cos(omega[4 + (i - 1)] * iteration$$
$$hb[4][i] \mathrel{+}= mu * e * \sin(omega[4 + (i - 1)] * iteration$$

As we saw before, $ha[4][i]$ is the value from the array ha column 4 line 0 and $omega[4 + (i - 1)]$ is the value $omega[mi]$.
Because we are in a loop, in time $i = 1$, $ha[4][i]$ will have the value of ha column 4 line 1 and $omega[4 + (i - 1)]$ will have as value $omega[fa]$.
Finally, when $i = 1$, $ha[4][i]$ will have the value of ha column 4 line 2 and $omega[4 + (i - 1)]$ will be $omega[re]$ as we can see in this representation.



Then, we can see in the result graph[34] without the threshold, the value of the error "do" is divided by two and seems to be disappearing. Thus, this new equation is a success.

After reviewing the graph, I decided to change threshold in the test to know whose notes were played and obtain a more beautiful graph[35].

```
// check amplitude values if each note is played are not
for (int j = 0; j < played.length; j++) {
    // check if the value is more than 0.035 when the sound begins and less than 0.001 when it finishes
    if (((values[j][i] > 0.035) && !played[j]) || (played[j] && (values[j][i] > 0.001))) {
        played[j] = true;
    } else {
        played[j] = false;
    }
}
```

[33] Global variable : a variable that can be used in all the program
[34] Annex 21 page 39: Deletion of "do" error without thresholds in octave 5
[35] Annex 22 page 39 : Deletion of "do" error in octave 5

Now, I have to try my new program[36] with chord music. If we look at the graph[37] we can see that the error of "do" is really disappeared.

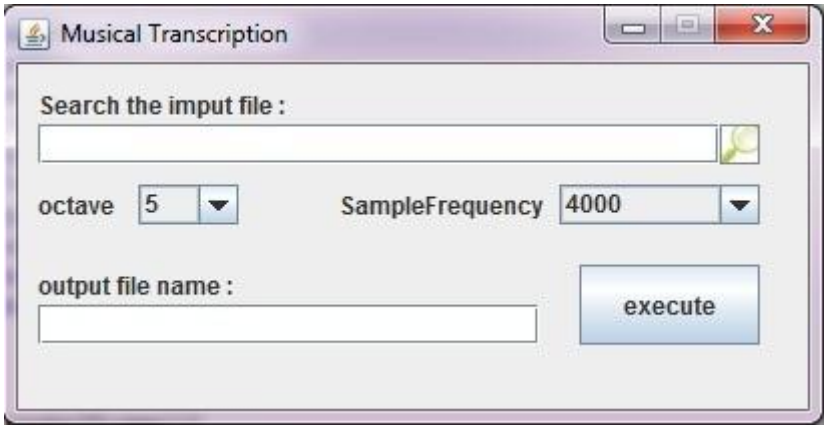
Finally, I try this new program with a real simple French music. I chose the French kids song named "Au Claire de la Lune". I change a little bit this music to make this song played on the octave 5 with chord.

After compile my program we can see on the excel graph[38] that the values upper then 0 are corresponding on the played value on the software that we use to make them.

To conclude, the method with the Boolean array is a good solution that I found to test the played sound. Moreover, the success of the last analysis to delete the "do" error makes my project of my intership complete and successful.

2. Graphic interface

Because I have a little bit time in front of me before the end of my intership in Hachinohe kosen, I made a little graphic interface to use the "Musical Transcription with LMS algorithm" program easily.



In Java language, to make this kind of graphic interface is quite simple. First of all, as I have done in my program[39], we have to initialise the needed object after their declaration.

Then, if we have actions in this graphic interface with an object, we have to add to this object a listener. The principle of a listener is to "listen" the action and change when we active the action. There is many kind of listener but in this graphic interface I only use the "ActionListener" that only wait the action form an event.

After that, we place and add the objects in the panel, and then, the panel in the frame. The question is what are a panel and a frame? If we want make an easy comparison, this frame is a picture frame and the panel is the picture that we will put in. We can only have one frame, but we can have many panels.

To finish, associating this graphic interface to my program, I create some parameter that I can change like the name of the file, the frequency, the name of the output file and the octave. For the octave, now it is not operational with the whole of octave because I have to find the good threshold for each one. That is the reason why, this is my new goal and why not, to extend this algorithm for more than one octave.

All these analysis, tests, and ameliorations, made my intership complete and more than I thought. I can use and improved all the skills that I earned during my schooling thanks to this intership.

[36] Look at the source code : Deletion of "do" error annex 32 pages 52-54
[37] Annex 23 page 40: Deletion of "do" error in chord of octave 5
[38] Annex 24 page 40: Analysis of a French children music "Au Claire de la Lune"
[39] Look at the source code " Graphic interface " annex 33 pages 55-56

New skills thanks to Japan intership

I. In professional life

This intership in Japan was really a chance that was benefited for me. The “musical transcription with the LMS algorithm” subject was a really good choice. The association between computer science, mathematics and music was really interesting. There are many difficulties that we have to understand before but the main theme is a good recipe.

During this intership, I learnt many things about working in autonomy even if we are in a group. Then, work in this kind of laboratory seems more like working in the real life than being at school. So, I improved my skill to be adapted in the working environment.

With this project, I can improve my skills to develop a functional program using imported file, mathematics equation, analysis and exported file. I also improve my analysis sense to find a way to resolve an encountered problem.

II. In social life

This intership in Hachinohe kosen gave me the opportunities to develop my general culture about Japanese way of life and school system.

First of all, I was really surprised of Japan’s beautiful landscapes. For example, when we went to do Hanami during the golden week, in one hour we were pass from a plain to mountains. I was really wonderful.

Moreover, the place of the culture in Japan is really important. I discovered many annual events like the entrance ceremony with all the new students, then, the dormitory festival with all the students who live in the kosen’s dormitory and finally, the sport festival with my attribute class. Most the students are in one of the “bukatsu”: school clubs. The ones who do not have a club are the ones who have a part-time job. I was a member of the “hikebana” club, the hikebana is the Japanese art of flower arranging, and I also was one of the members of the International Frendship Club.

In this case, the sense of the tradition and the respect of the work of the other people are written in Japanese everyday life. Maybe, that is the reason why, the link between teachers and their students are better than French one: teachers trust their student and student are respectful.

Conclusion

During this intership in Japan, I improve my skills that I earned during my two year in the IUTA of Lille. I also learnt many things about Japanese culture and way of life. That is the reason why I would like to thank again all the people who help me to make this intership real.

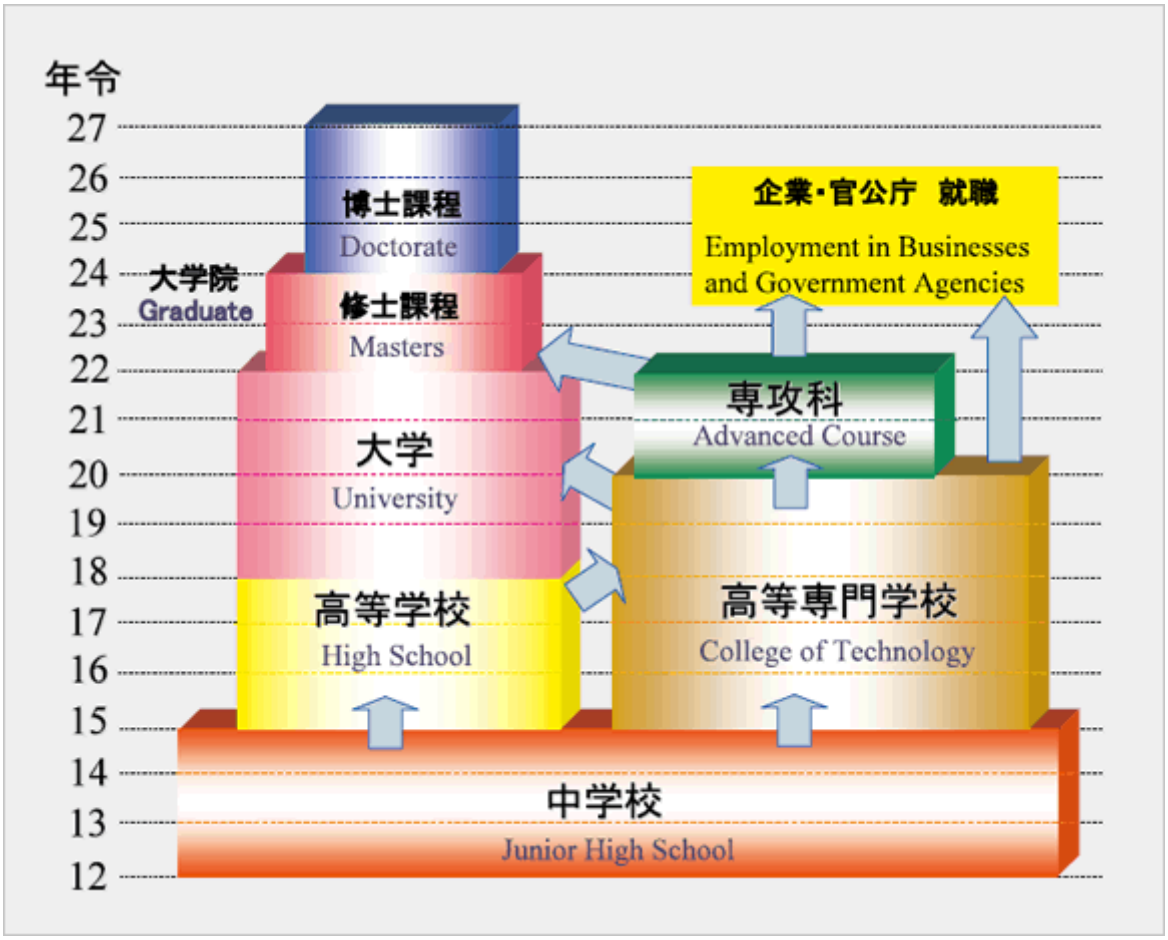
Then, about the project's programming, even if I was at first in difficulty because I chose to make this project in Java and not in C language. During this intership, I use and improved my analysis sense that I earned during my studding in the IUT trying to find the best solution to finish in time my project.

At last but not least, I can say that globally my intership was a success. Moreover, all of this intership permits me to improve my Japanese and learn more about Japanese culture. With its, I made many relationships with Japanese people. These are the reasons why, going to Japan for an intership was an opportunity that I am really thankful. I learnt and saw so many things that will never forget into this intership. Therefore, if I have another opportunity to do an intership like this one, I won't let it go and come back.

Appendices

I. School System

Annex 1: Japanese kosen course system

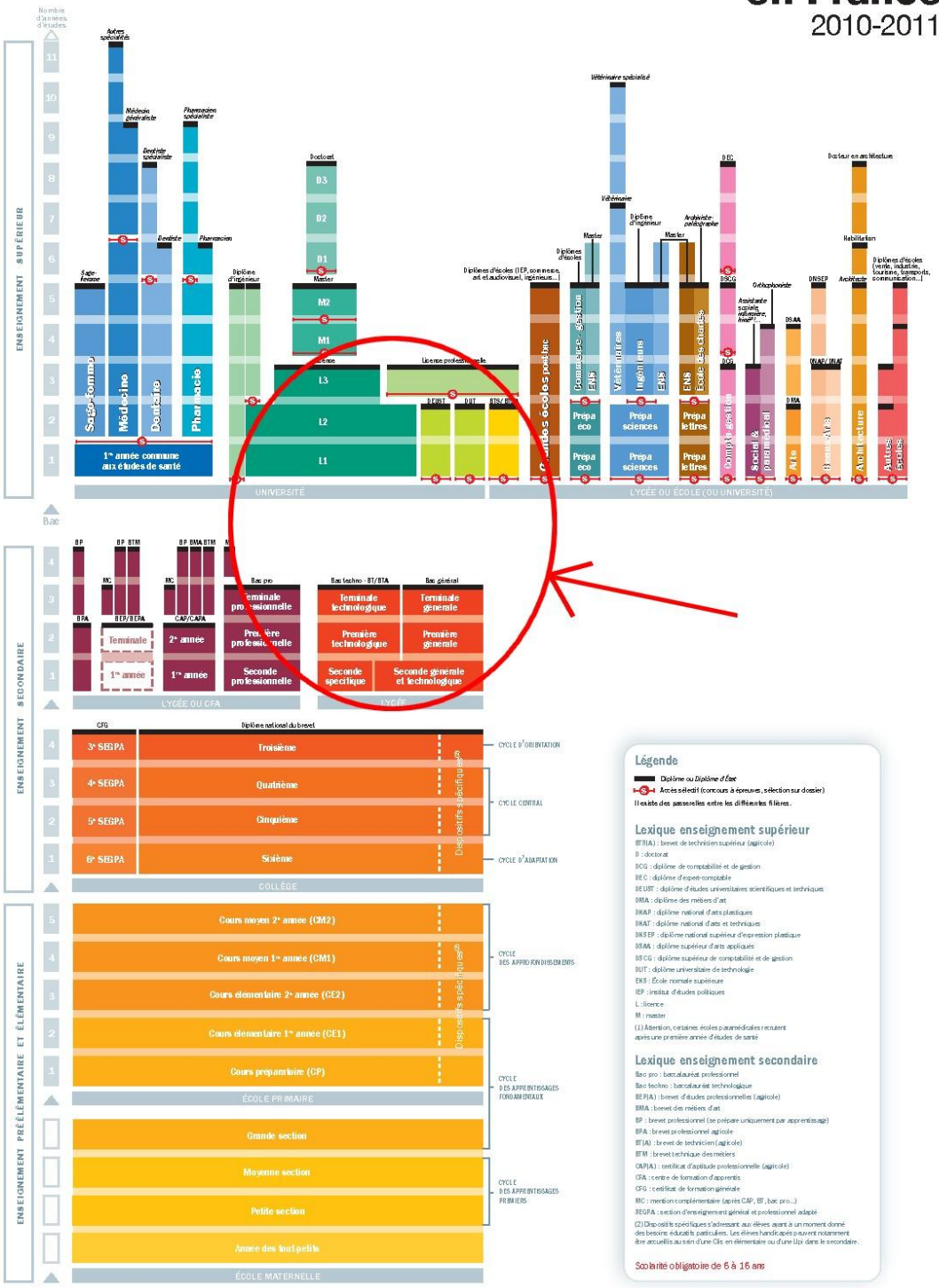


Annex 2: Translation of Japanese Kosen into French school system

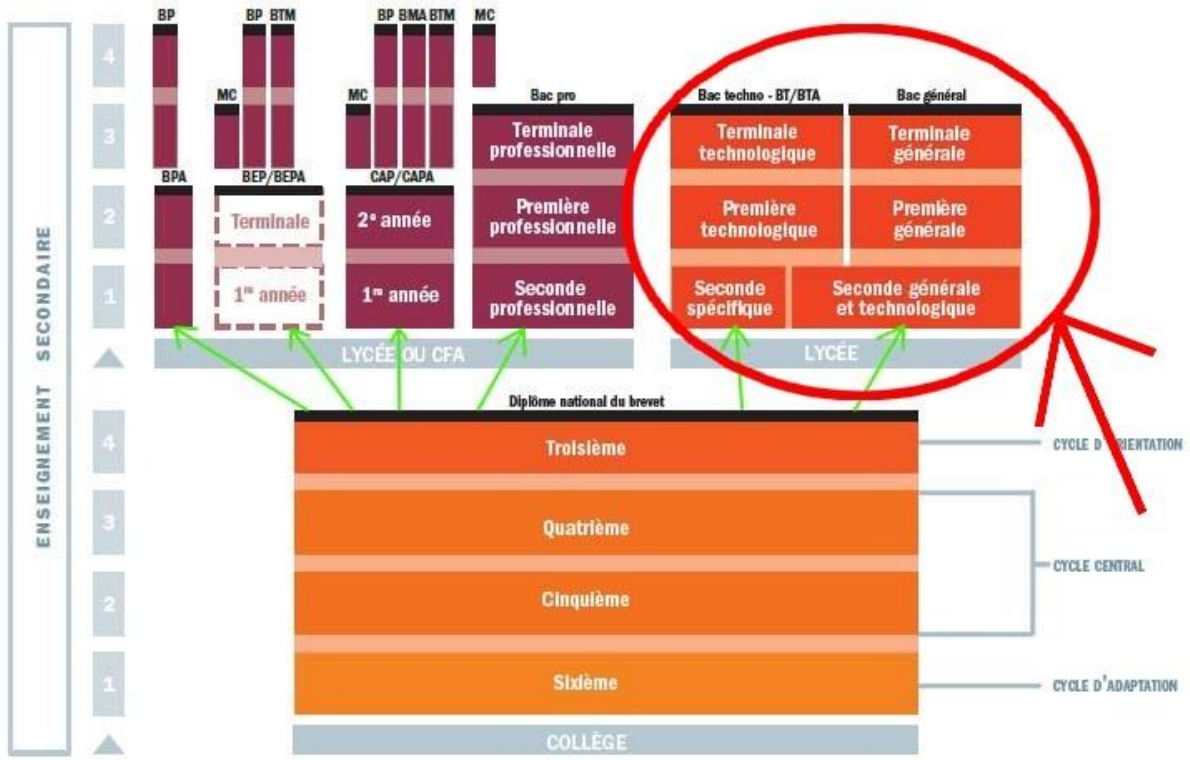
onisep

L'info nationale et régionale sur les formations et les métiers

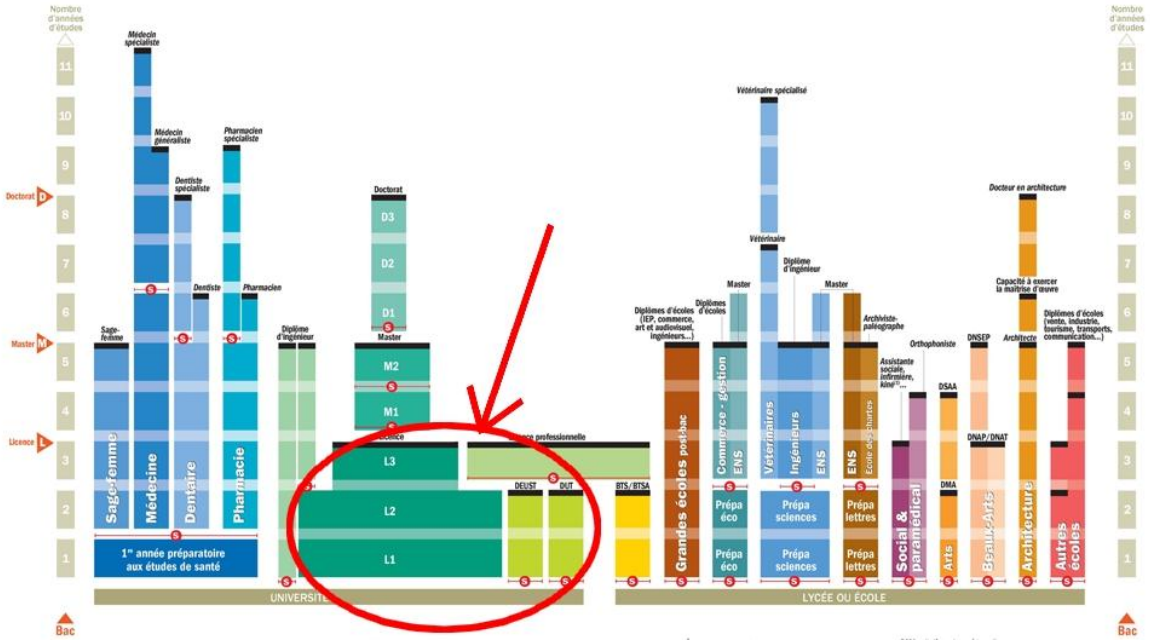
L'enseignement en France 2010-2011



Annex 3: Zoom on the translation of first year to fourth year in Japanese Kosen into French school system



Annex 4: Zoom on the translation of fifth year to second advanced course year in Japanese Kosen into French school system

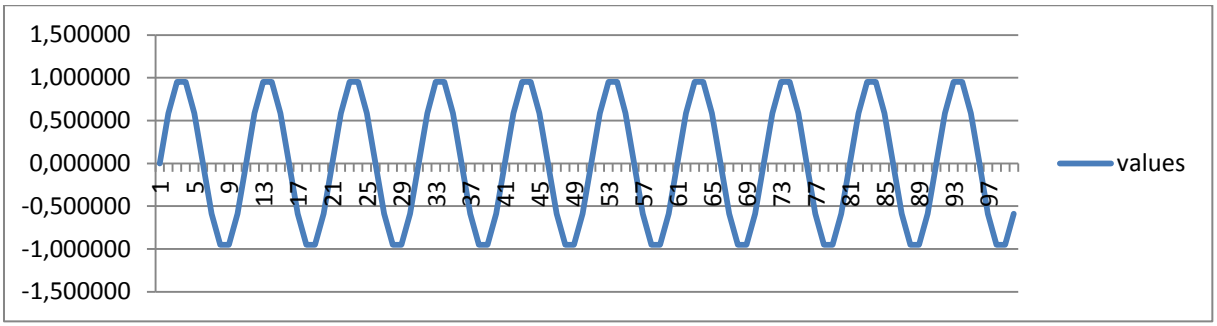


Annex 5: E5 class during the annual sports festival

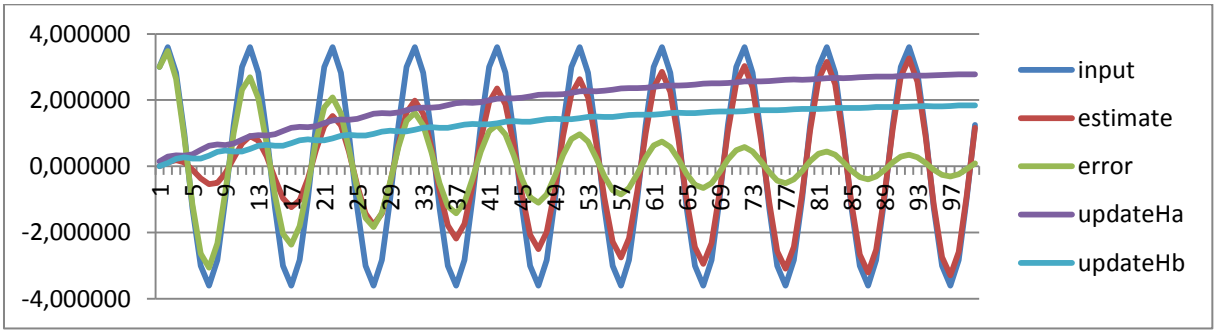
Annex 6: Kenken Laboratory

II. Graphs of intership project

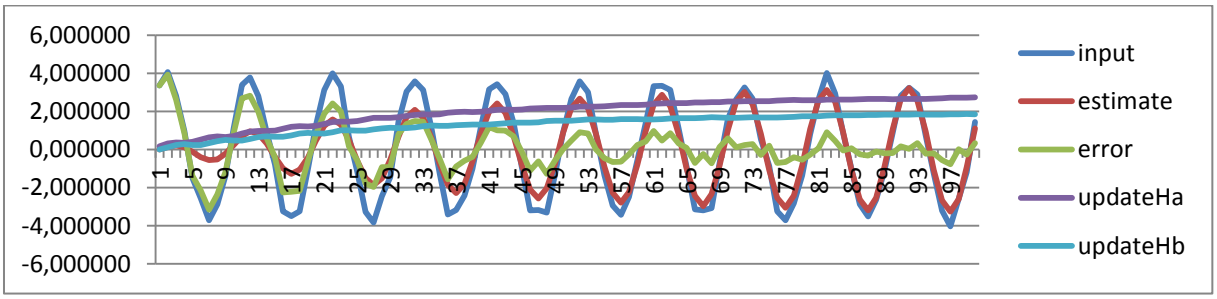
Annex 7: Fundamental values



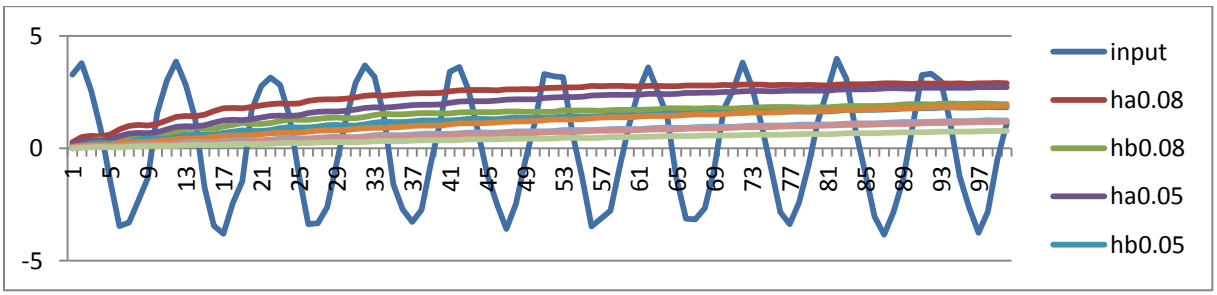
Annex 8: Simple LMS without noise



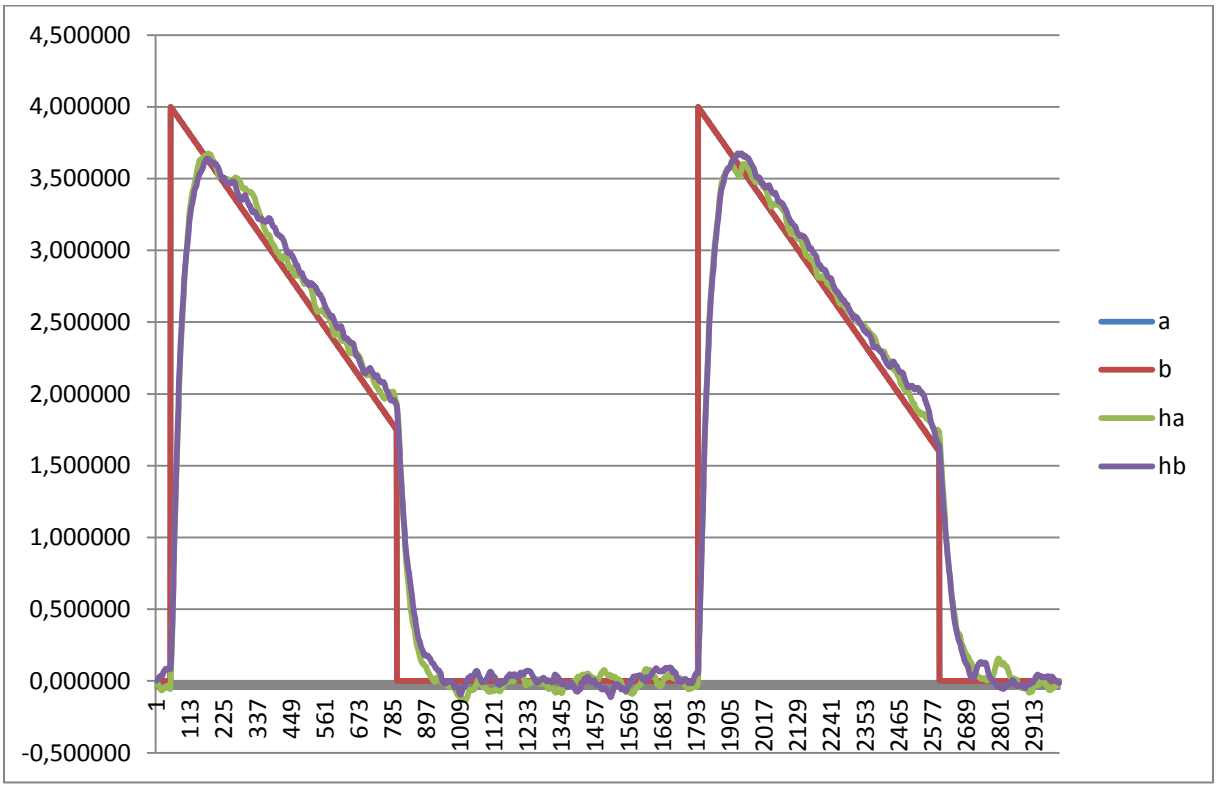
Annex 9: Simple LMS with random noise



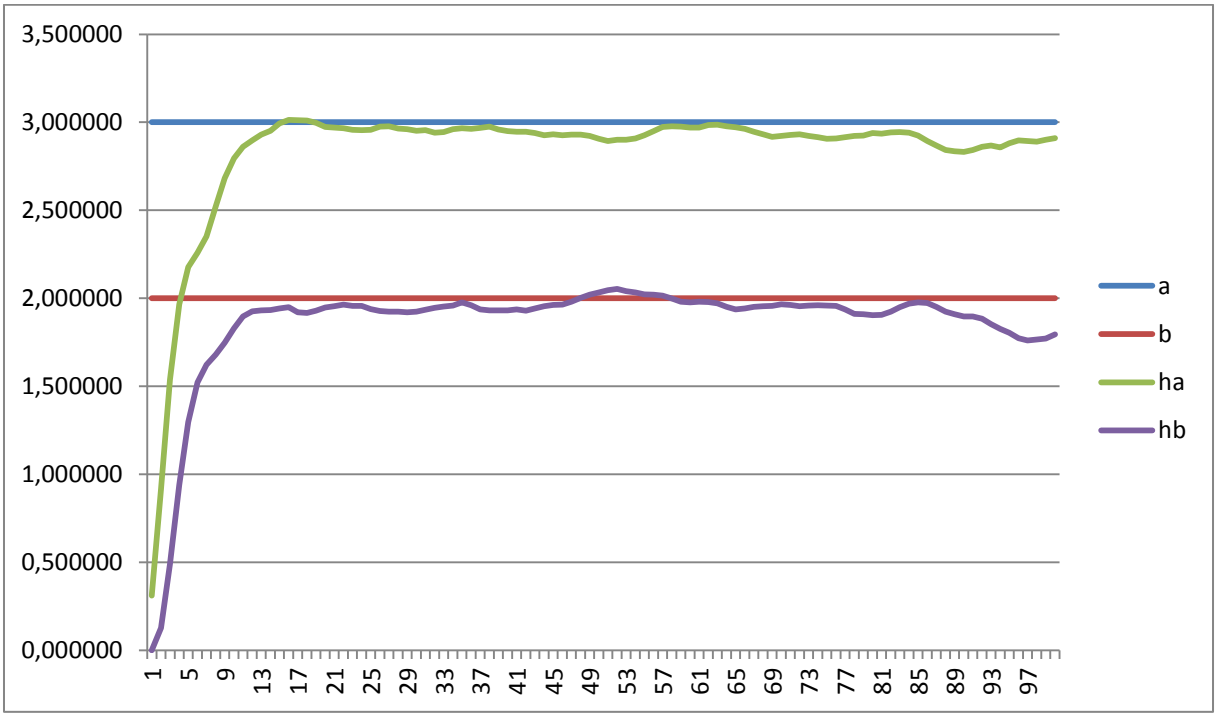
Annex 10: Optimisation of μ with Simple LMS



Annex 11: Improved LMS with simple values



Annex 12: Improved LMS and multi-frequencies

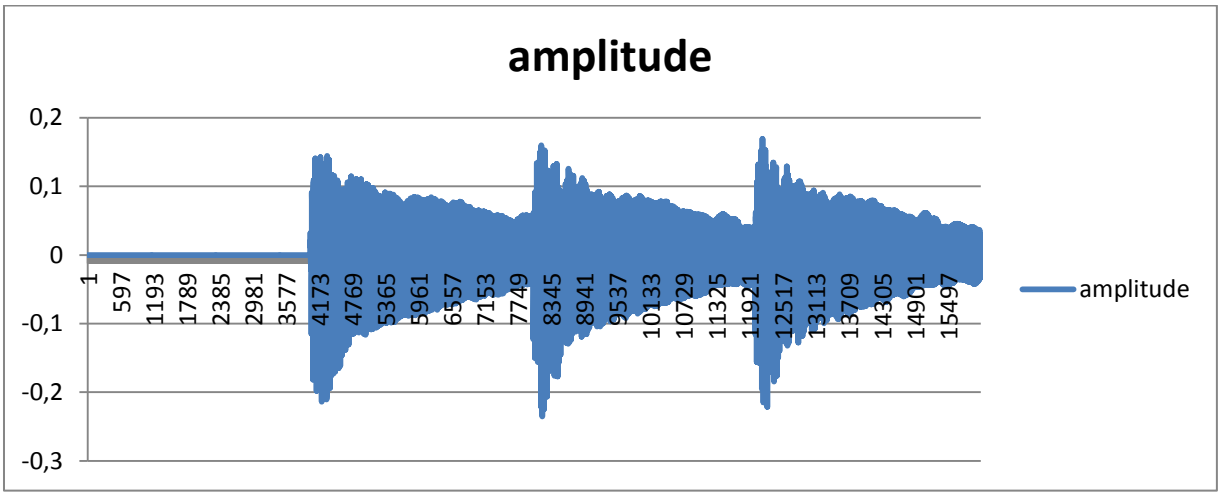


Annex 13: Table of music frequencies. Surrounded values are the ones I used for the Octave 5

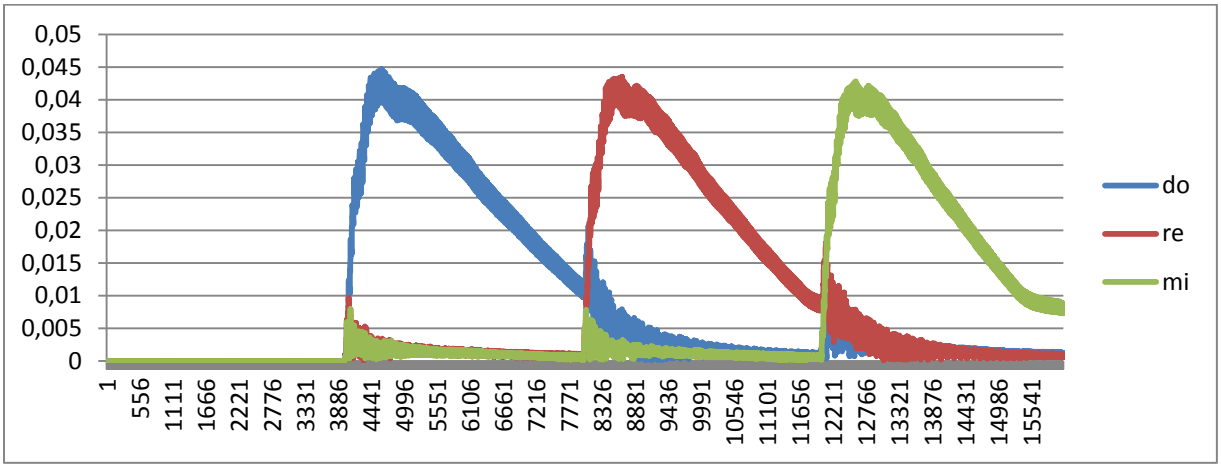
倍音

音階	オクターブ番号と倍音								
	2	3	4	5	6				
ド	65.41	130.8	196.23	261.64	327.05				
		130.8	261.64	392.46	523.28	654.1			
			261.64	523.28	784.92	1046.56	1308.2		
				523.28	1046.56	1569.84	2093.12	2616.4	
					1046.56	2093.12	3139.68	4186.24	5232.8
ド#	69.3	138.6	207.9	277.2	346.5				
		138.6	277.2	415.8	554.4	693			
			277.2	554.4	831.6	1108.8	1386		
				554.4	1108.8	1663.2	2217.6	2772	
					1108.8	2217.6	3326.4	4435.2	5544
レ	73.42	146.8	220.26	293.68	367.1				
		146.8	293.66	440.49	587.32	734.15			
			293.66	587.32	880.98	1174.64	1468.3		
				587.33	1174.66	1761.99	2349.32	2936.65	
					1174.66	2349.32	3523.98	4698.64	5873.3
レ#	77.78	155.6	233.34	311.12	388.9				
		155.6	311.12	466.68	622.24	777.8			
			311.13	622.26	933.39	1244.52	1555.65		
				622.25	1244.5	1866.75	2489	3111.25	
					1244.51	2489.02	3733.53	4978.04	6222.55
ミ	82.41	164.8	247.23	329.64	412.05				
		164.8	329.62	494.43	659.24	824.05			
			329.63	659.26	988.89	1318.52	1648.15		
				659.26	1318.52	1977.78	2637.04	3296.3	
					1318.5	2637	3955.5	5274	6592.5
ファ	87.31	174.6	261.93	349.24	436.55				
		174.6	349.22	523.83	698.44	873.05			
			349.23	698.46	1047.69	1396.92	1746.15		
				698.46	1396.92	2095.38	2793.84	3492.3	
					1396.91	2793.82	4190.73	5587.64	6984.55
ファ#	92.5	185	277.5	370	462.5				
		185	370	555	740	925			
			369.99	739.98	1109.97	1479.96	1849.95		
				739.99	1479.98	2219.97	2959.96	3699.95	
					1479.98	2959.96	4439.94	5919.92	7399.9
ソ	98	196	294	392	490				
		196	392	588	784	980			
			392	784	1176	1568	1960		
				783.99	1567.98	2351.97	3135.96	3919.95	
					1567.98	3135.96	4703.94	6271.92	7839.9
ソ#	103.8	207.7	311.49	415.32	519.15				
		207.7	415.3	622.95	830.6	1038.25			
			415.3	830.6	1245.9	1661.2	2076.5		
				830.61	1661.22	2491.83	3322.44	4153.05	
					1661.22	3322.44	4983.66	6644.88	8306.1
ラ	110	220	330	440	550				
		220	440	660	880	1100			
			440	880	1320	1760	2200		
				880	1760	2640	3520	4400	
					1760	3520	5280	7040	8800
ラ#	116.5	233.1	349.62	466.16	582.7				
		233.1	466.16	699.24	932.32	1165.4			
			466.16	932.32	1398.48	1864.64	2330.8		
				932.33	1864.66	2796.99	3729.32	4661.65	
					1864.66	3729.32	5593.98	7458.64	9323.3
シ	123.5	246.9	370.41	493.88	617.35				
		246.9	493.88	740.82	987.76	1234.7			
			493.88	987.76	1481.64	1975.52	2469.4		
				987.77	1975.54	2963.31	3951.08	4938.85	
					1975.53	3951.06	5926.59	7902.12	9877.65

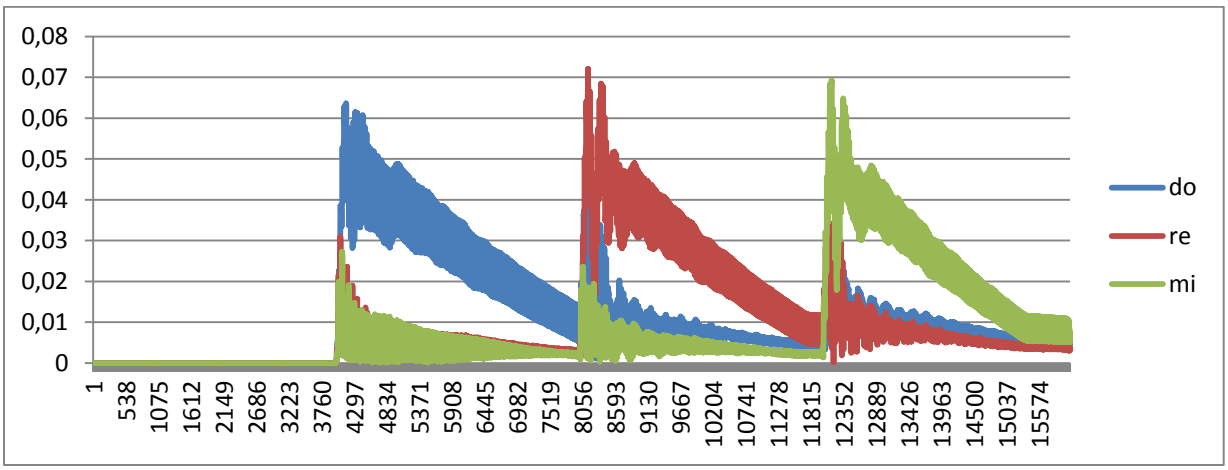
Annex 14: Importation of “Doremi”



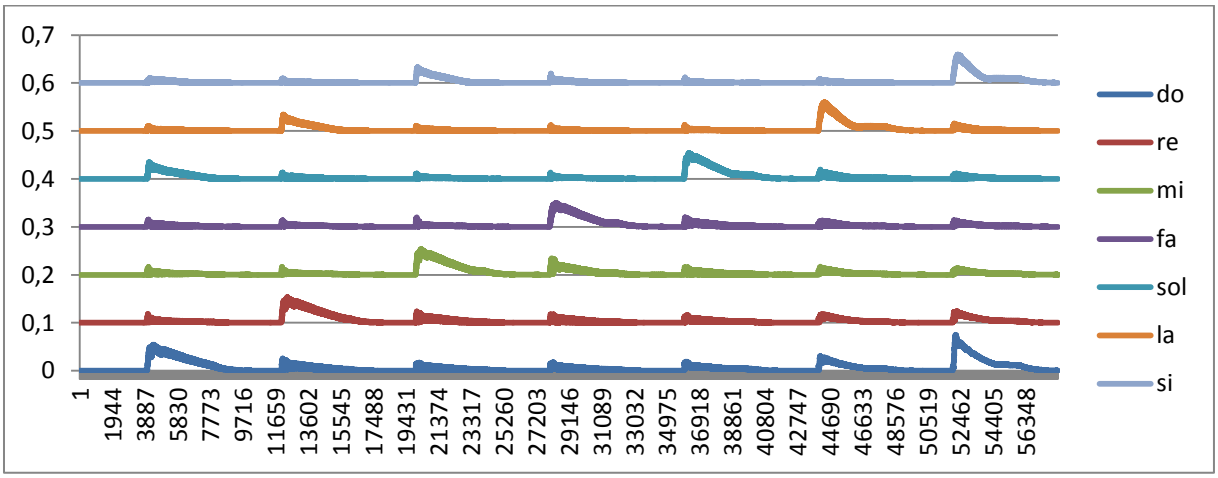
Annex 15: Simple LMS with “Doremi” music



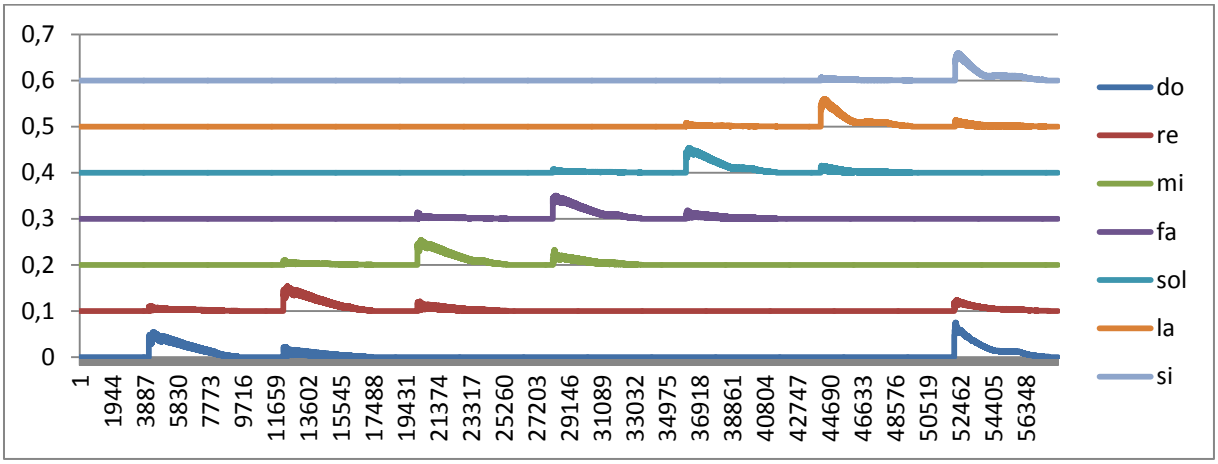
Annex 16: Improved LMS with “Doremi” music



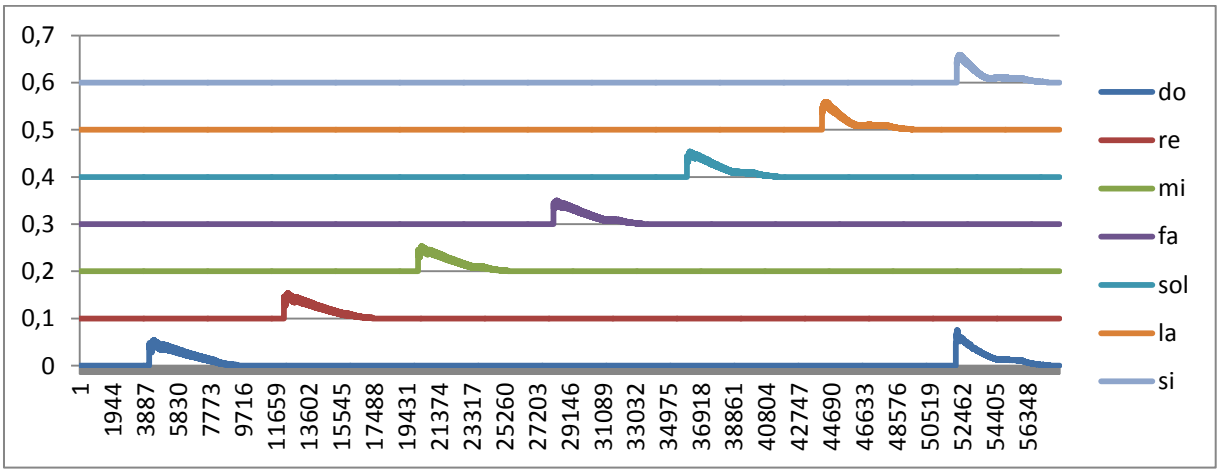
Annex 17: Improved LMS with the octave 5



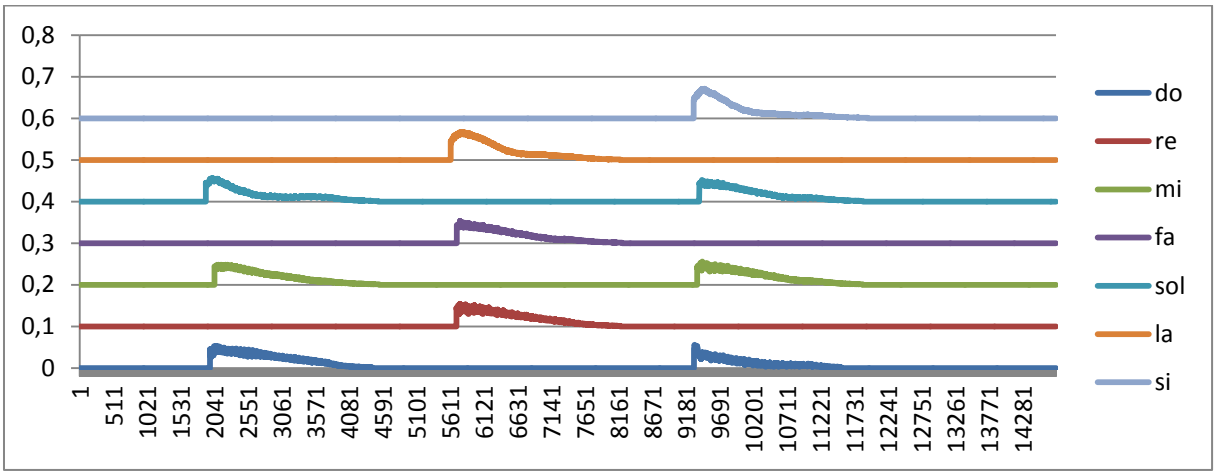
Annex 18: Improved LMS with the adjacent tones errors on the octave 5



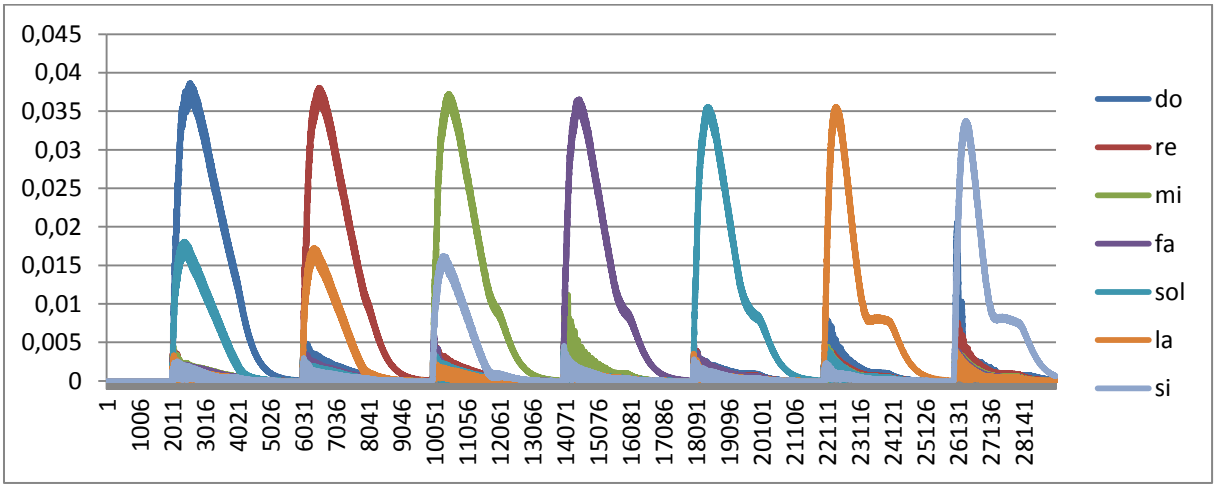
Annex 19: Improved LMS without the adjacent tones errors on the octave 5



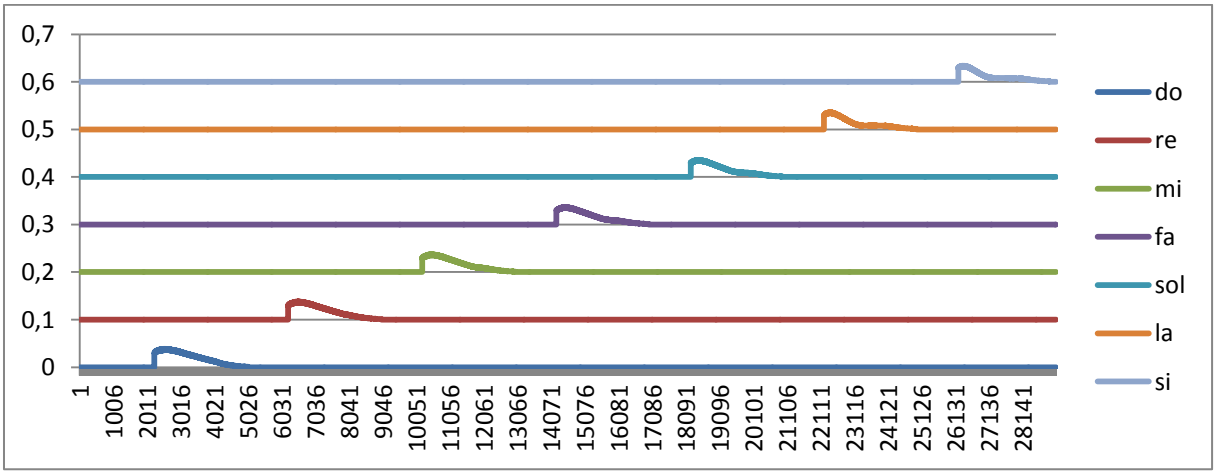
Annex 20: Improved LMS with chord on octave 5



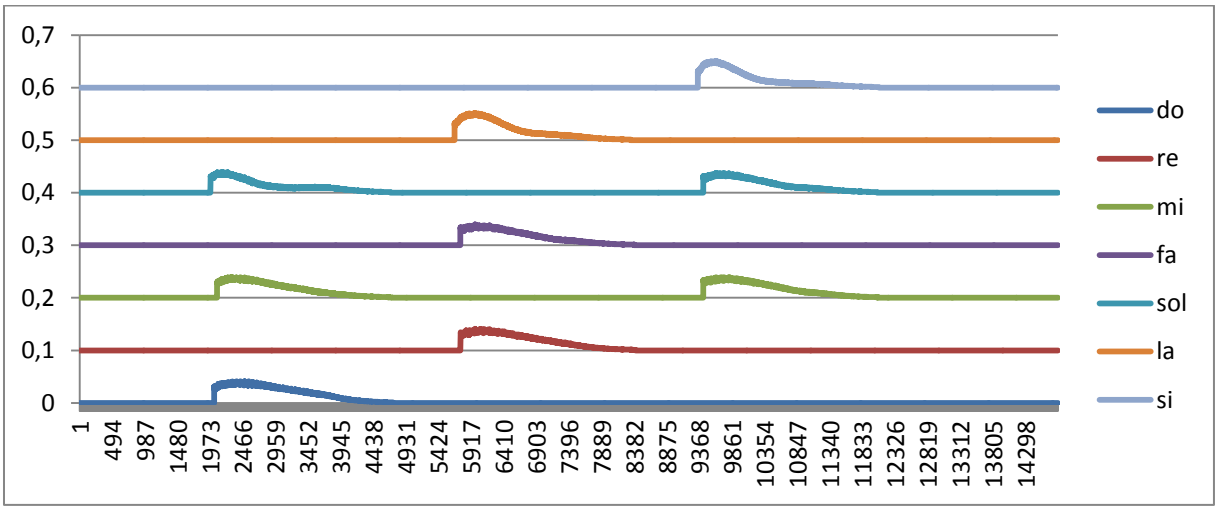
Annex 21: Deletion of "do" error without thresholds in octave 5



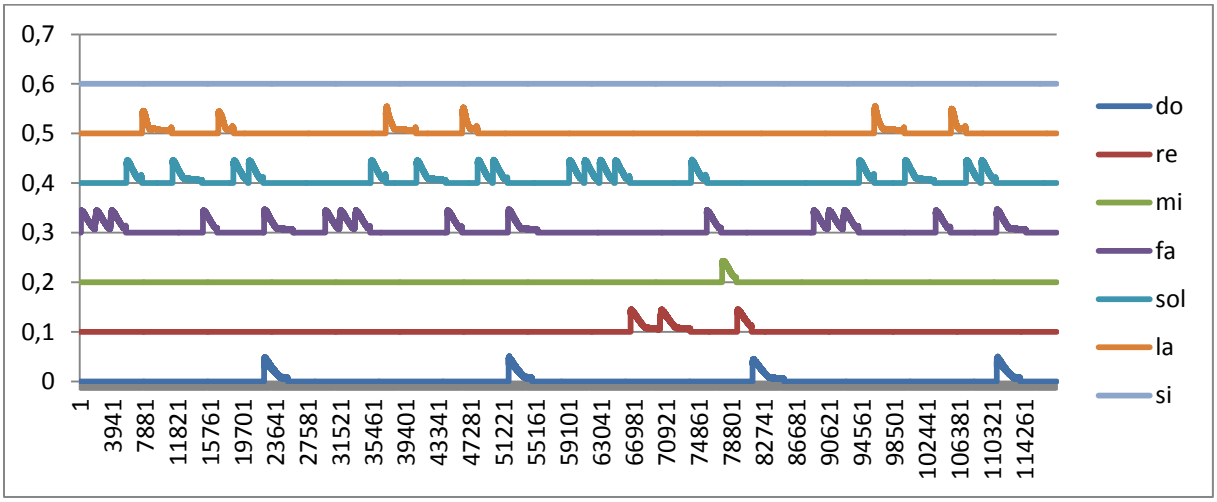
Annex 22: Deletion of "do" error in octave 5



Annex 23: Deletion of “do” error in chord of octave 5



Annex 24: Analysis of French children music “Au Claire de la Lune”



III. Source codes

Annex 25: Model of source code for the exportation in a CSV file in JAVA

```
import com.csvreader.CsvWriter;

public class ModelCsvWitter {

	// declaration of global variables
	String[][] values;
	String[] headers = { "column name" };
	String outputFile, changement;
	boolean alreadyExists;
	int time;

	// constructor of the program : constructor contains only initalisation
	public ModelCsvWitter() {
		// initialisation of the array with the number of columns (hearders) and lignes (time)
		values = new String[headers.length][time];
		outputFile = "uname.csv";				// name of the output file
		alreadyExists = new File(outputFile).exists();		// show if the file with this name already existe
		exectution();
	}

	// method for the execution of the program
	public void exectution() {
		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');
			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}
			// loop for writing the values
			for (int i = 0; i < time; i++) {

				// calcul

				// change values calculated before into string and put it in an array
				String[] temp = new String[] { "calculed value" };

				// change the dot to the comma and write values in the file
				for (int k = 0; k < temp.length; k++) {
					values[k][i] = dotToComma(temp[k]);
					csvOutput.write(values[k][i] + "");
				}
				// end of the line of the file
				csvOutput.endRecord();
			}
			// close the file
			csvOutput.close();
		} catch (IOException e) {}
	}

	// change the dot which seperate the interger part and the decimal part of a number that you give to a comma
	public String dotToComma(String value) {
		changement = "";
		for (int i = 0; i < value.length(); i++) {
			if (value.charAt(i) != '.') {
				changement += "" + value.charAt(i);
			} else {changement += "" + ',';}
		}
		return changement;
	}

	// main
	public static void main(String args[]) {
		ModelCsvWitter test = new ModelCsvWitter();
	}
}
```

Annex 26: Source code for the generation of fundamental values

```
public class ProjetPartie01_SinusoidalSignal {

	// declaration and initialisation of non-changing variables
	final double pi2 = Math.atan(1.0) * 8;
	final int fs = 1000, f = 100, amplitude = 1;

	// declaration of global variables
	String[] values;
	String outputFile, changement;
	boolean alreadyExists;



	/**
	 *  constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie01_SinusoidalSignal() {

		// initialisation of global variables
		values = new String[f];		// initialisation of the array with the number of frequency f
		outputFile = "ProjetPartie01_SinusoidalSignal.csv"; // name of the output file
		alreadyExists = new File(outputFile).exists();			// show if the file with this name already existe

		exectution();
	}



	/**
	 *  method for the execution of the program
	 */
	public void exectution() {

		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');

			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				csvOutput.write("values");
				csvOutput.endRecord();
			}

			// loop for writing the values
			for (int i = 0; i < 100; i++) {

				// change values into string
				values[i] = dotToComma("" + amplitude * Math.sin(pi2 * f * i / fs));

				// write values in the file
				csvOutput.write(values[i] + "");

				// end of the line of the file
				csvOutput.endRecord();
			}

			// close the file
			csvOutput.close();

		} catch (IOException e) {
			e.printStackTrace();
		}

	}

…
```

Annex 27: Simple LMS without random noise

```
public class ProjetPartie02_LMSPerfect {

	// declaration and initialisation of non-changing variables
	final int time = 100;
	final int fs = 1000, f = 100;
	final int a = 3, b = 2;

	final double pi = Math.atan(1.0) * 4;
	final double omega = pi * f * 2.0 / fs;
	final double mu = 0.05;
	final String[] headers = { "input", "estimate", "error", "updateHa", "updateHB" };

	// declaration of global variables
	String[][] values;
	String outputFile, changement;
	boolean alreadyExists;
	double x, xh, e, ha = 0.0, hb = 0.0;

	/**
	 *  constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie02_LMSPerfect() {
		// initialisation of the array with the number of columns (hearders) and lignes (time)
		values = new String[headers.length][time];
		outputFile = "ProjetPartie02_LMSPerfect.csv"; // name of the output file
		alreadyExists = new File(outputFile).exists(); // show if the file with this name already existe
		exectution();
	}

	/**
	 *  method for the execution of the program
	 */
	public void exectution() {
		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');
			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}
			// loop for writing the values
			for (int i = 0; i < time; i++) {
				// calcul
				x = a * Math.cos(omega * i) + b * Math.sin(omega * i);
				xh = ha * Math.cos(omega * i) + hb * Math.sin(omega * i);
				e = x - xh;
				ha = ha + mu * e * Math.cos(omega * i);
				hb = hb + mu * e * Math.sin(omega * i);
				// change values calculated before into string and put it in an array
				String[] temp = new String[] { "" + x, "" + xh, "" + e,"" + ha, "" + hb };
				// change the dot to the comma and write values in the file
				for (int k = 0; k < temp.length; k++) {
					values[k][i] = dotToComma(temp[k]);
					csvOutput.write(values[k][i] + "");
				}
				// end of the line of the file
				csvOutput.endRecord();
			}
			// close the file
			csvOutput.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
```

…

Annex 28: Improved LMS with multi-frequencies

```
public class ProjetPartie06_SimulationProgram {

	// declaration and initialisation of non-changing variables
	final int time = 100;
	final int fs = 8000;
	final double pi = Math.atan(1.0) * 4;
	final double mu = 0.01;
	final double gamma = 0.8;
	final double gradient = 0.003;
	final String[] headers = { "a", "b", "ha", "hb" };

	// declaration of global variables
	String[][] values;
	double[] f = { 523.28, 587.33, 659.26 };
	double[] omega;

	String outputFile, changement;
	boolean alreadyExists;
	double fai, x, xh, e, a = 3, b = 2;
	double ha = 0.0, hb = 0.0;
	double iha = 0.0, ihb = 0.0, iha2 = 0.0, ihb2 = 0.0;

	/**
	 *  constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie06_SimulationProgram() {

		// initialisation of the array with the number of columns (hearders) and lignes (time)
		values = new String[headers.length][time];
		outputFile = "ProjetPartie06_SimuationProgram.csv"; // name of the output file
		alreadyExists = new File(outputFile).exists();		// show if the file with this name already existe

		// initialisation of omega thanks to the fequencies f
		omega = new double[f.length];
		for (int i = 0; i < f.length; i++) {
			omega[i] = 2 * pi * f[i]/fs;
		}

		exectution();
	}

	/**
	 *  method for the execution of the program
	 */
	public void exectution() {

		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');

			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}

			// loop for writting the values
			for (int i = 0; i < time; i++) {

				fai = Math.random() - 0.5;

				x = 0.0;
				xh = 0.0;

				// calcul of multifrequencies
				for (int j = 0; j < f.length; j++) {
					x += a * Math.cos(omega[j] * i) + b * Math.sin(omega[j] * i) + fai;
					xh += ha * Math.cos(omega[j] * i) + hb * Math.sin(omega[j] * i);
					e = x - xh;
```

```
                                        ha = (1.0 + gamma) * iha - gamma * iha2 + mu * e * Math.cos(omega[j] * i);
                                        hb = (1.0 + gamma) * ihb - gamma * ihb2 + mu * e * Math.sin(omega[j] * i);

                                        iha2 = iha;
                                        iha = ha;

                                        ihb2 = ihb;
                                        ihb = hb;
                                }

                                // change values calculated before into string and put it in an array
                                String[] temp = new String[] { "" + a, "" + b, "" + iha,
                                                "" + ihb };

                                // change the dot to the comma and write values in the file
                                for (int k = 0; k < temp.length; k++) {
                                        values[k][i] = dotToComma(temp[k]);
                                        csvOutput.write(values[k][i] + "");
                                }

                                // end of the line of the file
                                csvOutput.endRecord();
                        }

                        // close the file
                        csvOutput.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }

        }

        /**
         * change the dot which seperate the interger part and the decimal part of a
         * number that you give to a comma
         */
        public String dotToComma(String value) {
                changement = "";
                for (int i = 0; i < value.length(); i++) {
                        if (value.charAt(i) != '.') {
                                changement += "" + value.charAt(i);
                        } else {
                                changement += "" + ',';
                        }
                }
                return changement;
        }


        /**
         *  main
         */
        public static void main(String args[]) {
                @SuppressWarnings("unused")
                ProjetPartie06_SimulationProgram test = new ProjetPartie06_SimulationProgram();
        }
}
```

Annex 29: Importation of wave files in JAVA

```
public class ProjetPartie07Fini_ReadExportWaveFile {


	/** declaration of variable used for writting the csv file **/
	String[] headers = { "amplitude"};
	String outputFile, changement;
	boolean alreadyExists;

	/** declaration of variables used in the reading wave file methode **/
	// bigest value of a data
	private static final double MAX_16_BIT = Short.MAX_VALUE;
	double[] sampleValue; // the array where the samples values will be stock

	String[][] values;



	/**
	 * constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie07Fini_ReadExportWaveFile(String filename) {

		// initialisation of global variables
		// initialisation of the array with the number of columns (hearders) and lignes (time)
		sampleValue = read(filename);

		values = new String[headers.length][sampleValue.length];
		outputFile = "ProjetPartie07_ReadExportWaveFile.csv";
	// name of the output file
		alreadyExists = new File(outputFile).exists();	// show if the file with this name already existe

		exectution();
	}


	/**
	 * method for the execution of the program
	 */
	public void exectution() {

		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');

			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}

			// loop for writing the values
			for (int i = 0; i < sampleValue.length; i++) {

				// change values calculated before into string and put it in an array
				String temp = "" + sampleValue[i];

				// change the dot to the comma and write values in the file

					values[0][i] = dotToComma(temp);
					csvOutput.write(values[0][i] + "");
```

```
                                        // end of the line of the file
                                        csvOutput.endRecord();
                                }

                                // close the file
                                csvOutput.close();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }

        }



        /**
         * Read audio samples from a file (in .wav or .au format) and return them as
         * a double array with values between -1.0 and +1.0.
         */
        public static double[] read(String filename) {
                byte[] data = readByte(filename);
                int N = data.length;
                double[] d = new double[N / 2];
                for (int i = 0; i < N / 2; i++) {
                        d[i] = ((short) (((data[2 * i + 1] & 0xFF) << 8) + (data[2 * i] & 0xFF)))
                                        / ((double) MAX_16_BIT);
                }
                return d;
        }



        /**
         * return data as a byte array
         */
        private static byte[] readByte(String filename) {
                byte[] data = null;
                AudioInputStream ais = null;
                try {
                        // try to read from file
                        File file = new File(filename);
                        if (file.exists()) {
                                ais = AudioSystem.getAudioInputStream(file);
                                data = new byte[ais.available()];
                                ais.read(data);
                        }
                        // try to read from URL
                        else {
                                URL url = StdAudio_CreationLectureSon.class
                                                .getResource(filename);
                                ais = AudioSystem.getAudioInputStream(url);
                                data = new byte[ais.available()];
                                ais.read(data);
                        }
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                        throw new RuntimeException("Could not read " + filename);
                }
                return data;
        }

...
```

Annex 30: Simple LMS with Doremi music

```
public class ProjetPartie08_SimpleLMSWaveFileAnalyse {

	/** declaration of variable used for writting the csv file **/
	String[] headers = { "do", "re", "mi"};
	String outputFile, changement;
	boolean alreadyExists;

	/** declaration of variables used in the reading wave file methode **/
	// bigest value of a data
	private static final double MAX_16_BIT = Short.MAX_VALUE;
	double[] sampleValue; // the array where the samples values will be stock

	/** declaration of variables used in the calcul of estimate values **/
	final int fs = 8000; // music in 8Hz
	final double pi = Math.atan(1.0) * 4;
	final double mu = 0.01;
	final double gamma = 0.6;

	String[][] values;

	// array of hat, hat1, hat2
	double[] hado, hare, hami, hbdo, hbre, hbmi;

	// frequencies of octave5
	double[] f = { 523.28, 587.33, 659.26};

	double[] omega;
	double x, xh, e;
	double ampDo, ampRe, ampMi;



	/**
	 * constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie08_SimpleLMSWaveFileAnalyse(String filename) {

		// initailisation of global variables
		sampleValue = read(filename);

		// initialisation of the array with the number of columns (hearders) and lignes (time)
		values = new String[headers.length][sampleValue.length];
		outputFile = "ProjetPartie08_SimpleLMSWaveFileAnalyse.csv";		// name of the output file
		alreadyExists = new File(outputFile).exists();		// show if the file with this name already existe


		// initialisation of omega thanks to the fequencies f
		omega = new double[f.length];
		for (int i = 0; i < f.length; i++) {
			omega[i] = 2 * pi * f[i]/fs;
		}


		// initialisation of ha and hb
		hado = new double[f.length];
		hare = new double[f.length];
		hami = new double[f.length];
		hbdo = new double[f.length];
		hbre = new double[f.length];
		hbmi = new double[f.length];

		exectution();

	}
```

```
	/**
	 * method for the execution of the program
	 */
	public void exectution() {

		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter(new FileWriter(outputFile, true), '.');

			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}

			// loop for writing the values
			for (int i = 0; i < sampleValue.length; i++) {

				x = sampleValue[i];

				//calcul for do
				xh = 0.0;
				for (int j = 0; j < f.length; j++) {
					xh += hado[j] * Math.cos(omega[j] * i) + hbdo[j] * Math.sin(omega[j] * i);
					e = x - xh;
				}
				for (int j = 0; j < f.length; j++) {
				hado[j] = hado[j] + mu * e * Math.cos(omega[j] * i);
				hbdo[j] = hbdo[j] + mu * e * Math.sin(omega[j] * i);
				}
				ampDo = Math.sqrt(hado[0]*hado[0] + hbdo[0]*hbdo[0]);


				//calcul for re
				xh = 0.0;
				for (int j = 0; j < f.length; j++) {
					xh += hare[j] * Math.cos(omega[j] * i) + hbre[j] * Math.sin(omega[j] * i);
					e = x - xh;
				}
				for (int j = 0; j < f.length; j++) {
				hare[j] = hare[j] + mu * e * Math.cos(omega[j] * i);
				hbre[j] = hbre[j] + mu * e * Math.sin(omega[j] * i);
				}
				ampRe = Math.sqrt(hare[1]*hare[1] + hbre[1]*hbre[1]);


…

				// change values calculated before into string and put it in an array
				String[] temp = new String[] { "" + ampDo, "" + ampRe, "" + ampMi};

				// change the dot to the comma and write values in the file
				for (int k = 0; k < temp.length; k++) {
					values[k][i] = dotToComma(temp[k]);
					csvOutput.write(values[k][i] + "");
				}
				// end of the line of the file
				csvOutput.endRecord();
			}
			// close the file
			csvOutput.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

}

…
```

Annex 31: Improved LMS with octave 5

```
public class ProjetPartie10_ImprovedLMSOctave5OptimisedBest {

	/** declaration of variable used for writting the csv file **/
	String[] headers = { "do", "re", "mi", "fa", "sol", "la", "si" };
	String outputFile, changement;
	boolean alreadyExists;

	/** declaration of variables used in the reading wave file methode **/
	// bigest value of a data
	private static final double MAX_16_BIT = Short.MAX_VALUE;
	double[] sampleValue; // the array where the samples values will be stock

	/** declaration of variables used in the calcul of estimate values **/
	final int fs = 8000; // music in 8Hz
	final double pi = Math.atan(1.0) * 4;
	final double mu = 0.01;
	final double gamma = 0.6;
	String[][] values;

	// array of hat, hat1, hat2
	double[][] haDo, haRe, haMi, haFa, haSol, haLa, haSi;
	double[][] hbDo, hbRe, hbMi, hbFa, hbSol, hbLa, hbSi;

	// frequencies of octave5
	double[] f = { 523.28, 587.33, 659.26, 698.46, 783.99, 880, 987.77 };

	double[] omega;
	double x, xh, e;
	double ampDo, ampRe, ampMi, ampFa, ampSol, ampLa, ampSi;

	/**
	 * constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie10_ImprovedLMSOctave5OptimisedBest(String filename) {

		// initailisation of global variables
		sampleValue = read(filename);

		// initialisation of the array with the number of columns (hearders) and lignes (sampleValue)
		values = new String[headers.length][sampleValue.length];

		 // name of the output file
		outputFile = "ProjetPartie10_ImprovedLMSOctave5.csv";

		 // show if the file with this name already existe
		alreadyExists = new File(outputFile).exists();

		// initialisation of omega thanks to the fequencies f
		omega = new double[f.length];
		for (int i = 0; i < f.length; i++) {
			omega[i] = 2 * pi * f[i] / fs;
		}

		// initialisation of ha and hb
		haDo = new double[3][f.length];
		haRe = new double[3][f.length];
		haMi = new double[3][f.length];
		haFa = new double[3][f.length];
		haSol = new double[3][f.length];
		haLa = new double[3][f.length];
		haSi = new double[3][f.length];

		hbDo = new double[3][f.length];
		hbRe = new double[3][f.length];
		hbMi = new double[3][f.length];
		hbFa = new double[3][f.length];
		hbSol = new double[3][f.length];
		hbLa = new double[3][f.length];
		hbSi = new double[3][f.length];

		exectution();
	}
```

```
	/**
	 * method for the execution of the program
	 */
	public void exectution() {
		try {
			// use FileWriter constructor that specifies open for appending
			CsvWriter csvOutput = new CsvWriter( new FileWriter(outputFile, true), '.');
			// if the file didn't already exist then we need to write out the header line
			if (!alreadyExists) {
				for (int l = 0; l < headers.length; l++) {
					csvOutput.write(headers[l]);
				}
				csvOutput.endRecord();
			}

			// loop for writing the values
			for (int i = 0; i < sampleValue.length; i++) {
				x = sampleValue[i];
				ampDo = amplitudeValue(i, x, haDo, hbDo, 0);
				ampRe = 0.1 + amplitudeValue(i, x, haRe, hbRe, 1);
				ampMi = 0.2 + amplitudeValue(i, x, haMi, hbMi, 2);
				ampFa = 0.3 + amplitudeValue(i, x, haFa, hbFa, 3);
				ampSol = 0.4 + amplitudeValue(i, x, haSol, hbSol, 4);
				ampLa = 0.5 + amplitudeValue(i, x, haLa, hbLa, 5);
				ampSi = 0.6 + amplitudeValue(i, x, haSi, hbSi, 6);

				// change values calculated before into string and put it in an array
				String[] temp = new String[] { "" + ampDo, "" + ampRe, "" + ampMi, "" +
					ampFa, "" + ampSol, "" + ampLa, "" + ampSi };

				// change the dot to the comma and write values in the file
				for (int k = 0; k < temp.length; k++) {
					values[k][i] = dotToComma(temp[k]);
					csvOutput.write(values[k][i] + "");
				}
				// end of the line of the file
				csvOutput.endRecord();
			}
			// close the file
			csvOutput.close();
		} catch (IOException e) {
			System.out.println("Failed");
		}
	}


	/**
	 * calcul the amplitude
	 */
	public double amplitudeValue(int iterationNum, double x, double[][] ha,
			double[][] hb, int soundNum) {
		xh = 0.0;
		for (int j = 0; j < f.length; j++) {
			xh += ha[0][j] * Math.cos(omega[j] * iterationNum) + hb[0][j]
				* Math.sin(omega[j] * iterationNum);
			e = x - xh;
			ha[0][j] = (1.0 + gamma) * ha[1][j] - gamma * ha[2][j] + mu * e
				* Math.cos(omega[j] * iterationNum);
			hb[0][j] = (1.0 + gamma) * hb[1][j] - gamma * hb[2][j] + mu * e
				* Math.sin(omega[j] * iterationNum);

			ha[2][j] = ha[1][j];
			ha[1][j] = ha[0][j];
			hb[2][j] = hb[1][j];
			hb[1][j] = hb[0][j];
		}
		return (Math.sqrt(ha[0][soundNum] * ha[0][soundNum] + hb[0][soundNum]
			* hb[0][soundNum]));
	}

...
```

Annex 32: Deletion of “do” error

```
public class ProjetPartie15_NewTestOctave5 {

	/** declaration of variable used for writting the csv file **/
	String[] headers = { "do", "re", "mi", "fa", "sol", "la", "si" };
	String outputFile, changement;
	boolean alreadyExists;


	/** declaration of variables used in the reading wave file methode **/
	// bigest value of a data
	private static final double MAX_16_BIT = Short.MAX_VALUE;
	double[] sampleValue; // the array where the samples values will be stock


	/** declaration of variables used in the calcul of estimate values **/
	final int fs = 4000; // music in 4Hz
	final double pi = Math.atan(1.0) * 4;
	final double mu = 0.01;

	double[][] values;

	// array of hat, hat1, hat2
	double[][] ha, hb;

	// frequencies of siOctave4 octave5 doOctave6
	double[] f = { 493.88, 523.28, 587.33, 659.26, 698.46, 783.99, 880, 987.77, 1046.56 };

	double[] omega;
	double x, xh, e;

	/** declaration of variables used in test **/
	boolean[] played; // check the played values



	/**
	 * constructor of the program : constructor contains only initalisation
	 */
	public ProjetPartie15_NewTestOctave5(String filename) {

		// read the wave fil and stock the value in the array
		sampleValue = read(filename);

		// initialisation of the array with the number of columns (hearders) and
		// lignes (sampleValue)
		values = new double[f.length][sampleValue.length];

		// name of the output file
		outputFile = "ProjetPartie15_NewTestOctave5.csv";

		// show if the file with this name already existe
		alreadyExists = new File(outputFile).exists();

		// initialisation of omega thanks to the fequencies f
		omega = new double[f.length];
		for (int i = 0; i < f.length; i++) {
			omega[i] = 2 * pi * f[i] / fs;
		}

		// initialisation of ha and hb
		ha = new double[f.length][3];
		hb = new double[f.length][3];

		// initialisation of check array
		played = new boolean[f.length];
		for (int i = 0; i < played.length; i++) {
			played[i] = false;
		}

		exectution();
	}
```

```
/**
 * methode for the execution of the program
 */
public void exectution() {

	try {
		// use FileWriter constructor that specifies open for appending
		CsvWriter csvOutput = new CsvWriter(
				new FileWriter(outputFile, true), '.');

		// if the file didn't already exist then we need to write out the
		// header line
		if (!alreadyExists) {
			for (int i = 0; i < headers.length; i++) {
				csvOutput.write(headers[i]);
			}
			csvOutput.endRecord();
		}

		// loop for writing the values
		for (int i = 0; i < sampleValue.length; i++) {

			x = sampleValue[i];

			values[0][i] = amplitudeValue(i, x, 1);
			values[1][i] = amplitudeValue(i, x, 2);
			values[2][i] = amplitudeValue(i, x, 3);
			values[3][i] = amplitudeValue(i, x, 4);
			values[4][i] = amplitudeValue(i, x, 5);
			values[5][i] = amplitudeValue(i, x, 6);
			values[6][i] = amplitudeValue(i, x, 7);


			// check amplitude values if each note is played are not
			for (int j = 0; j < played.length; j++) {
				// check if the value is more than 0.035 when the sound begins
				//and less than 0.001 when it finishes
				if (((values[j][i] > 0.035) && !played[j]) || (played[j] && (values[j][i] > 0.001))) {
					played[j] = true;
				} else {
					played[j] = false;
				}
			}

			// make the amplitude not played at 0
			for (int j = 0; j < played.length; j++) {
				if (!played[j]) {
					values[j][i] = 0.0;
				}
			}
		}

		// writting the values
		for (int i = 0; i < sampleValue.length; i++) {
			// change values calculated before into string and put it in an
			// array make the graph in multiline
			String[] temp = new String[] { "" + values[0][i],
					"" + (values[1][i] + 0.1), "" + (values[2][i] + 0.2),
					"" + (values[3][i] + 0.3), "" + (values[4][i] + 0.4),
					"" + (values[5][i] + 0.5), "" + (values[6][i] + 0.6) };

			// change the dot to the comma and write values in the file
			for (int k = 0; k < temp.length; k++) {
				csvOutput.write((dotToComma(temp[k])) + "");
			}
			// end of the line of the file
			csvOutput.endRecord();
		}
		// close the file
		csvOutput.close();
	} catch (IOException e) {
		System.out.println("Failed");
	}

}
```

```
        /**
         * calcul the amplitude with the ajacents tones
         */
        public double amplitudeValue(int iterationNum, double x, int soundNum) {
                xh = 0.0;
                for(int i = 0; i<3; i++){
                        xh += ha[soundNum][i]*Math.cos(omega[soundNum+(i-1)]*iterationNum)
                                        + hb[soundNum][i]*Math.sin(omega[soundNum+(i-1)]*iterationNum);
                        e = x - xh;
                        ha[soundNum][i] += mu * e * Math.cos(omega[soundNum + (i-1)]*iterationNum);
                        hb[soundNum][i] += mu * e * Math.sin(omega[soundNum + (i-1)]*iterationNum);
                }
                return (Math.sqrt(ha[soundNum][1] * ha[soundNum][1] + hb[soundNum][1] * hb[soundNum][1]));
        }

…
```

Annex 33: Graphic interface

```
public class MusicalTranscription_InterfaceGraphique extends JPanel{
	private static final long serialVersionUID = 1L;
	private static final String frameTitle = "Musical Transcription";
	public final Integer[] octaves = new Integer[]{5,2,3,4,6};
	public final Integer[] hertz = new Integer[]{4000,8000,11025,16000,22050,32000};
	public final FileNameExtensionFilter filter = new FileNameExtensionFilter("Wave files","wav");
	public JFrame myFrame;
	public File file;
	// Graphic objects
	public JTextField path, output;
	public JButton search, execute;
	public JComboBox<Integer> myOctaves, myHertz;
	public JFileChooser myFileChooser;
	public JLabel imputName, octave, frequency, outputName;

	/**
	 * Constructor of the class
	 */
	MusicalTranscription_InterfaceGraphique(){
		this.initComponent();
		this.createJFrame();
	}

	/**
	 * Initialisation of the panel
	 */
	public void initComponent(){
		this.setLayout(null);

		//initialisation
		this.imputName = new JLabel("Search the imput file : ");
		this.path = new JTextField();
		this.search = new JButton(new ImageIcon("ressources/recherche.gif"));
		this.myOctaves = new JComboBox<Integer>(octaves);
		this.myOctaves.setSelectedItem("5");
		this.myFileChooser = new JFileChooser();
		this.myFileChooser.setFileFilter(filter);
		this.myHertz = new JComboBox<Integer>(hertz);
		this.octave = new JLabel("octave");
		this.frequency = new JLabel("SampleFrequency");
		this.execute = new JButton("execute");
		this.outputName = new JLabel("output file name : ");
		this.output = new JTextField();

		//add the listenner to the list
		this.search.addActionListener(new ChooserListener());
		this.execute.addActionListener(new calculate());

		//placement in the panel
		this.imputName.setBounds(10, 10, 200, 20);
		this.path.setBounds(10, 30, 340, 20);
		this.search.setBounds(350, 30, 20, 20);
		this.octave.setBounds(10, 60, 50, 20);
		this.myOctaves.setBounds(60, 60, 50, 20);
		this.frequency.setBounds(160, 60, 110, 20);
		this.myHertz.setBounds(270, 60, 100, 20);
		this.outputName.setBounds(10, 100, 200, 20);
		this.output.setBounds(10, 120, 250, 20);
		this.execute.setBounds(280, 100, 90, 40);

		//add into the panel the object initialised
		this.add(imputName);
		this.add(path);
		this.add(search);
		this.add(octave);
		this.add(myOctaves);
		this.add(frequency);
		this.add(myHertz);
		this.add(outputName);
		this.add(output);
		this.add(execute);
	}
```

```
	/**
	 * Create the frame where we put the panel
	 */
	private void createJFrame(){
		this.myFrame = new JFrame();
		this.myFrame.setTitle(frameTitle);
		this.myFrame.setSize(400, 200);
		this.myFrame.setContentPane(this);
		this.myFrame.setLocationRelativeTo(null);
		this.myFrame.setVisible(true);
		this.myFrame.setResizable(false);
		this.myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	/**
	 * permit the list to choose the value in the list
	 */
	private class ChooserListener implements ActionListener{
		public void actionPerformed(ActionEvent e) {
			int returnVal = myFileChooser.showOpenDialog(MusicalTranscription_InterfaceGraphique.this);
			if (returnVal == JFileChooser.APPROVE_OPTION) {
				file = myFileChooser.getSelectedFile();
				path.setText(file.getPath());
			}
		}
	}

	/**
	 * send the informations that we kept before to the program that trascript the music into a CSV file
	 */
	private class calculate implements ActionListener{
		public void actionPerformed(ActionEvent e) {
			new MusicalTranscription_Program(path.getText(), Integer.parseInt(myHertz.getSelectedItem()+""),
				Integer.parseInt(myOctaves.getSelectedItem()+""), output.getText()+".csv");
		}
	}

	/**
	 * Main
	 */
	public static void main(String[] args){
		new MusicalTranscription_InterfaceGraphique();
	}
}
```

Glossary

C language:
Programming language appeared in 1972 by Ritchie for system programming.

DUT Informatique de gestion :
Diplôme Universitaire de Technologie option Informatique de Gestion.
The DUT is a diploma that we earn after two year in an IUT. The translation of my studying speciality "Informatique de gestion' is a kind of "computer management". One of our speciality is "safety programming" for big company or bank.

Eclipse software:
It is a mutli-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It is also free and open source software.

IUT:
"Institut Universitaire et technologie"

JavaDoc:
Developed by Sun Microsytems, the JavaDoc permit a visualisation of Java's classes' documentation.

Java language:
Programming language appeared in 1995 by Sun Microsystems for object programming.

References

Hachinohe city:
http://en.wikipedia.org/wiki/Hachinohe,_Aomori
http://fr.wikipedia.org/wiki/Hachinohe

Java documentation:
http://docs.oracle.com/javase/6/docs/api/

Read CSV file:
http://www.dijit.fr/export-csv-en-java-tutorial-reflection-annotations/
http://www.csvreader.com/java_csv_samples.php

Read Wave file:
http://www.javafr.com/codes/JOUER-SON-WAV-JAVA_29515.aspx
http://introcs.cs.princeton.edu/java/stdlib/