# MA1 Acoustic Labs 2
# Analysis and additive synthesis of a piano note

Author : Hervé Lissek

October 4, 2016

| | |
|---|---|
| Duration : | 2 × 4 hours |
| Support : | Data file *notepiano.mat* |
| Prerequisite : | Signal processing |
| References : | M. Rossi, *Audio*, pages 115-178, Presses Polytechniques et Universitaires Romandes, 2007 |
| | Wikipedia page on synthesizers / ADSR section |
| Hardware & Software : | computer with a sound card |
| | Matlab licence |
| | headphones |
| Deadline for the reports (Labs 01 & 02) : | October 18, 2016 |

*Every periodic signal can be considered as a linear combination of pure sinusoidal signals (harmonics), the frequencies of which are integer multiples of the lowest frequency (fundamental)* Jean Baptiste Joseph Fourier (1812) (see fig. 1).

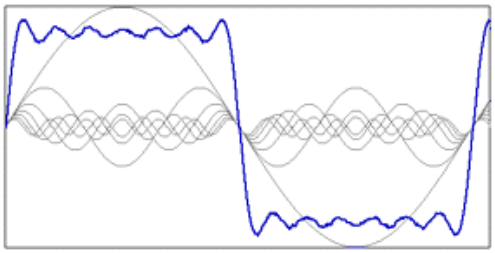


**Figure 1** – Square waveform (in blue) and its constitutive harmonics (grey)

## Objectives of the lab

This work aims at synthesizing an audio signal resembling an original recording (here a piano note) by additive sound synthesis. This technique allows shaping a given musical timbre by adding harmonic sinusoidal signals. In a first step, a preliminary study aiming at identifying the frequency components of the original signal will be undertaken. However, the frequency components are not sufficient in musical acoustics, and one should also consider the occurrence of transients in the time-domain waveform (attacks/extinctions). All these time- and frequency-domain characteristics of the original sound will be then considered in the synthesis step. Last, the limitation of the resulting synthetic sounds will be discussed.

At the end, you should be able to :

1. Calculate and display the power spectrum of a given signal ;
2. Characterize the variation of a time-domain waveform ;
3. Identify the harmonic and the time envelope of a signal.

*Hint* : prepare the template of your report (Word, LaTeX, etc.) in order to spare time at the end of the laboratory.

# 1 Analysis and additive synthesis of a stationary sound

## 1.1 Identification of the fundamental frequency and its harmonics

In this part, you will analyse the frequency spectrum of the signal.

1. Import the measurement data in the file *notepiano.mat* corresponding to the recording of a piano note. You can use the function *load('notepiano.mat')* or double-click on the file in the matlab explorer.
   The values $y$ are the amplitudes of the signal s(t) in [Pa] and *fs* corresponds to the sampling frequency [Hz].
2. Display the time-domain waveform $s_0$. Listen and comment what you hear.
3. Compute the FFT on a well chosen portion of the signal $s_0$ (use $nfft = 8192$). Display the result.

Now, you should identify the fundamental and its harmonics in signal $s_0$. There is no simple solution to automatically detect these frequencies, due to the energy decrease with frequency.

4. Identify, **manually**, the fundamental $f_0$ and its 7 first harmonics $f_i$ (the frequency values, but also the amplitudes $A_i(dB)$) ; you can display the corresponding values of frequencies and amplitude peaks in the FFT representation, using the *data cursor tool*) and the *Export Cursor Data to Workspace* function (right-click on each Datatip) ;
5. Calculate the frequency ratios $\frac{f_i}{f_0}$ and amplitude ratios $\frac{A_i(dB)}{A_0(dB)}$ of each harmonics with respect to the fundamental ;
6. Compare the measured harmonic frequencies with the theoretical harmonics with the same fundamental. Up to which harmonic $N$ of the original signal is it similar to the theoretical model ? In the following, you will only consider these $i \leq N$ harmonics.

## 1.2 Synthesis part 1 : elementary signals

Starting from the fundamental frequency and the measured harmonics, you will create a signal by additive synthesis.

1. First, create the following sinusoidal signal :

$$s_1(t) = \sum_{i=0}^{N} a_i \sqrt{2} sin(2\pi f_i t), \tag{1}$$

   where $a_i$ is the rms value of the amplitude corresponding to harmonic $f_i$ of the original signal (to be derived knowing $A_i(dB)$)
2. Compute the FFT of signal $s_1$ and compare to the original signal $s_0$. Display original and the synthetic signals on the same graph.
3. Listen to original signal $s_0(t)$ and synthetic signal $s_1(t)$, and conclude.

*Reminder :* sound pressure threshold $p_0 = 20\mu Pa$.

# 2 Analysis and synthesis of the time envelope

## 2.1 Extracting the time envelope

The timbre depends mainly on the sound spectrum as seen previously. But the transients (attacks/extinctions), which is characterized by the time enveloppe of the signal, is also strongly affecting the timbre of an instrument.
The ADSR enveloppe (Attack Decay Sutain Release) is generally used in synthesizers and allows describing the sound amplitude evolution in time of a musical instrument. It is modelled by 4 parameters (see fig. 2(a)) :

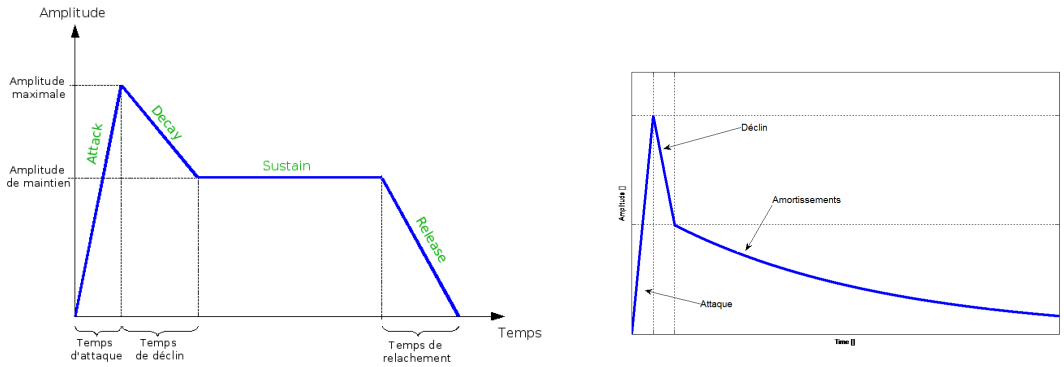
— the attack : describes the time ($t_A$) required for the sound to reach the maximal amplitude ;

— the decay : indicates the duration ($t_D - t_A$) after the first peak to reach the stationnary state (main state of the sound) ;

— the sustain : describes the duration ($t_S - t_D$) of the stationnary state (constant amplitude) ;

— the release : indicates the duration ($t_R - t_S$) between the stationnary state and the complete extinction.

In the proposed piano note, we assume the steady state is absent (the player does not activate the “sustain” pedal).




(a) The ADSR envelope and its 4 parameters

(b) The ADR enveloppe studied here and its 3 parameters

**Figure 2** – Enveloppes temporelles

The sound that you study will be modelled by a relatively short attack and sustain times, followed by an exponential release (cf. fig. 2(b)). The release phase can be modelled as :

$$y_{release}(t) = s_1(t_D)e^{-\alpha(t-t_D)} \text{ for } t > t_D \tag{2}$$

You will extract the time envelope of the signal (as illustrated on Figure 2) by applying the *Hilbert* transform.

1. Calculate the envelope $e(t)$ of signal $s_0(t)$ by Hilbert transform. For that, apply the Hilbert transform on signal $s_0(t)$, then filter the absolute value of this transform with a $2^{nd}$ order low-pass Butterworth filter with cut-off frequency $f_c = 100$ Hz ;
   **Hint :** useful Matlab functions : hilbert(), butter(), filter()
2. Draw the resulting envelope and overlay the result to the original signal $s_0(t)$ on a same graph.

From the achieved envelope :

3. Measure the attack, decay and release times ;
4. Calculate the damping coefficient $\alpha$ (useful Matlab function : polyfit()).
5. Draw the envelope $e(t)$ with the 3 phases as in figure 2(b).

### 2.2 Synthesis part 2 : temporal envelope synthesis

In this second synthesis part, you will account for the time envelope $e(t)$, identified in the previous section, to your synthetic sound.

1. Create signal $s_2(t)$ such as :

$$s_2(t) = s_1(t) \times e(t) \tag{3}$$

2. Draw $s_1$ and $s_2$ waveforms, as well as their frequency spectra.
3. Listen to the signals and comment.

## 3 Optional : Time-frequency analysis

### 3.1 Spectrogram of the original signal

In this last part, you will investigate some ways for improving the synthetic signal, thanks to a time-frequency analysis of the recorded signal.

1. Compute the spectrogram of the original signal $s_0$ (consider $nfft = 512$ and an overlap of 0.5). You can consult the Matlab documentation :

```
doc spectrogram
```

Knowing a priori the values of the harmonics, we favour a better time resolution than frequency, justifiying such a low value for nfft. You can inspire from the following code lines :

```
Ltotal = length(s);
Lwindow = nfft;
Loverlap = ... ; % overlap duration (in samples)
nbwin = fix((Ltotal-Lwindow)/(Lwindow-Loverlap))+1; % number of windows with ↩
    the given overlap
YdB = zeros(nfft/2+1,nbwin);
vt2 = linspace(0,length(s)/fs,nbwin);

for numwin = 1:nbwin
    vec = (numwin-1)*(Lwindow - Loverlap)+1:(numwin-1)*(Lwindow - Loverlap)+↩
        Lwindow;
    sigwin = s(vec);
    [YdB(:,numwin),vf2] = myFFT(...);
end

imagesc(...); axis xy
colorbar
```

2. Draw the time evolution of the fundamental and its harmonics. You can ease this by finding the index of each frequency vector corresponding to the harmonic frequencies vFreqHarm :

```
    [valeur,indice] = min(abs(vf2 - vFreqHarm(k))); % k correspondant a la kieme↩
        -1 harmonique
    plot(vt2,log(YdB(indice,:)));
```

3. Comment.