Manipulations Audio pour ISN

Christophe Barès

3 février 2014

L'ensemble de ce document a été réalisé avec *IPython notebook*. Sous windows, l'ensemble des bibliothèques utilisées ici peuvent être facilement installées à l'aide de la distribution Python(x,y) librement téléchargeable à l'adresse : http://code.google.com/p/pythonxy/

Ce logiciel n'est pas nécessaire à la reproduction des commandes proposées, qui peuvent être utilisées avec d'autres installations de python.

Importations utiles pour l'ensemble du document (la première ligne n'est valable que sous IPython) :

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import IPython.display as dsp
import pandas as pd
import pylab
pylab.rcParams['figure.figsize'] = (9.0, 7.0)
pylab.rcParams['font.weight']= 'normal'
pylab.rcParams['font.size']= 18
#pylab.rcParams.keys()
```

1 Qu'est-ce qu'un son ?

1.1 Tout d'abord, un peu de physique...

Définition

Une onde acoustique est une perturbation mécanique (onde de compression-dilatation du milieu) qui se propage dans un milieu matériel.

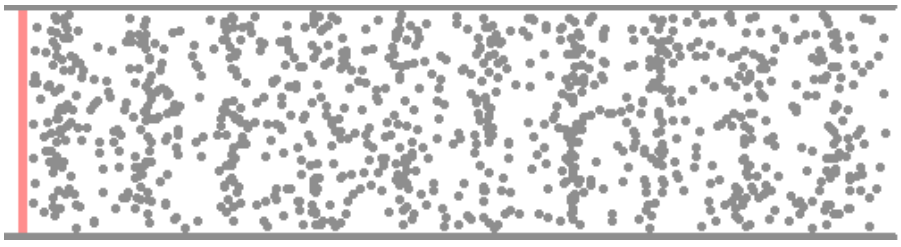


FIGURE 1 – source : web-sciences

Ce milieu peut être aussi bien un **gaz** (l'air), mais aussi un **liquide** (l'eau) et même un **solide** (un métal).

Onde sinusoïdale

Pour un movement d'exitation sinusoïdale, on peut observer la pression (ou la vitesse des molécules) :

– à un instant donné :

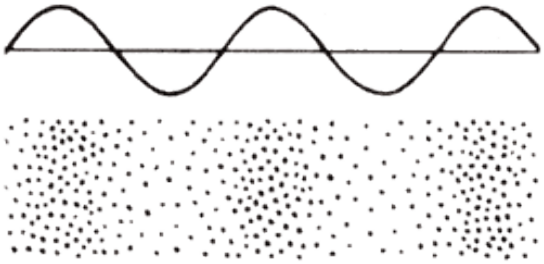
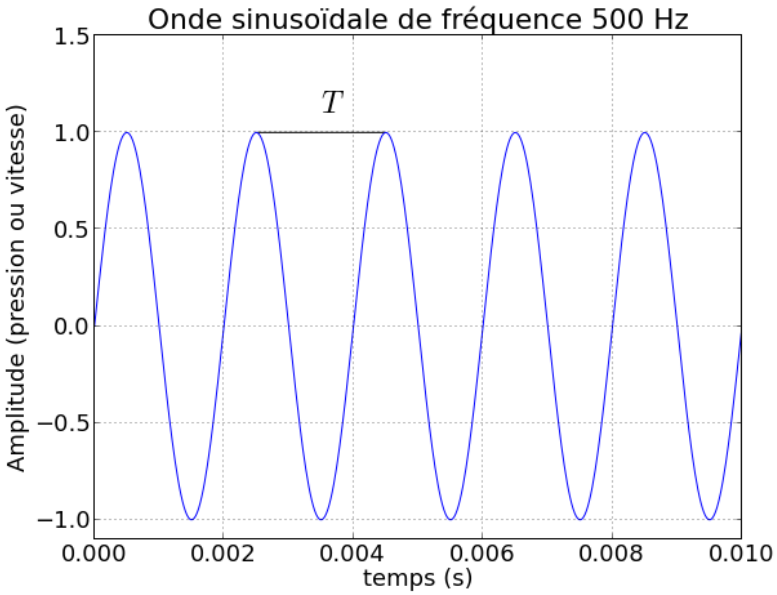


FIGURE 2 – source : musicologie.org

Sur cette “photo”, on peut mesurer la longueur d’onde λ (ou péridode spatiale) (en m).

– ou à un endroit donné :

In [2]:
```
def sinu(t, f=100., phi=0):
    w = 2*np.pi*f
    return np.sin(w*t+phi)

f = 500. #Hz
Fs = 44100 # échantillons par seconde
t = np.linspace(0, 5, 5*Fs)
x = sinu(t, f)

plt.plot(t,x)
plt.xlabel('temps (s)')
plt.ylabel('Amplitude (pression ou vitesse)')
plt.title(u'Onde sinusoïdale de fréquence %d Hz'%f)
plt.grid()
plt.hlines(1, 0.0025, 0.0025+1/f, 'k')
plt.annotate('$T$', (0.0035, 1.1), size=24)
plt.axis([0,0.01,-1.1,1.5]);
```



Ici on peut mesurer la période T du signal (en s) : pour chaque instant t, on a : $x(t+T) = x(t)$

Cette période est l’inverse de la fréquence $f = \frac{1}{T}$, qui s’exprime en Hz

Vitesse du son

La vitesse du son est définie par : $c = \dfrac{\lambda}{T}$

Matériaux	c en m·s^{-1}	Matériaux	c en m·s^{-1}	Matériaux	c en m·s^{-1}	Matériaux	c en m·s^{-1}
Air	343	PVC (souple)	2 000	Eau	1 480	PVC (rigide)	2 400
Glace	3 200	Béton	3 100	Verre	5 300	Hêtre	3 300
Acier	5 600 à 5 900	Granite	6 200	Plomb	1 200	Péridotite	7 700
Titane	4 950	Sable sec	10 à 300				

source : Wikipedia

Spectre audible

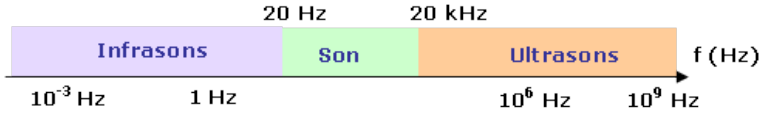


FIGURE 3 – source : web-sciences

Ce spectre dépend des individus, ainsi que de l'âge…

Spectre musical

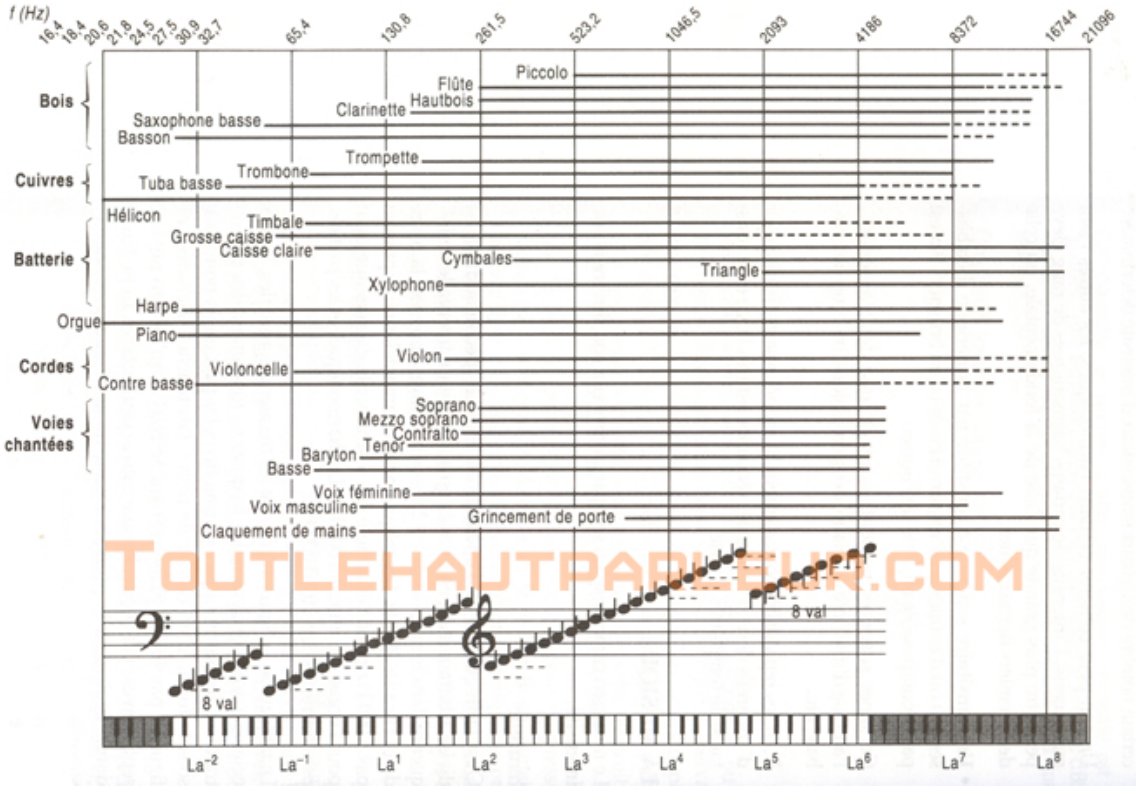


FIGURE 4 – source : TeamAudion

Spectre d'un signal

Pour un signal périodique => décomposition en **série de Fourier** : $f(x) = \sum_{n=-\infty}^{+\infty} c_n(f) e^{i2\pi \frac{n}{T} x}$

avec les coefficients $c_n(f)$, appelés **coefficients de Fourier** de f, définis par la formule :

$$c_n(f) = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-i2\pi \frac{n}{T} t} \mathrm{d}t$$

Pour un signal non périodique => **transformée de Fourrier continue**

$$\mathcal{F}(f) : \omega \mapsto \hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\, \mathrm{e}^{-i\omega t}\, \mathrm{d}t$$

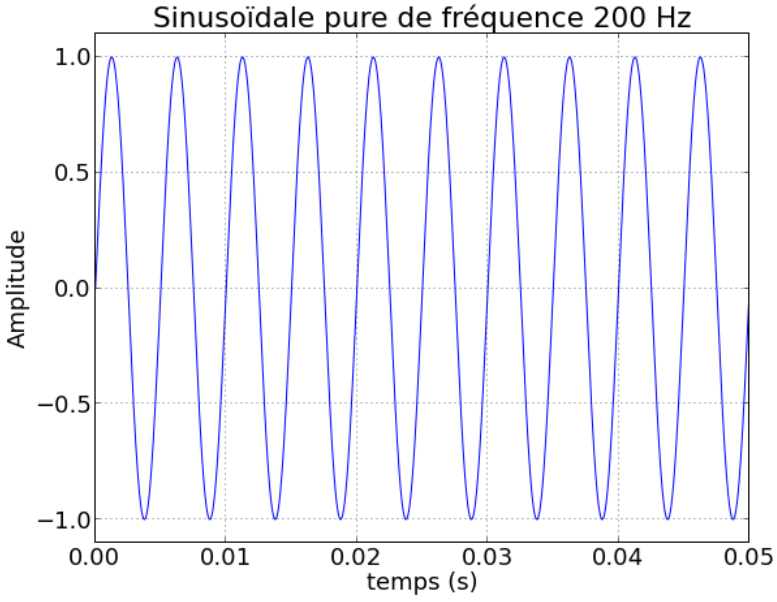
avec t en secondes et ω la pulsation (en rad.s^{-1}).Pour un signal numérique => **transformée de Fourier discrète** (TFD)

$$\mathrm{S}(k) = \sum_{n=0}^{\mathrm{N}-1} s(n) \cdot e^{-2i\pi k \frac{n}{\mathrm{N}}} \qquad \text{pour} \qquad 0 \leqslant k < \mathrm{N}$$

On utilise en pratique la **transformée de Fourier rapide** (FFT ou Fast Fourier Transform) qui est un algorithme particulier de TFD.

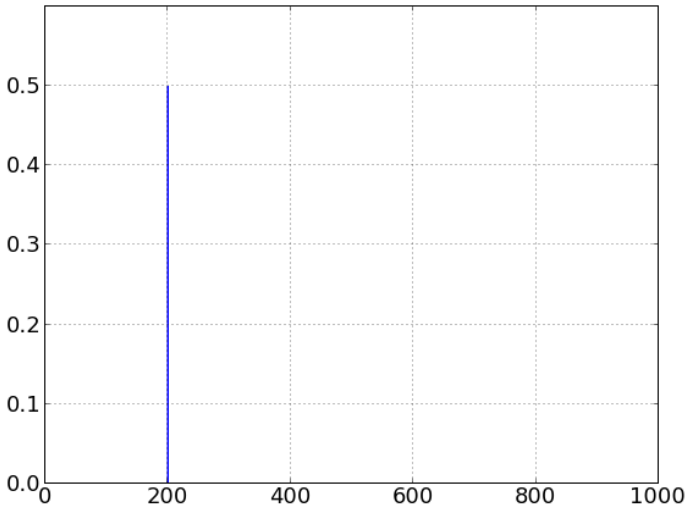
Sinusoïdale pure

In [3]:
```
f1 = 200. #Hz
Fs = 44100
t = np.linspace(0, 5, 5*Fs)
x = sinu(t, f1)

plt.plot(t,x)
plt.xlabel('temps (s)')
plt.ylabel('Amplitude')
plt.title(u'Sinusoïdale pure de fréquence %d Hz'%f1)
plt.grid()
plt.axis([0,0.05,-1.1,1.1]);
```



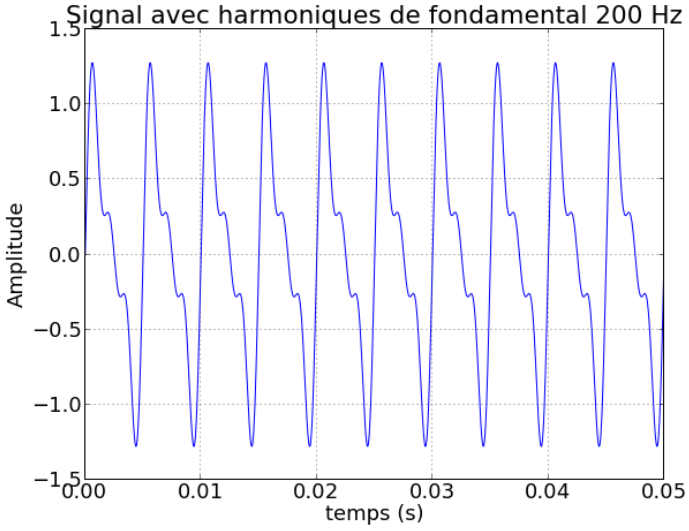
In [4]:
```
from maudio import Audio
Audio(data=x, rate=Fs)
```

In [5]:
```
freq = np.fft.fftfreq(len(t), d=1./Fs)
y = np.fft.fft(x, len(t))

plt.plot(freq, np.abs(y)/len(y))
plt.grid()
plt.axis([0, 1000, 0,.6]);
```

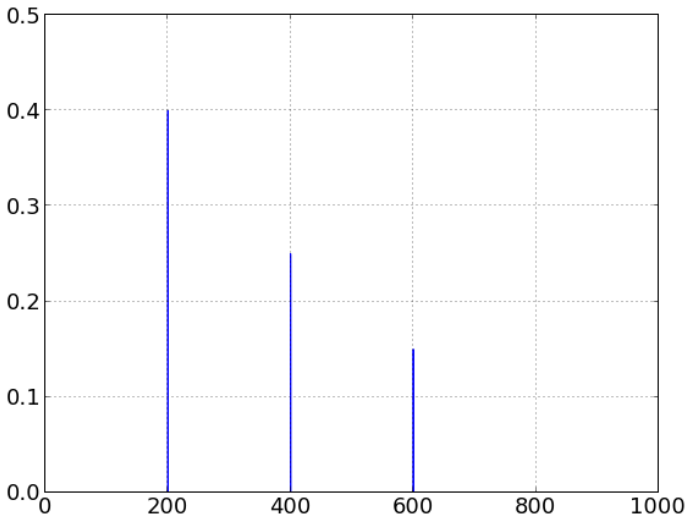


Sinusoïdale avec harmoniques

In [10]:
```
f1 = 200. #Hz
t = np.linspace(0, 5, 5*Fs)
x = .8*sinu(t, f1) + 0.5* sinu(t, 2*f1) + 0.3* sinu(t, 3*f1)

plt.plot(t,x)
plt.xlabel('temps (s)')
plt.ylabel('Amplitude')
plt.title(u'Signal avec harmoniques de fondamental %d Hz'%f1)
plt.grid()
plt.axis([0,0.05,-1.5,1.5]);
```

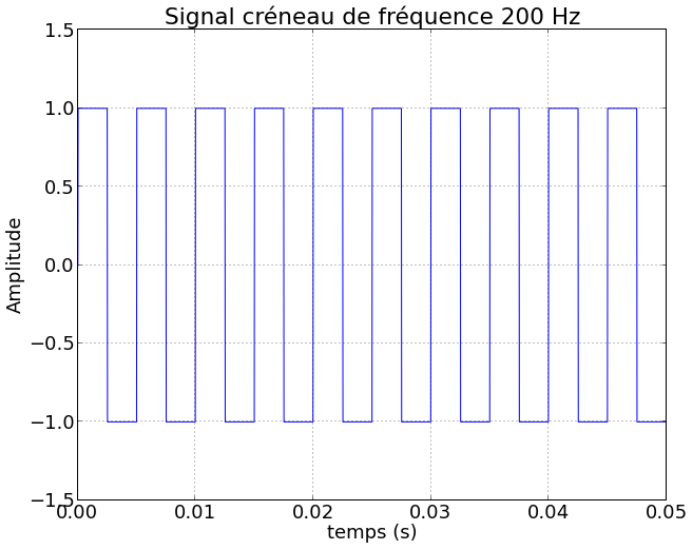


In [12]:
```
freq = np.fft.fftfreq(len(t), d=1./Fs)
y = np.fft.fft(x, len(t))

plt.plot(freq, np.abs(y)/len(y))
plt.grid()
plt.axis([0, 1000, 0,.5]);
```



Signal créneau

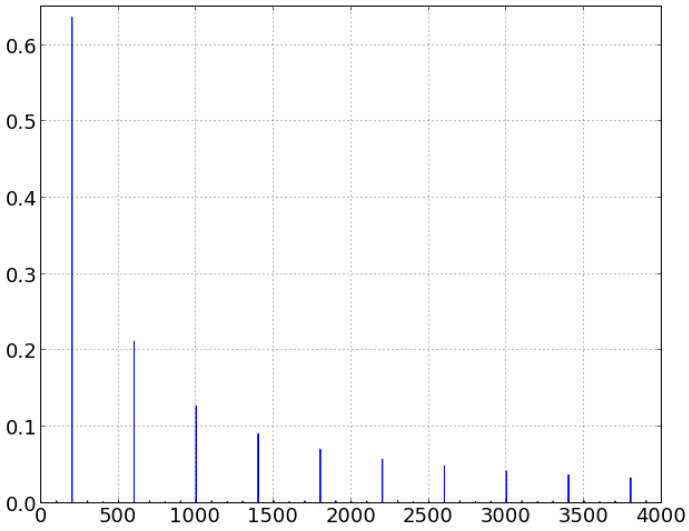
In [10]:
```
f1 = 200. #Hz
t = np.linspace(0, 5, 5*Fs)
x = np.sign(sinu(t, f1))

plt.plot(t,x)
plt.xlabel('temps (s)')
plt.ylabel('Amplitude')
plt.title(u'Signal créneau de fréquence %d Hz'%f1)
plt.grid()
plt.axis([0,0.05,-1.5,1.5]);
```



In [11]:
```
from maudio import Audio
Audio(data=x, rate=Fs)
```

Out [11]: <maudio.Audio at 0x3d53e50>

In [12]:
```
freq = np.fft.fftfreq(len(t), d=1./Fs)
y = np.fft.fft(x, len(t))

plt.plot(freq, np.abs(y)/len(y))
plt.grid()
plt.axis([0, 4000, 0,.65]);
```



Coefficients de Fourier du signal carré :

$$c_n(f) = \frac{4}{\pi n} \text{ si n est impaire} c_n(f) = 0 \text{ si n est impaire}$$

In [13]:
```
pd.DataFrame(4/np.pi/np.arange(1,15,2)/2, columns=['Amplitudes'],
             index=['%d'%i for i in range(1,10,2)]).T
```

Out [13]:

	1	3	5	7	9
Amplitudes	0.63662	0.212207	0.127324	0.090946	0.070736

1.2 Chaîne de traitement d'un signal (sonore ou non)...

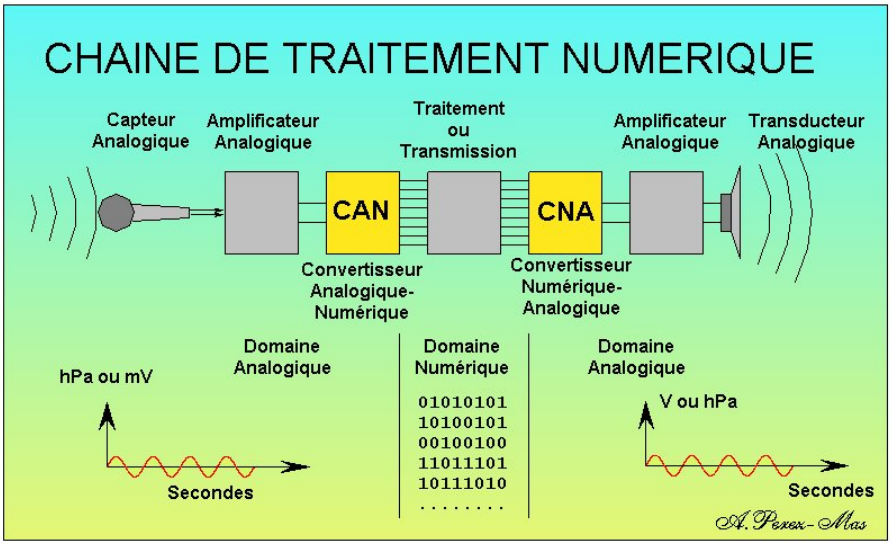


FIGURE 5 – source : Perez-Mas

CAN : Conversion Analogique-Numérique

source : lemm univ -Lille1

1. Signal analogique
2. Échantillonnage
3. Quantification
4. Signal numérisé

Théorème d'échantillonnage de Nyquist-Shannon

> La représentation discrète d'un signal par des échantillons régulièrement espacés exige une fréquence d'échantillonnage supérieure au double de la fréquence maximale présente dans ce signal.

$$F_e \geq 2F_{\max}$$

Conséquence : Repliement de spectre

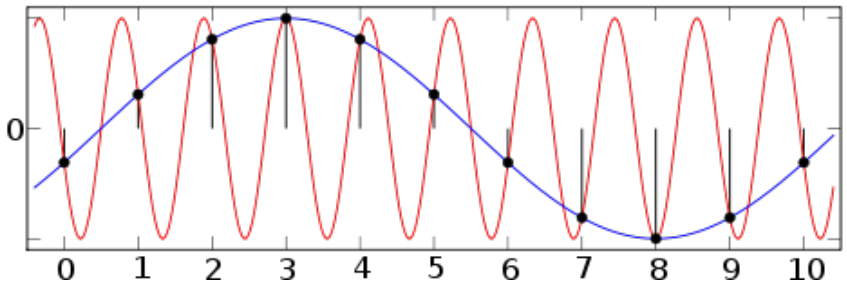


FIGURE 6 – source : Wikipedia

Fréquence du signal : 0,9 Hz

Fréquence d'échantillonnage : 1,0 Hz

Fréquence repliée : 0,1 Hz

Conséquence : effet de Moiré

FIGURE 7 – source Wikipedia

Quantification

Conversion du signal echantillonné vers un ensemble fini de valeurs discrète.

Résolution : Nombre de bits q du convertisseur (soit 2^q valeurs discrètes)

Pas de quantification : $\delta V = \dfrac{\Delta V}{N-1} = \dfrac{\Delta V}{2^q 1}$

exemple : Un Arduino posséde 6 DAC 10 bits (A0,... A5) et fonctionne sur 5 V

$q = 10$, $N = 1024$, et $\delta V = \dfrac{5}{1023} = 4,89$ mV

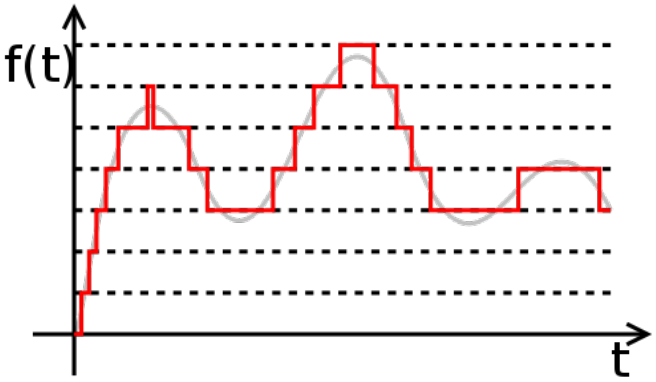


FIGURE 8 – source : wikipedia

Conséquence : Bruit de quantification

Pour un signal de répartition uniforme (triangle pleine échelle), SQNR $= 6.q$ dB

Mais en général, c'est pire : faible amplitude, signal non uniforme

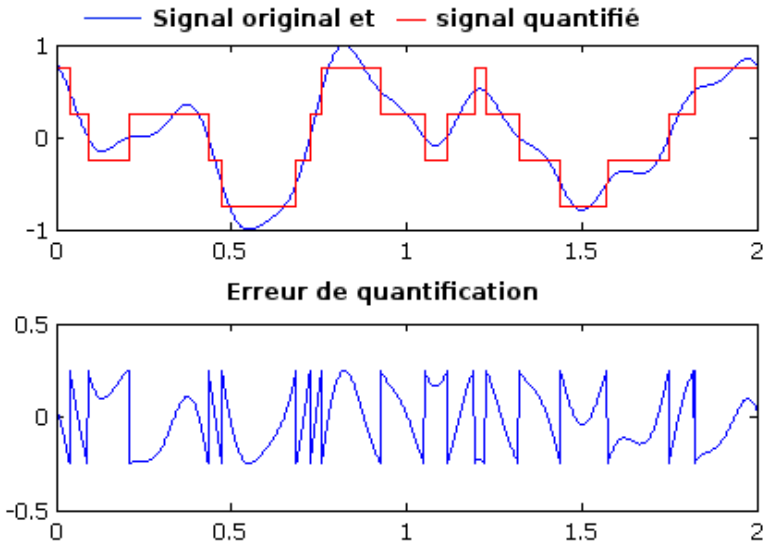


FIGURE 9 – source : wikipedia

Signaux audios

Support	GSM	CD audio	DVD video	DVD HD
Frequence d'échantillonnage(Hz)	8000	44100	48000	96000
Résolution (bits)	13	16	16	24
Canaux	1	2	1 à 6	1 à8

Fichiers audios

Fichiers	wav	mp3	ogg Vorbis	AAC	FLAC	Wavpack
Type de compression	néant	avec pertes	avec pertes	avec pertes	sans pertes	avec ou sans pertes
Taux de compression	0%	90%	90%	90%	50-75%	40% ou 85%
Exemple de taille (Mo)	38,6	3,5	3,2	3,5	23,4	23,2/5,5

1.3 Applications

Chargement de fichiers wav

Format du fichier Wav :

- Entête de 44 octets
- suivi des échantillons bruts (PCM ou Pulse Code Modulation)
- échantillons entrelacés par canaux
- Petit boutiste (*little endian*)

Bloc de déclaration d'un fichier au format WAVE

Décalage	nb d'octets	Fonction
0	4	constante 'RIFF'
4	4	taille du fichier - 8
8	4	Constante 'WAVE'

Bloc décrivant le format audio

Décalage	nb d'octets	Fonction
0xC	4	Constante 'fmt'
0x10	4	Nombre d'octets du bloc - 8 (0x10)
0x14	2	Format du stockage dans le fichier (1 : PCM, ...)
0x16	2	Nombre de canaux (de 1 à 6)
0x18	4	Fréquence d'échantillonnage (44100)
0x1C	4	Nombre d'octets à lire par seconde
0x20	2	Nombre d'octets par bloc d'échantillonnage (2o * 2 canaux)
0x22	2	Nombre de bits par échantillon (16)

Bloc des données

Décalage	nb d'octets	Fonction
0x24	4	Constante 'data'
0x28	4	Nombre d'octets des données = Taille du fichier - 44
0x3C		Données entrelacées par canaux

Exemple d'accès :

In [16]:
```
from struct import unpack
word = '<h' # 2o
dword = '<i' #4o
infile = open('licorne/track01.sym9.wav', 'rb')
infile.read(4)
```

Out [16]: 'RIFF'

In [17]:
```
infile.seek(0x4)
print "taille du fichier:", unpack(dword, infile.read(4))[0]+8
```

```
taille du fichier: 154738124
```

In [18]:
```
infile.seek(0x8)
infile.read(4)
```

Out [18]: 'WAVE'

In [19]:
```
infile.seek(0xC)
infile.read(4)
```

Out [19]: 'fmt '

In [20]:
```
infile.seek(0x18)
Fs = unpack(dword,infile.read(4))[0]
print "fréquence échantillonnage:", Fs
```

```
fréquence échantillonnage: 44100
```

In [21]:
```
infile.seek(0x22)
q = unpack(word, infile.read(2))[0]
print "Résolution:", q
```

```
Résolution: 16
```

In [22]:
```
infile.seek(0x24)
infile.read(4)
```

Out [22]: 'data'

In [23]:
```
infile.seek(0x28)
print "durée: %d s"% (unpack(dword, infile.read(4))[0]*8/q/Fs/2)
```

```
durée: 877 s
```

In [25]:
```
infile.close()
```

Analyse spectrale

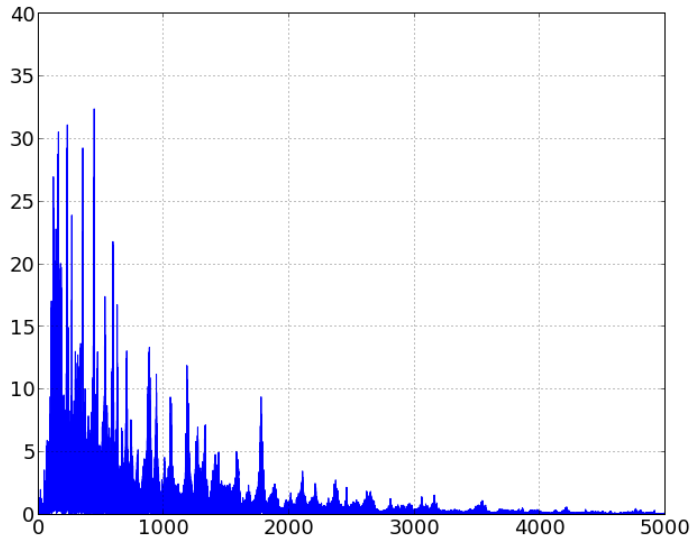
On a vu la FFT pour calculer le spectre d'un signal.

Sur un signal réel, comme un morceau de musique, le spectre devient trop riche.

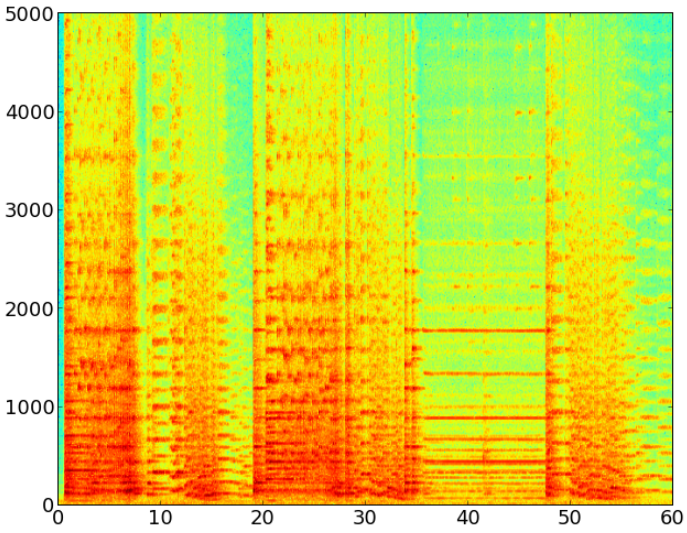
=> réduire le temps d'analyse = Transformée de Fourier à temps court

In [84]:
```
infile = open('sony-bmg/track04.sym9.wav', 'rb')
infile.seek(0x18)
Fs = unpack(dword,infile.read(4))[0]
t = np.linspace(0, 60 ,60*Fs)

infile.seek(44)
x = np.zeros(len(t))
for c in xrange(len(t)):
    g, d = unpack('<2h', infile.read(4))
    x[c] = g
    c += 1
```

In [86]:
```
freq = np.fft.fftfreq(len(t), d=1./Fs)
freq = np.linspace(0, Fs, len(t))
y = np.fft.fft(x, len(t))

plt.plot(freq, np.abs(y)/len(y))
plt.grid()
plt.axis([0, 5000, 0, 40]);
```



In [104]:
```
plt.specgram(x, NFFT=2*2048, Fs=Fs, noverlap=0)
plt.axis([0,60,0,5000]);
```



In [64]:
```
from maudio import Audio
Audio(data=x, rate=Fs)
```

Out [64]: <maudio.Audio at 0x4404b50>

Manipulation de fréquence

A. Égaliseur sonore

Banque de filtres sur plusieurs plages de fréquence

C. Détection de tempo

Utilisé par les DJ pour mixer des 'samples' entre eux

B. Changement de vitesse

Accélérer un son sans changer son changement de timbre

4. Analyse CD

5. Synthèse de vocal