Signaux, Sons et Images pour l'Informaticien: Fourier

Diane Lingrand

Polytech SI3

2018 - 2019

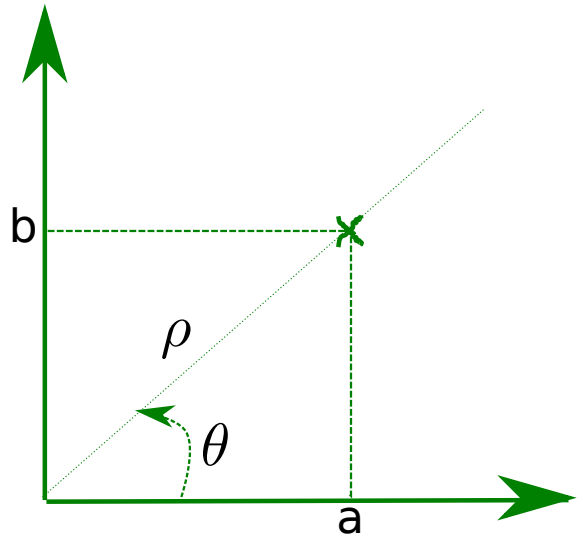
Outline

Outline

Nombre imaginaire

$$z = a + i\,b = \rho\,(\cos\theta + i\sin\theta) = \rho\,e^{i\theta}$$

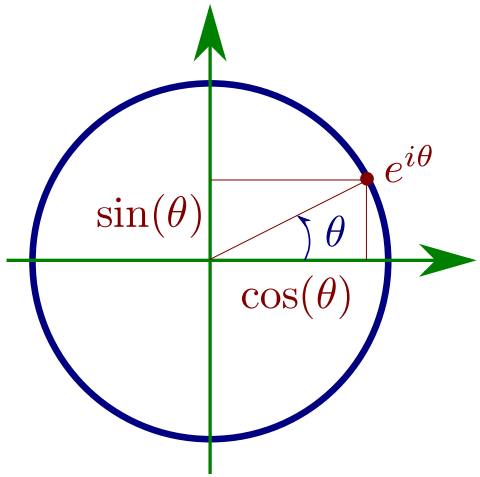
- partie réelle : a ou $\rho\cos(\theta)$
- partie imaginaire : b ou $\rho\sin(\theta)$
- module : ρ ou $\sqrt{a^2 + b^2}$
- argument : θ ou $\arctan\frac{b}{a}$



Nombre imaginaire (2)

- addition : $z_1 + z_2 = (a_1 + i\,b_1) + (a_2 + i\,b_2) = (a_1 + a_2) + i\,(b_1 + b_2)$
- produit : $\begin{aligned} z_1 * z_2 &= (a_1 + i\,b_1) * (a_2 + i\,b_2) \\ &= (a_1 * a_2 - b_1 * b_2) + i\,(a_1 * b_2 + a_2 * b_1) \\ &= \rho_1 * \rho_2\, e^{i(\theta_1+\theta_2)} \end{aligned}$

- cosinus : $\cos\theta = \dfrac{e^{i\theta} + e^{-i\theta}}{2}$
- sinus : $\sin\theta = \dfrac{e^{i\theta} - e^{-i\theta}}{2i}$
- $e^{i(\theta_1+\theta_2)} = e^{i\theta_1}\, e^{i\theta_2}$
- $e^{i\theta n} = (e^{i\theta})^n = (e^{in})^\theta$



Somme

$$x \neq 1 : \sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}$$

d'où :

$$\begin{aligned}
\theta \neq 0 \mod 2\pi : \sum_{k=0}^{n} e^{ik\theta} &= \sum_{k=0}^{n} (e^{i\theta})^k \\
&= \frac{1-e^{i(n+1)\theta}}{1-e^{i\theta}} \\
&= \frac{e^{i(n+1)\frac{\theta}{2}}}{e^{i\frac{\theta}{2}}} \frac{\sin((n+1)\frac{\theta}{2})}{\sin(\frac{\theta}{2})} \\
&= e^{in\frac{\theta}{2}} \frac{\sin((n+1)\frac{\theta}{2})}{\sin(\frac{\theta}{2})}
\end{aligned}$$

Outline

Transformée de Fourier

- Séries de Fourier :

$$s(t) = \sum_{f=-\infty}^{+\infty} c_n e^{2i\pi \frac{nt}{T}}, t \in [-\frac{T}{2}, \frac{T}{2}]$$

avec $c_n = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} s(t) e^{-2i\pi n \frac{t}{T}} dt = \frac{1}{T} S(\frac{n}{T})$.

- Transformée de Fourier :

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-2i\pi ft} dt$$

- Spectre d'amplitude : $|S(f)|$

Quelques propriétés de la transformée de Fourier

- linéarité : $\lambda_1\, s_1(t) + \lambda_2\, s_2(t) : \lambda_1\, S_1(f) + \lambda_2\, S_2(f)$
- produit : $s_1(t).s_2(t) : S_1(f) * S_2(f)$
- convolution : $s_1(t) * s_2(t) : S_1(f).S_2(f)$

Convolution :

$$(s_1 * s_2)(t) = \int_{-\infty}^{\infty} s_1(\tau) s_2(t-\tau) d\tau$$

Transformée de Fourier discrète (TFD)

$$s = \{sn = s(nT_e), n = 0..N-1\}$$

$$S(f) = TFD(s) = X(f) = \sum_{n=0}^{N-1} s(nT_e)\, e^{-2i\pi f \frac{n}{f_e}}$$

- $X(f)$ est complexe
- $X(f)$ est f_e-périodique
- Algorithme de transformée de Fourier rapide (FFT)

TFD

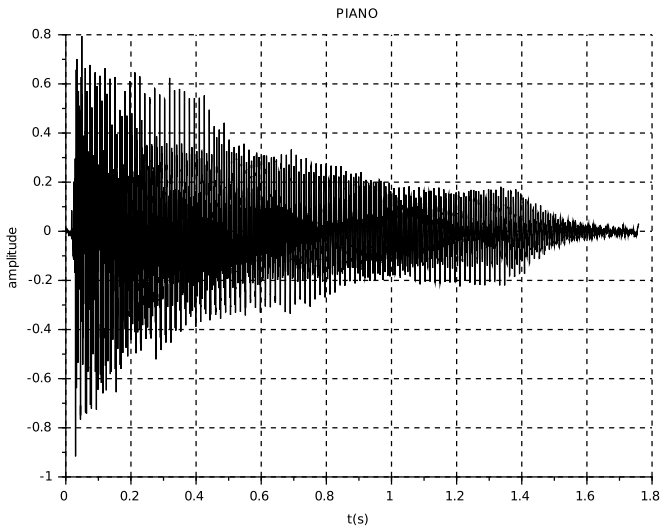
$$s = \{sn = s(nT_e), n = 0..N-1\}$$

$$S(f) = TFD(s) = X(f) = \sum_{n=0}^{N-1} s(nT_e)\, e^{-2i\pi f \frac{n}{f_e}}$$

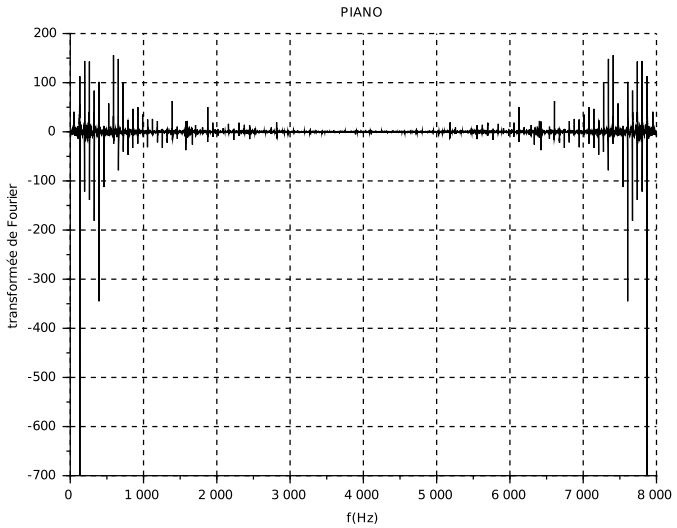
$$S = fft(s) = \{S_k = S(\frac{kf_e}{N}), k = 0..N-1\}$$

- autant d'échantillons dans s que S
- échantillons de S aux fréquences $f_k = kf_e/N, k = 0..N-1$
- résolution fréquentielle : $\Delta_f = f_{k+1} - f_k = \frac{f_e}{N}$
- $S(f)$ complexe. Module : $S = abs(fft(s))$

Spectre d'un son :PIANO.wav




```
s,fe=librosa.load(``PIANO.wav'')
t=[0:len(s)-1]/fe
plt.plot(t,s)
```




```
S=np.abs(np.fft.rfft(s))
tf=np.fft.rfftfreq(len(s),1/fe)
# ou tf=[0:len(s)-1]*fe/len(s)
plt.plot(tf,S)
```

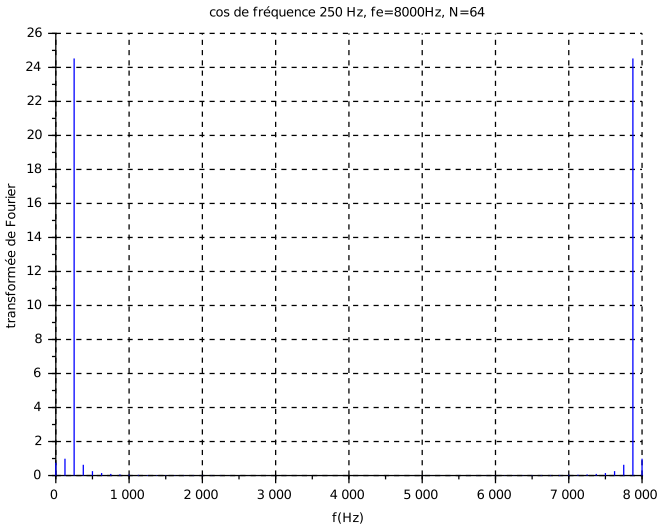
Spectre d'un signal sinusoïdal

$$s = \{sn = s(nT_e) = a\,\cos(2\pi f_0 n T_e), n = 0..N-1\}$$

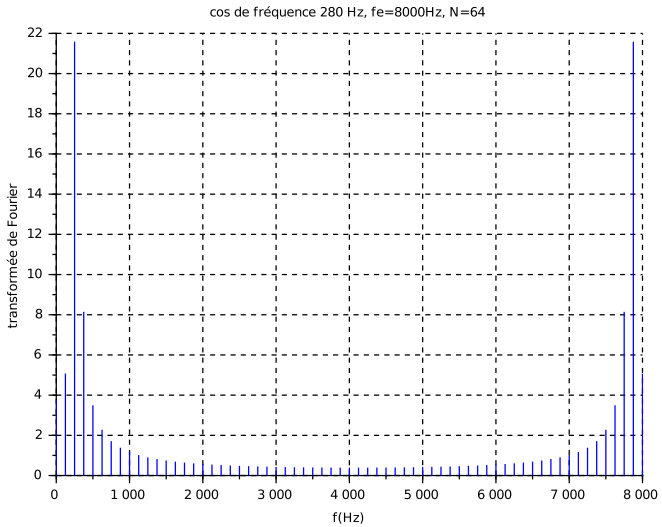
$$\begin{aligned}
S(f) &= TFD(s) = X(f) = \sum_{n=0}^{N-1} s(nT_e)\, e^{-2i\pi f \frac{n}{f_e}} \\
&= \sum_{n=0}^{N-1} a\,\cos(2\pi f_0 n T_e)\, e^{-2i\pi f \frac{n}{f_e}} \\
&= \sum_{n=0}^{N-1} \frac{a}{2}\left(e^{i2\pi f_0 n T_e} + e^{-i2\pi f_0 n T_e}\right) e^{-2i\pi f \frac{n}{f_e}} \\
&= \frac{a}{2}\sum_{n=0}^{N-1} e^{i2\pi(f_0-f)nT_e} + \frac{a}{2}\sum_{n=0}^{N-1} e^{i2\pi(-f_0-f)nT_e} \\
&= \frac{a}{2} e^{-i(N-1)\pi(f-f_0)T_e}\,\frac{\sin(N\pi(f-f_0)T_e)}{\sin(\pi(f-f_0)T_e)} \\
&\quad + \frac{a}{2} e^{-i(N-1)\pi(f+f_0)T_e}\,\frac{\sin(N\pi(f+f_0)T_e)}{\sin(\pi(f+f_0)T_e)}
\end{aligned}$$

échantillonné tous les $kf_e/N, k = 0..N-1$

Spectre d'un signal sinusoïdal : effet de f_e

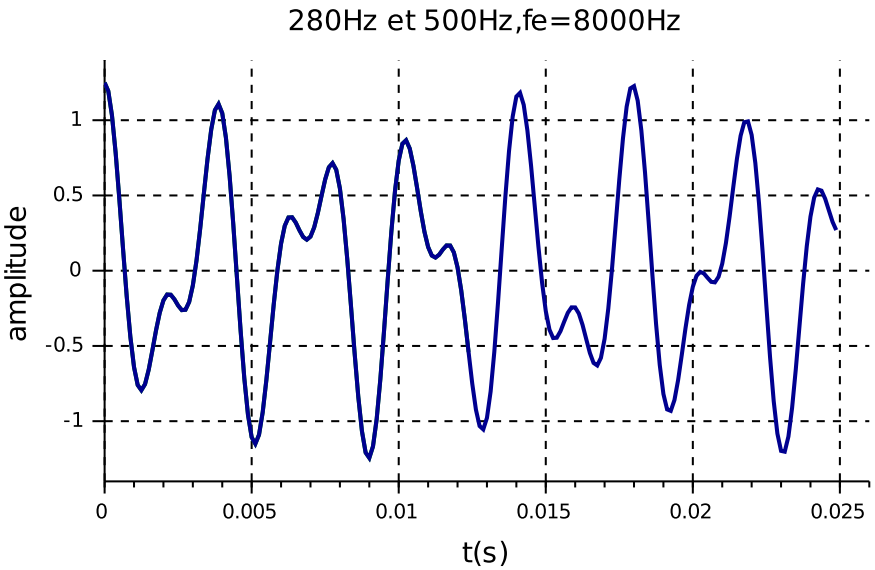


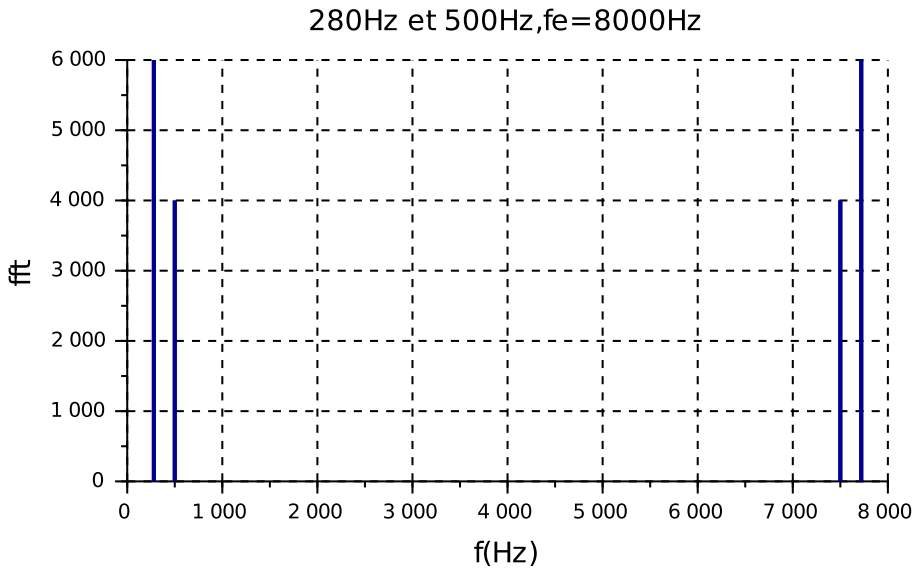

```
t=[0:1:64]/8000
s = 0.75*cos(2*%pi*250*t)
tf=[0:1:64]*8000/64
plot2d(tf,abs(fft(s)))
xgrid
xtitle('cos 250Hz,fe=8000Hz,N=64',
       'f(Hz)','fft')
```




```
t=[0:1:63]/8000
s = 0.75*cos(2*%pi*280*t)
tf=[0:1:63]*8000/64
plot2d3(tf,abs(fft(s)))
xgrid
xtitle('cos 280Hz,fe=8000Hz,N=64',
       'f(Hz)','fft')
```

Spectre d'un signal composé de 2 sinusoides






```
N=16000
fe=8000
t=[0:N-1]/fe;
s = 0.75*cos(2*%pi*280*t) + 0.5*cos(2*%pi*500*t);
tf=[0:N-1]*fe/N;
subplot(1,2,1);
plot2d(t(1:200),s(1:200));
xgrid
xtitle('280Hz et 500Hz,fe=8000Hz','t(s)','amplitude')
subplot(1,2,2);
plot2d3(tf,abs(fft(s)));
xgrid
xtitle('280Hz et 500Hz,fe=8000Hz','f(Hz)','fft')
```

Energie d'un son

Définition de l'énergie associée à un signal à partir des échantillons de ce signal :

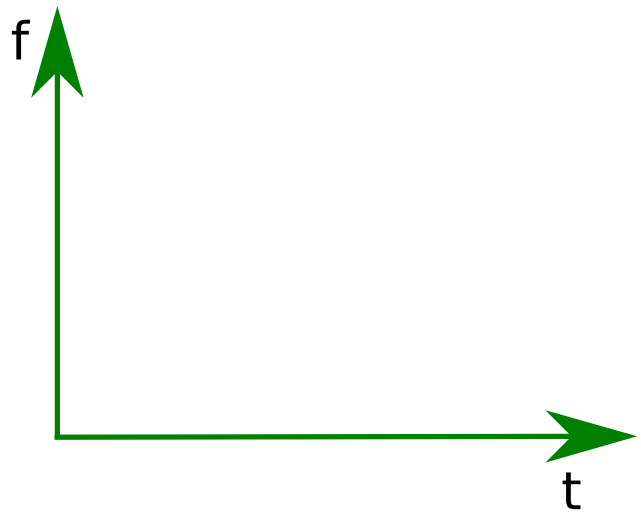
$$E(x) = \frac{1}{2} \sum_{n=0}^{N-1} x_n^2$$

Calcul de l'énergie à partir du spectre (théorème de Parseval) :

$$E(x) = \frac{1}{2N} \sum_{m=0}^{N-1} |x_m|^2$$

Spectrogramme ou sonagramme ou encore sonogramme

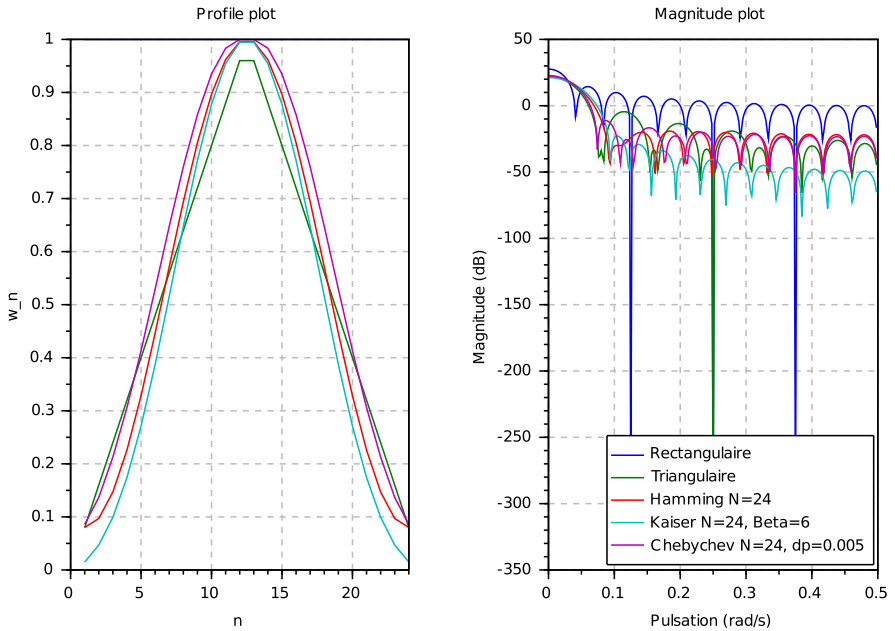
- Evolution du spectre au cours du temps.
- Découpage du temps en intervalles (ou *chunks*).
  - possibilité de recouvrement
  - formes de fenêtres (ou *windowing*)
- Intensité linéaire ou logarithmique



Fenêtrage

- découpage temporel : $h(t)$
- signal étudié : $s(t).h(t)$
- transformée de Fourier : $S(f) * H(f)$ (produit de convolution)

Types de fenêtrage



- rectangulaire
- triangulaire
- fenêtre de Hann
- fenêtre de Hamming
- Kaiser
- Chebychev

Calcul du spectrogramme en Scilab

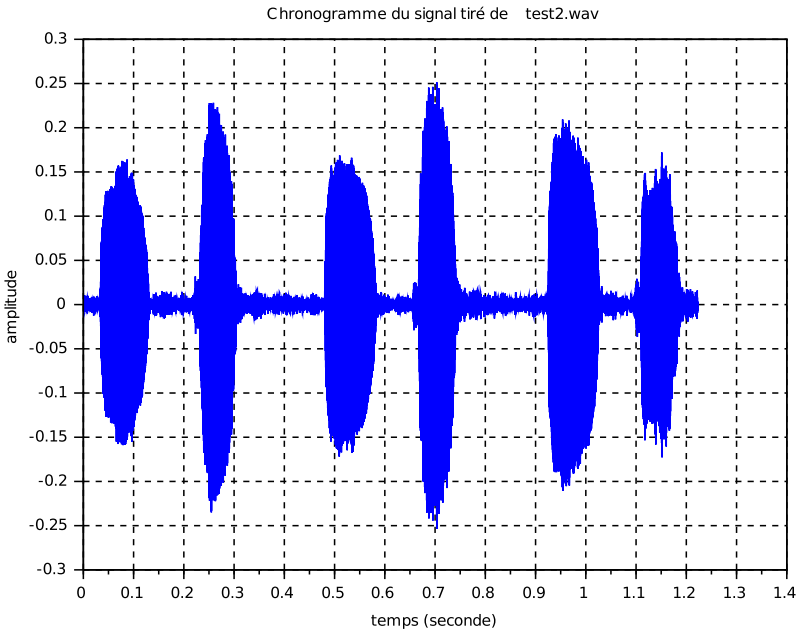
```
file='mesange.wav';
[y,fe,B]=wavread(file); // charger fichier wave
plot([0:length(y)-1]/fe,y) // dessin du chronogramme
xgrid
xtitle(["Chronogramme du signal ",file])
xlabel('temps (seconde)')
ylabel('amplitude')
playsnd(y,fe) // joue le son
D=1024; //taille de la fenêtre
hamming=window('hm',D);// fenêtre de Hamming

// calcul du spectre
spectre=[];
N=fix(length(y)/D)-1; //nbre de fenêtres
for k= 1:N
    debut=1+(k-1)*D;
    spe=1+abs(fft(y(debut:debut+D-1).*hamming));
    spectre=[spectre;20*log10(spe(1:D/2)/D)];
end
fr=[0:D/2-1]*fe/D;
t=(0:N-1)*D/fe;

xset("colormap", hotcolormap(128)); //graycolormap(128));
colorbar(min(spectre),0)
grayplot(t',fr',spectre)
xtitle(["Spectrogramme, fenêtre de taille: ",string(D)])
xlabel('temps (s)')
ylabel('fréquence (Hz)')
```

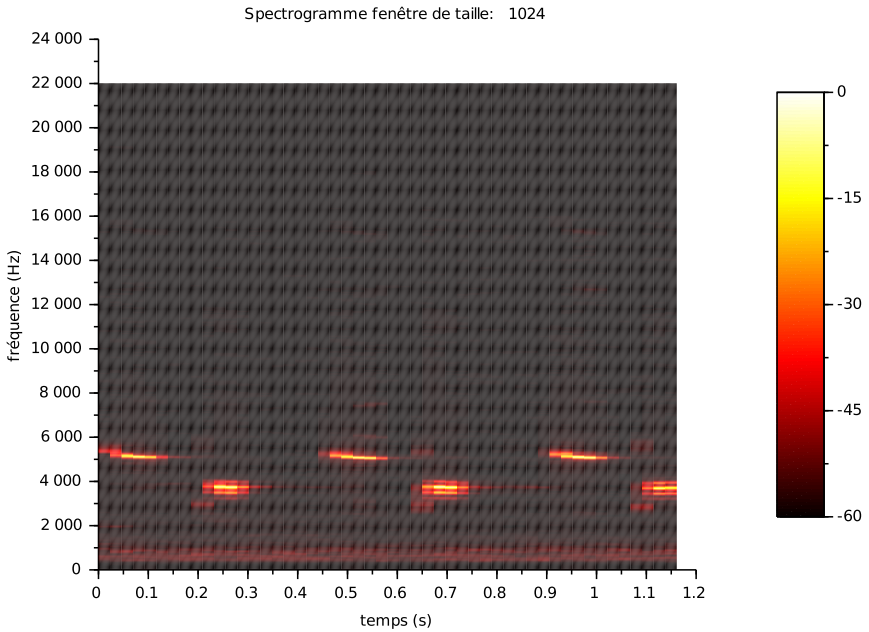
Calcul du spectrogramme en python

```
S = np.fft.rfft(s)
Sdb = librosa.amplitude_to_db(abs(S))
librosa.display.specshow(Sdb, sr=fe, x_axis='time', y_axis='hz')
```

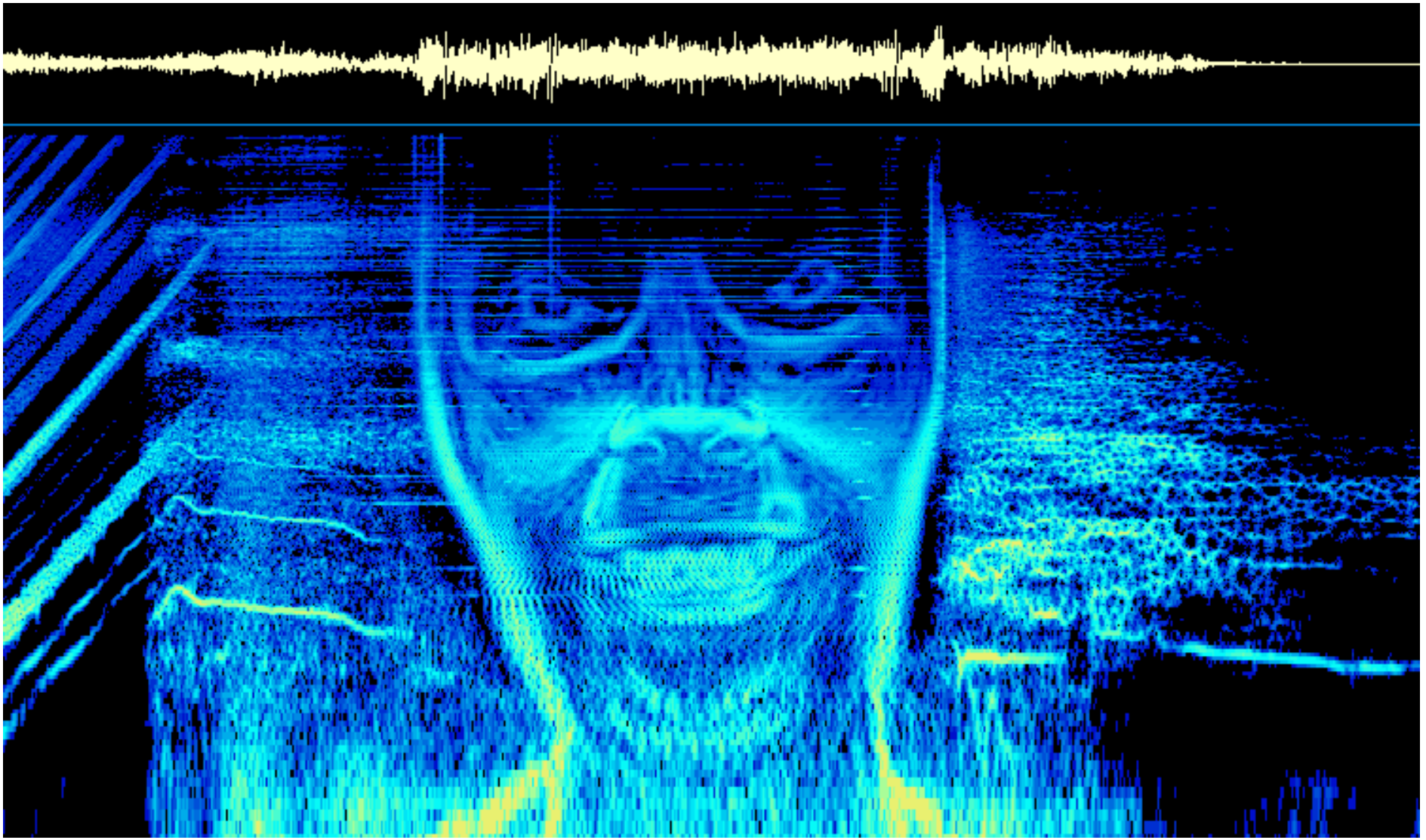
La mésange

[ext]



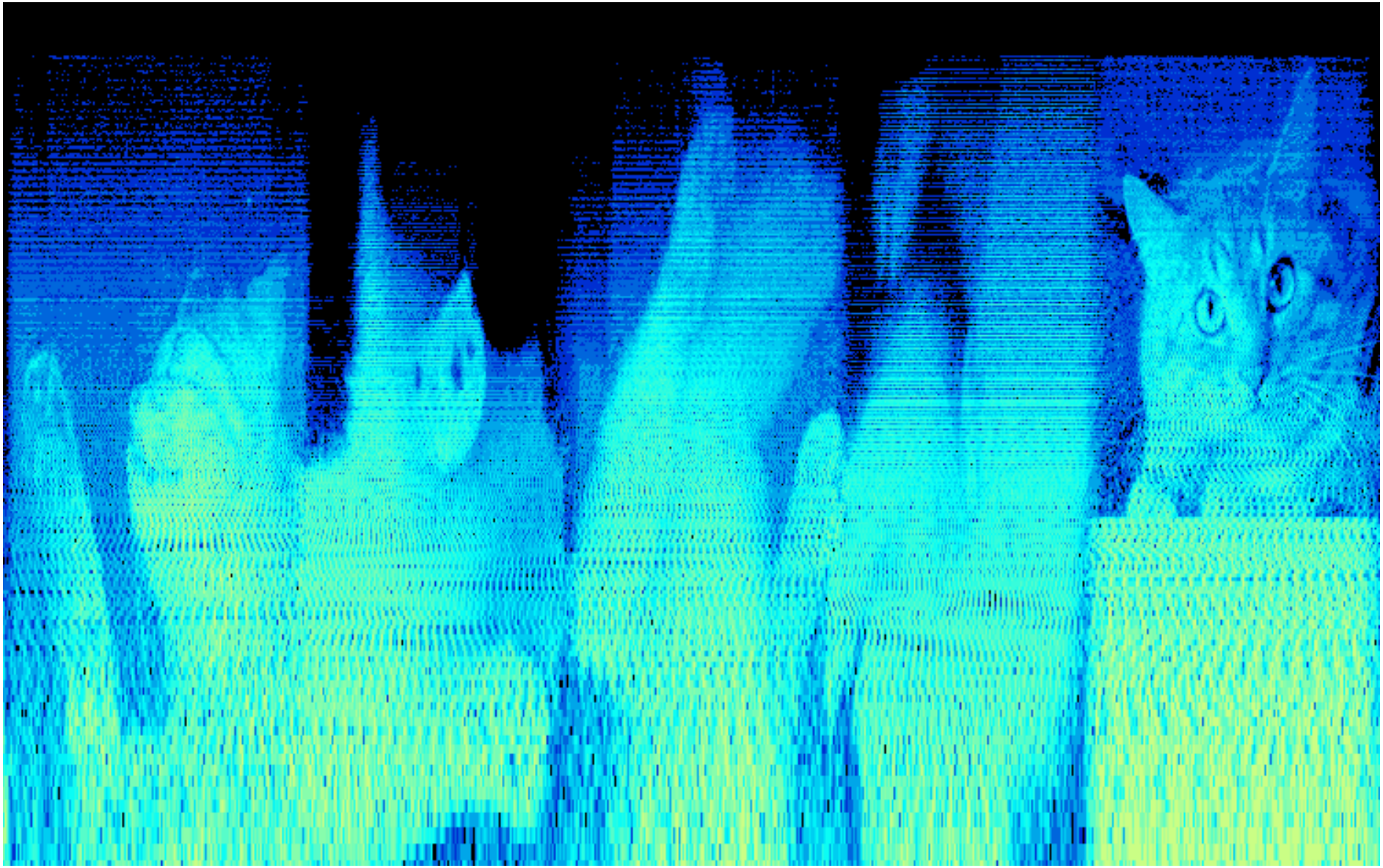


Stéganographie (1)



http://www.bastwood.com/?page_id=10

Stéganographie (2)



[ex

http://www.bastwood.com/?page_id=10

Transformée de Fourier inverse

- transformée de Fourier :

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-2i\pi ft} dt$$

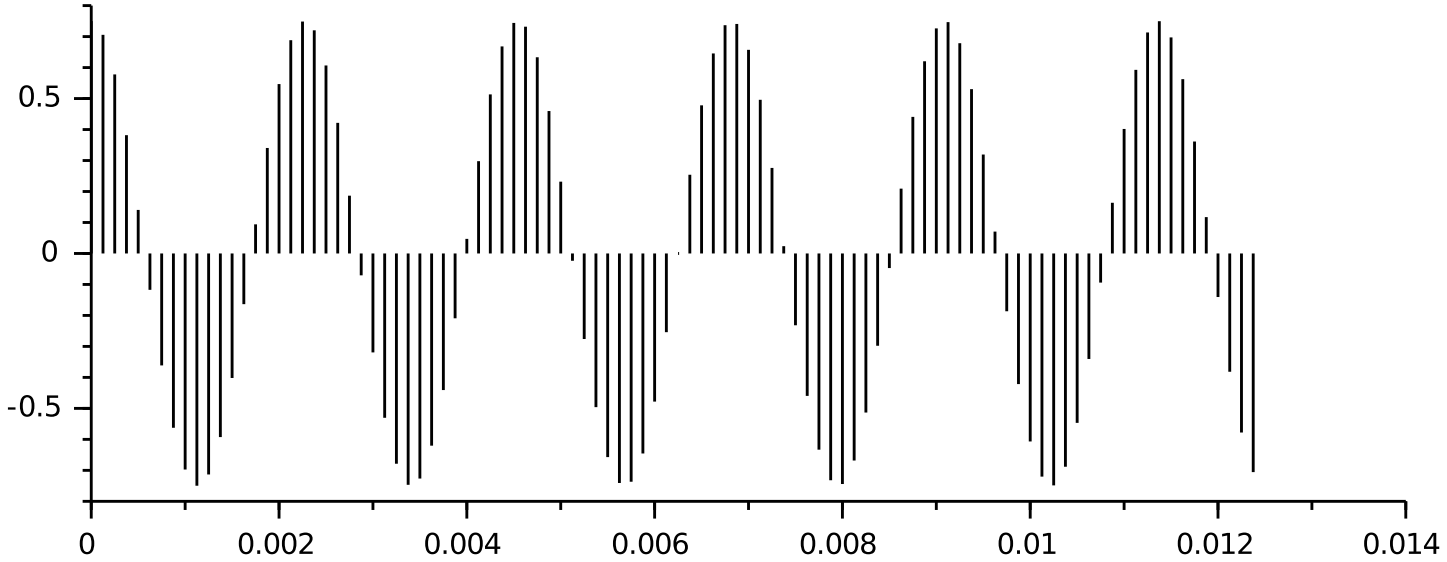
- transformée de Fourier inverse :

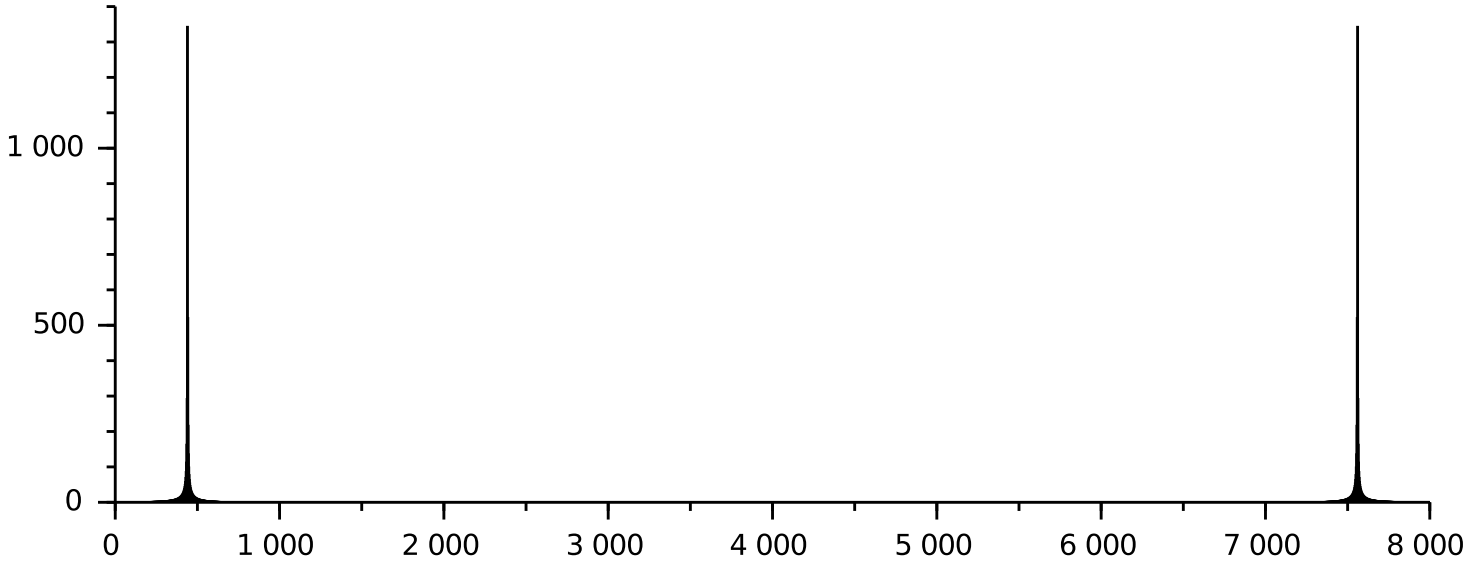
$$s(t) = \int_{-\infty}^{+\infty} S(f) e^{2i\pi ft} df$$

- transformée de Fourier inverse en Scilab : `ifft`

Outline

Sous échantillonnage : un exemple



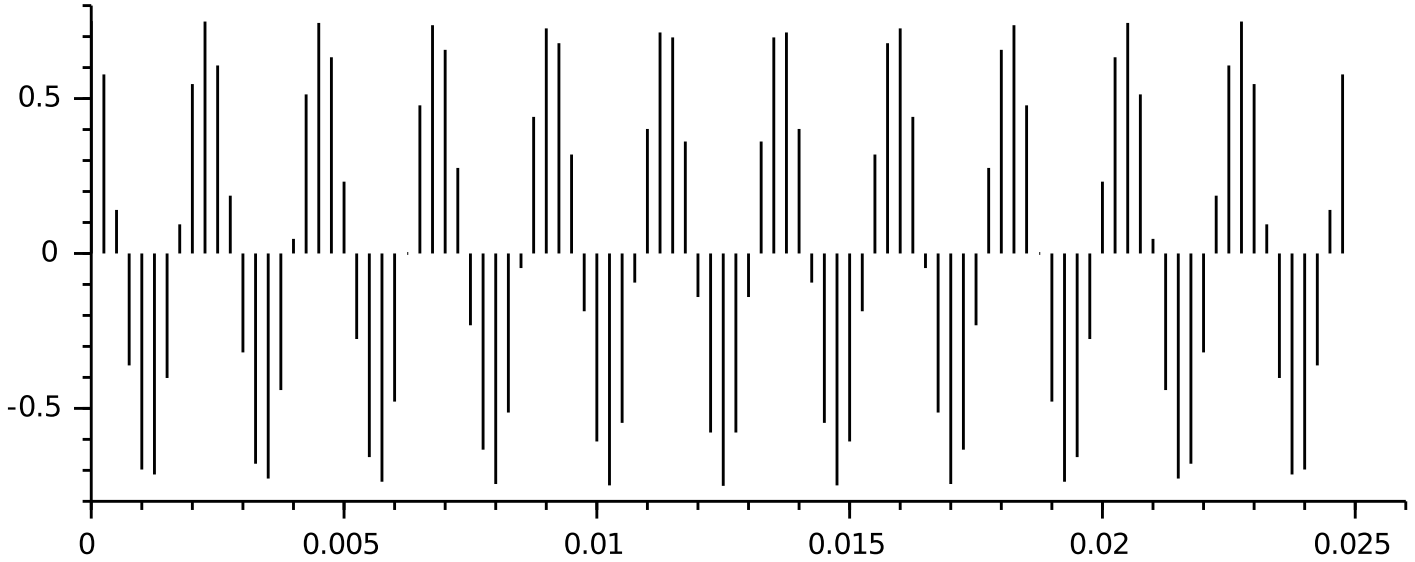


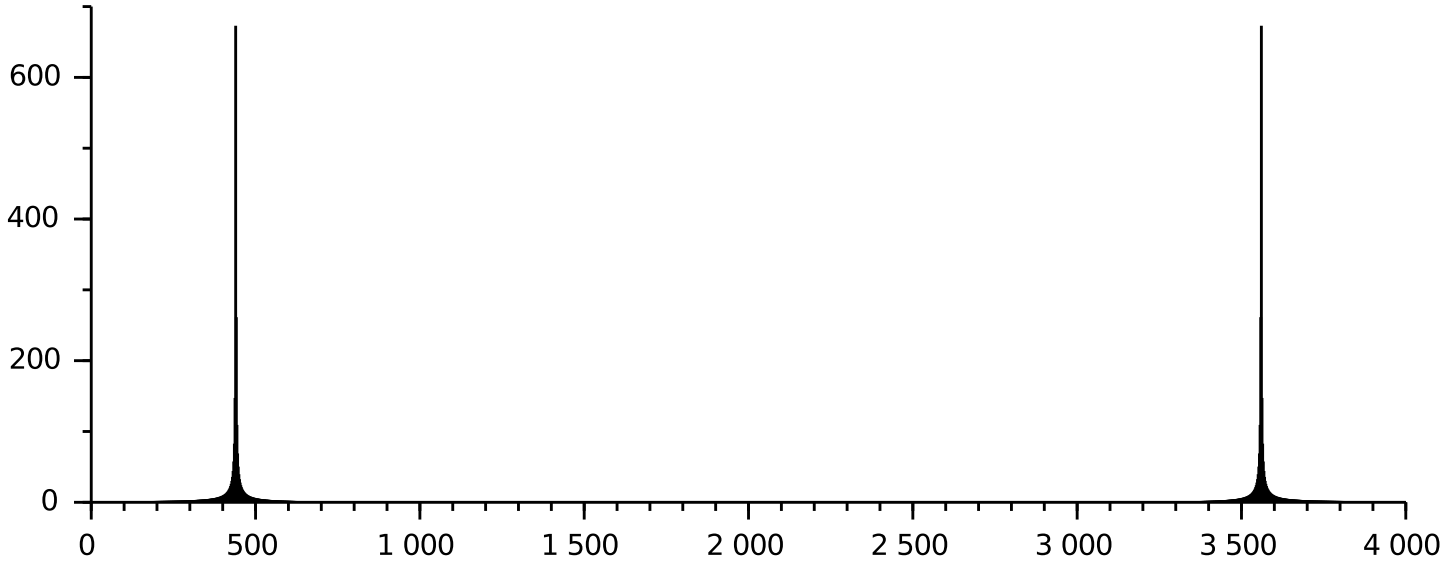
[ext]

Sous-échantillonnage : le code

```
N=4096
fe=8000
t=[0:N-1]/fe
s=0.75*cos(2*%pi*440*t)
clf()
subplot(411)
plot2d3(t(1:100),s(1:100))
subplot(412)
plot2d3(t*fe*fe/N,abs(fft(s)))
t2=t(1:2:length(t)-1)
s2=s(1:2:length(t)-1)
fe=fe/2
N=N/2
subplot(413)
plot2d3(t2(1:100),s2(1:100))
subplot(414)
plot2d3(t2*fe*fe/N,abs(fft(s2)))
```

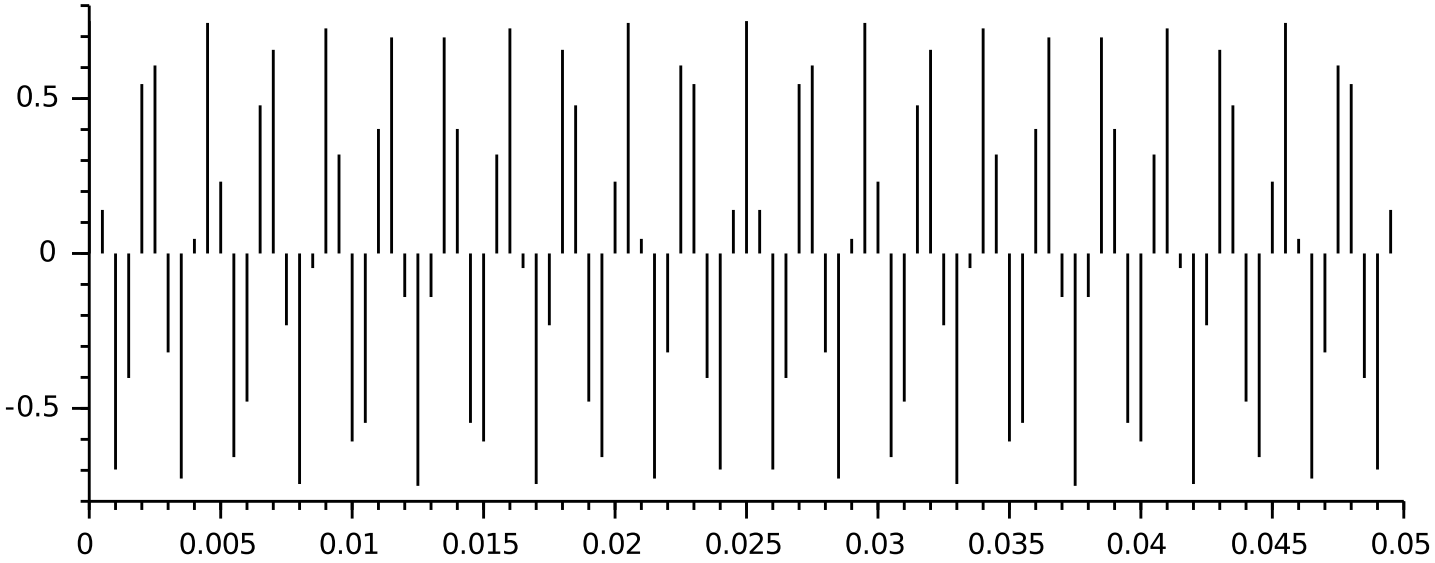
Sous-échantillonnage : par 2

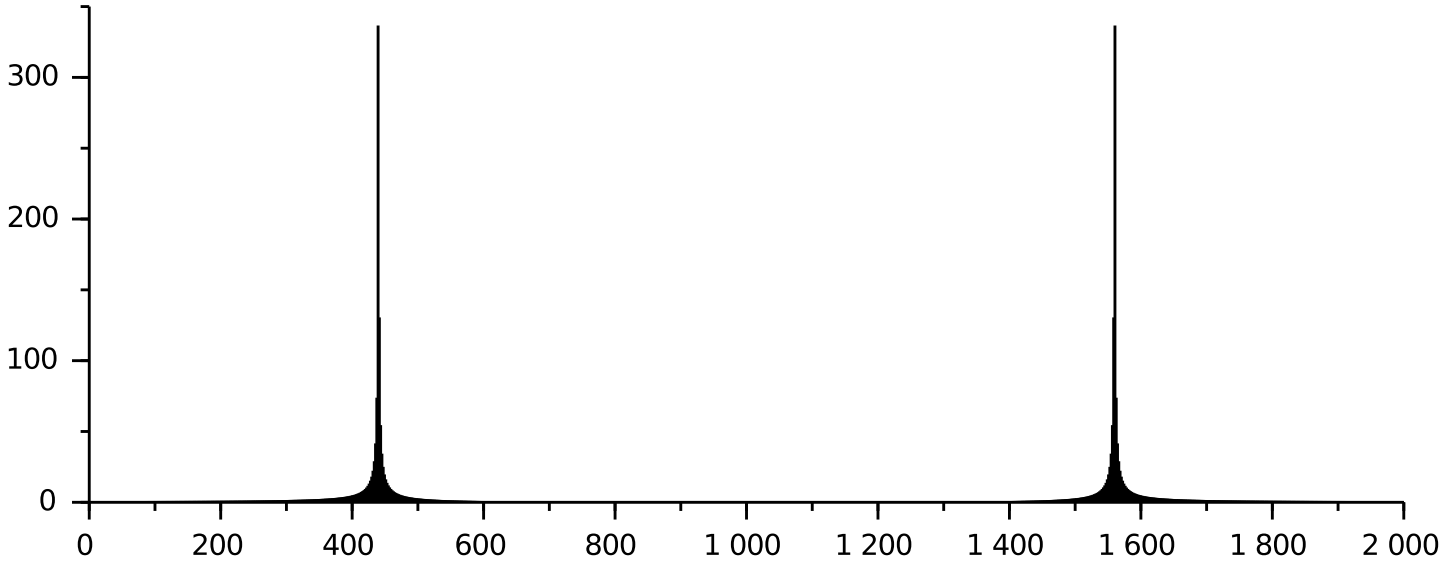




[ext]

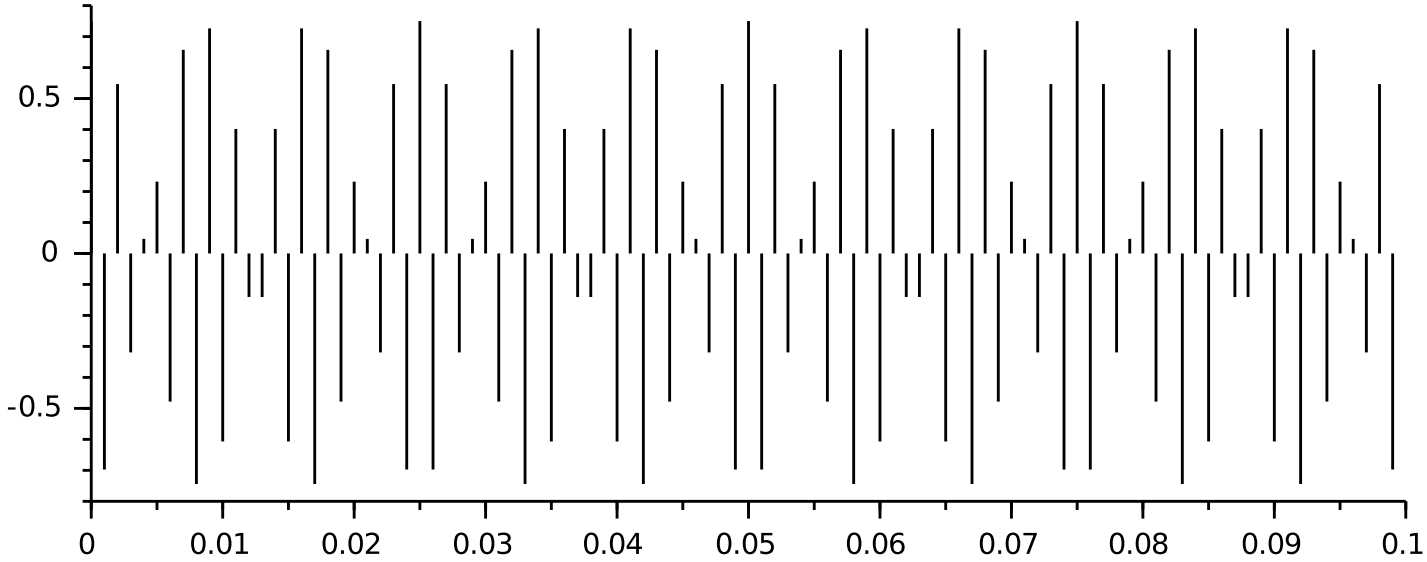
Sous-échantillonnage : par 4

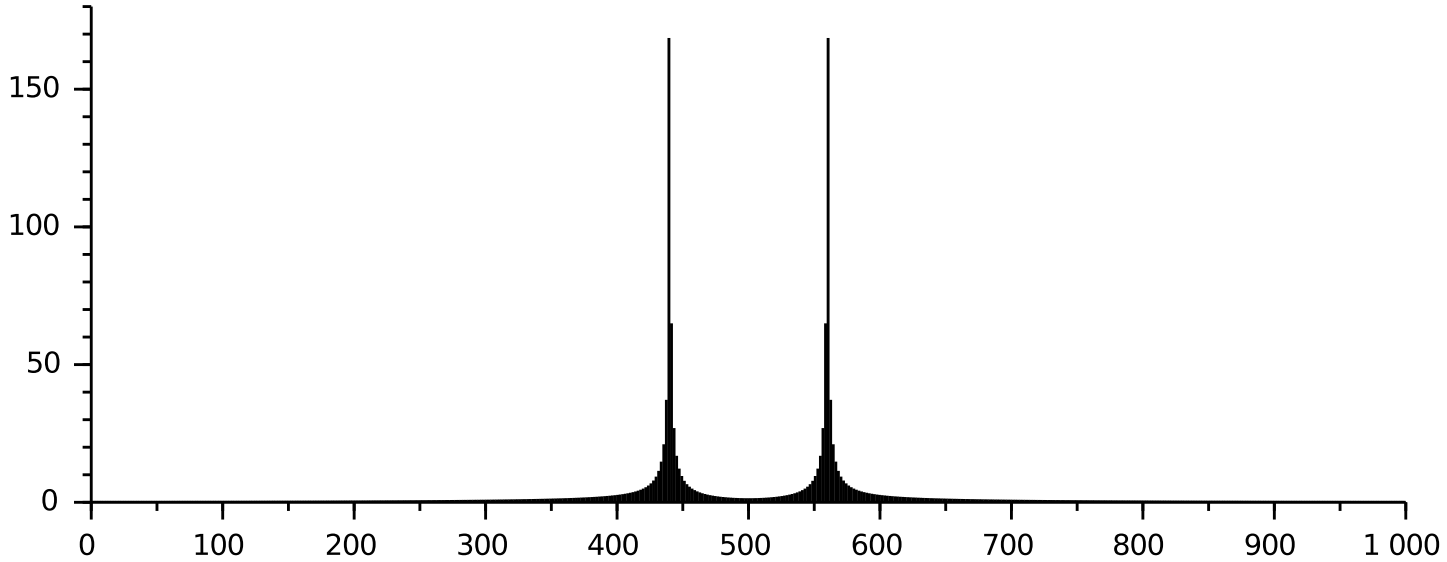




[ext]

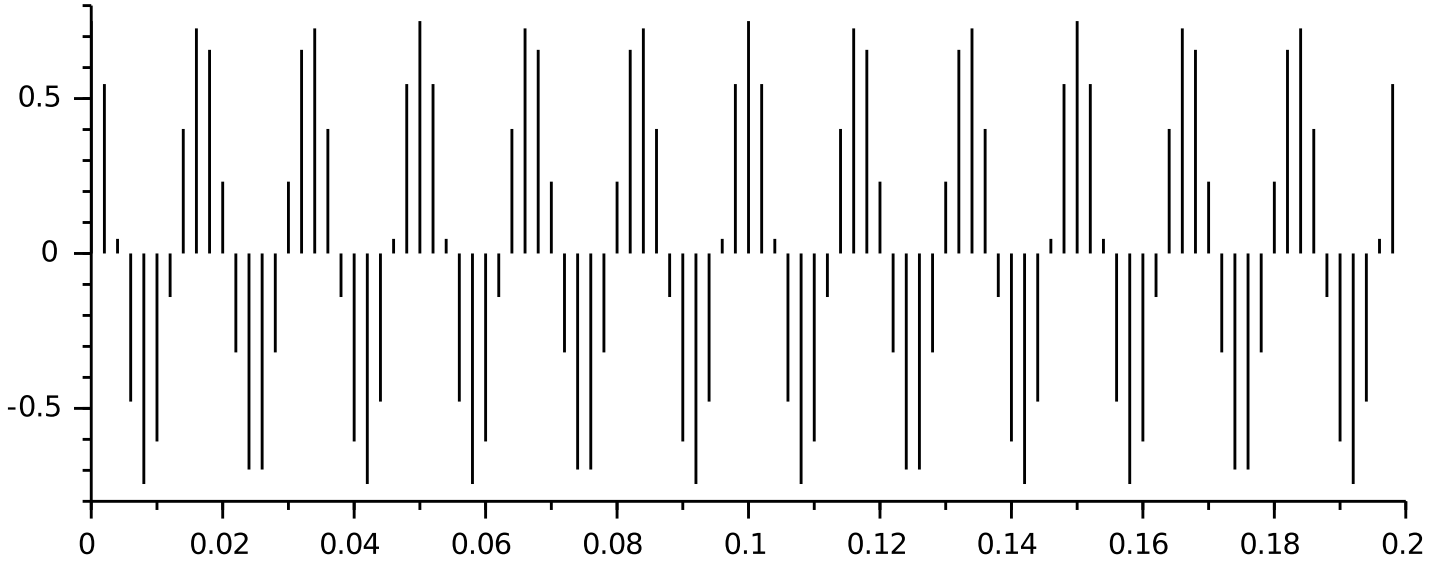
Sous-échantillonnage : par 8

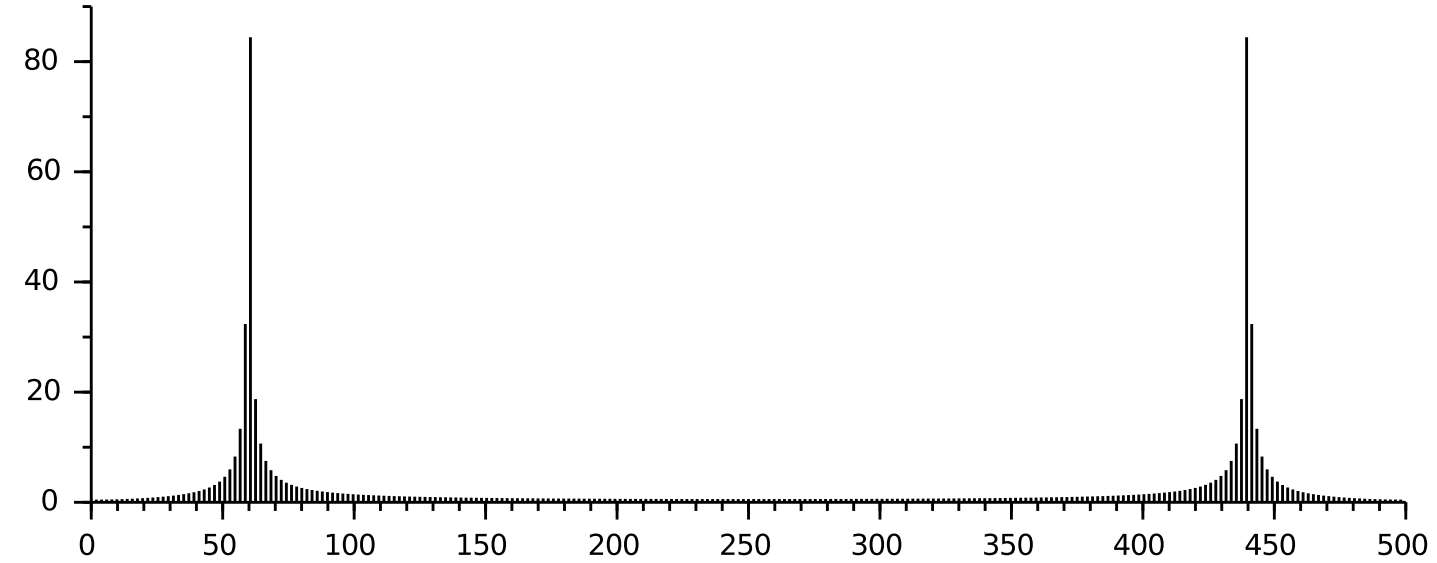




[ext]

Sous-échantillonnage : par 16



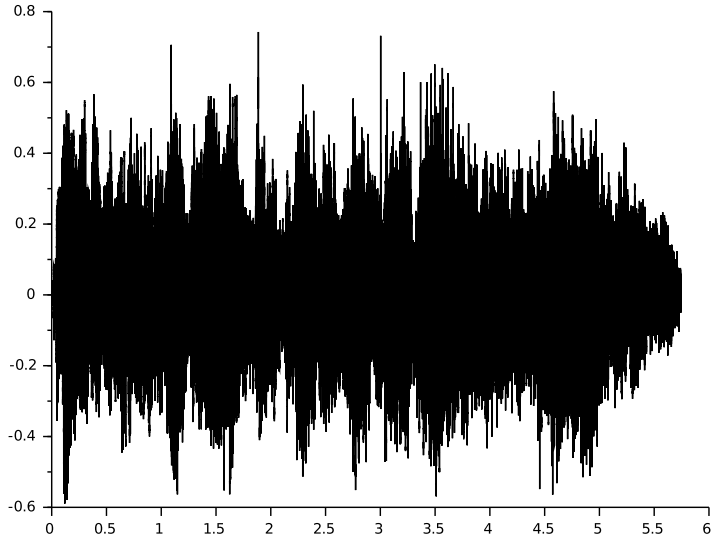


[ext]

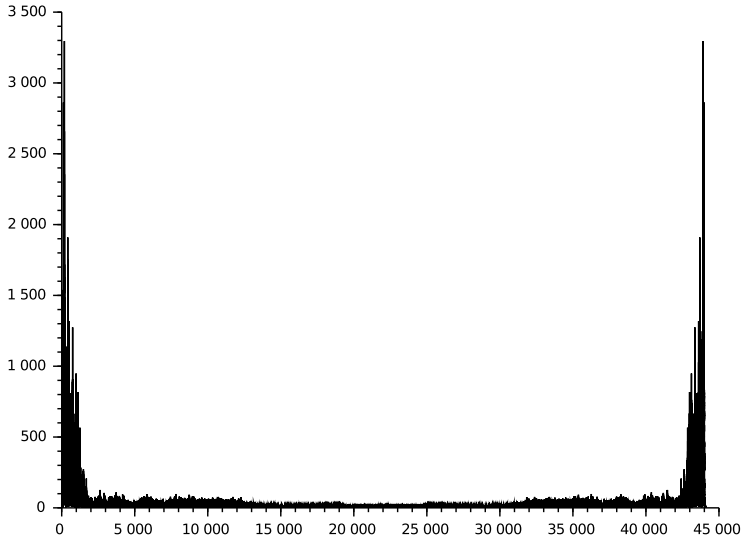
Théorème de Nyquist - Shannon

La fréquence d'échantillonnage doit être au moins deux fois supérieure à la fréquence maximale du signal.

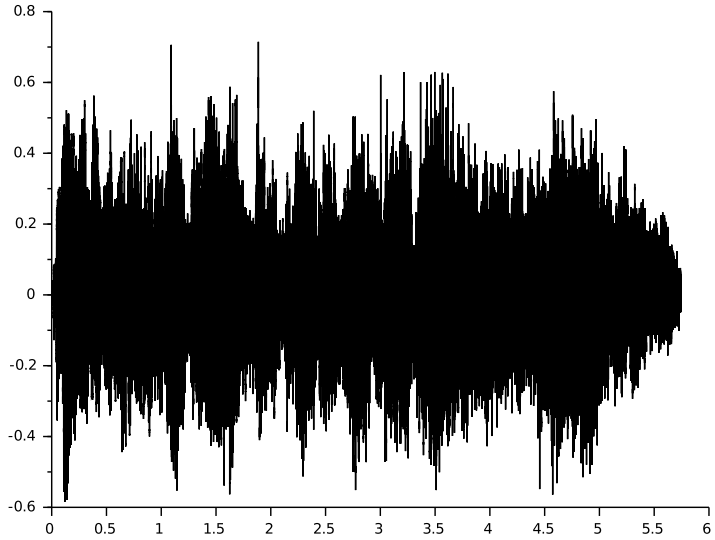
Exemple de dégradation par sous-échantillonnage



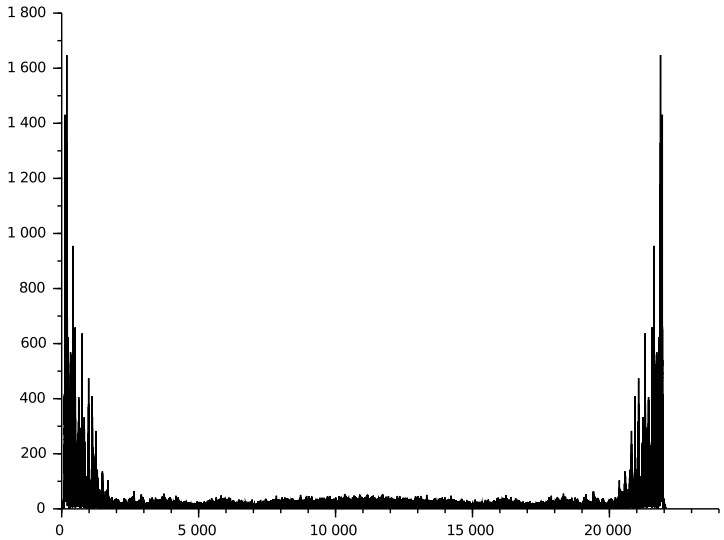
[ext]



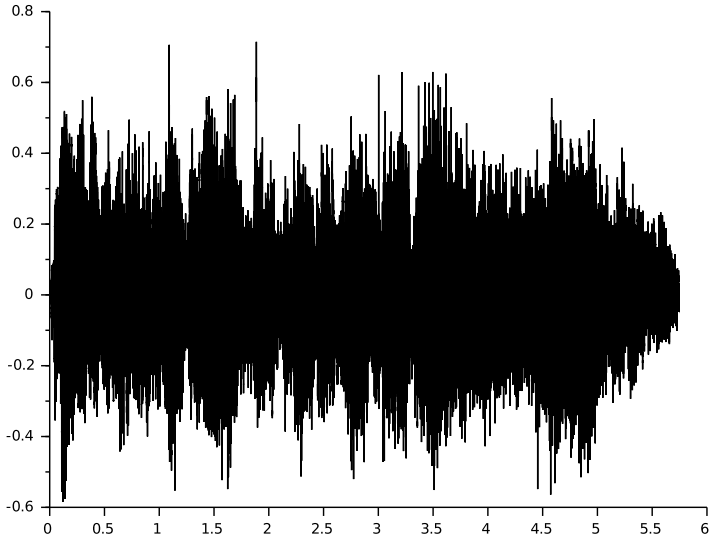
Exemple de dégradation par sous-échantillonnage : 2



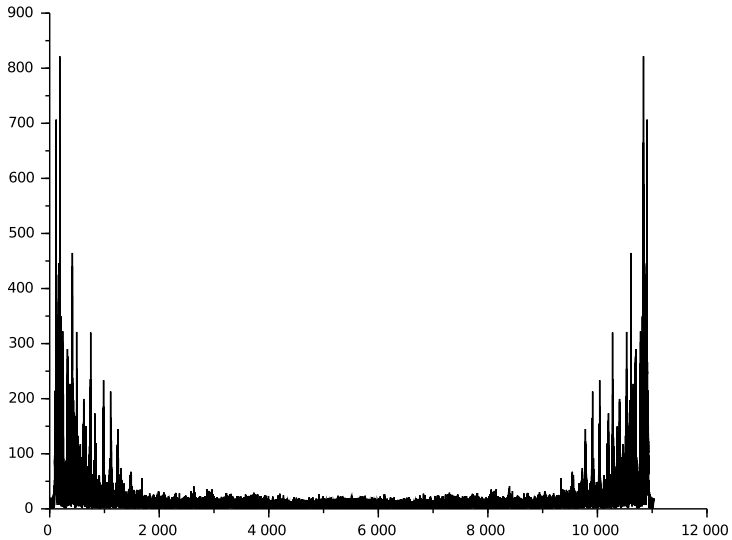
[ext]



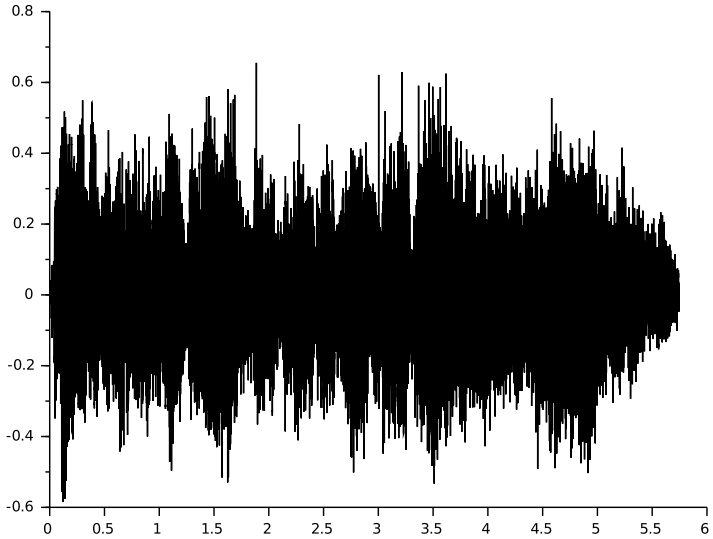
Exemple de dégradation par sous-échantillonnage : 4

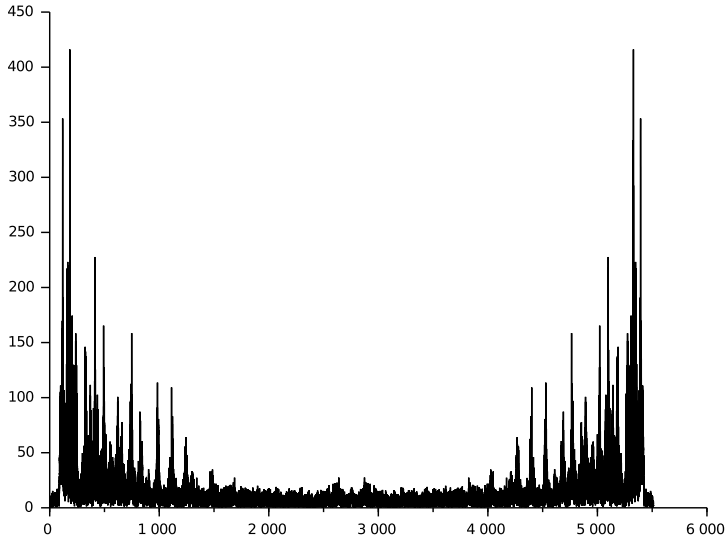


[ext]



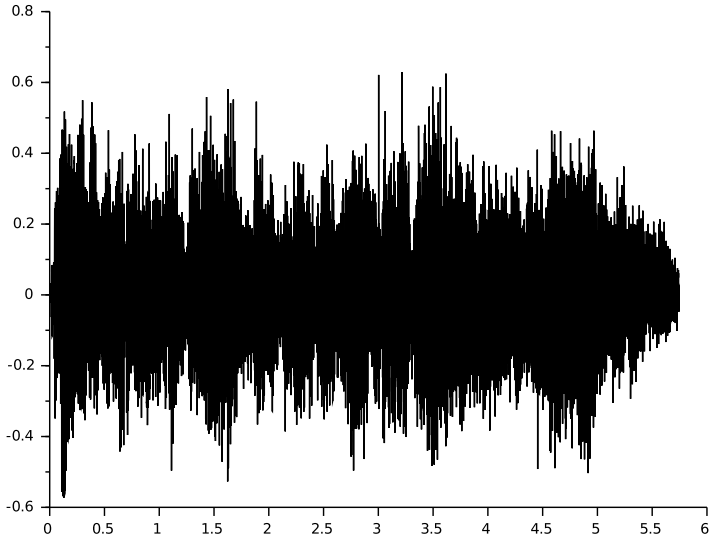
Exemple de dégradation par sous-échantillonnage : 8

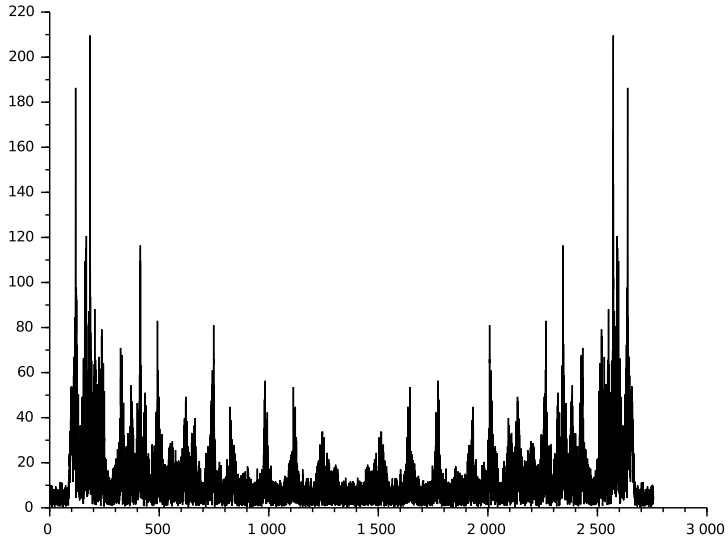




[ext]

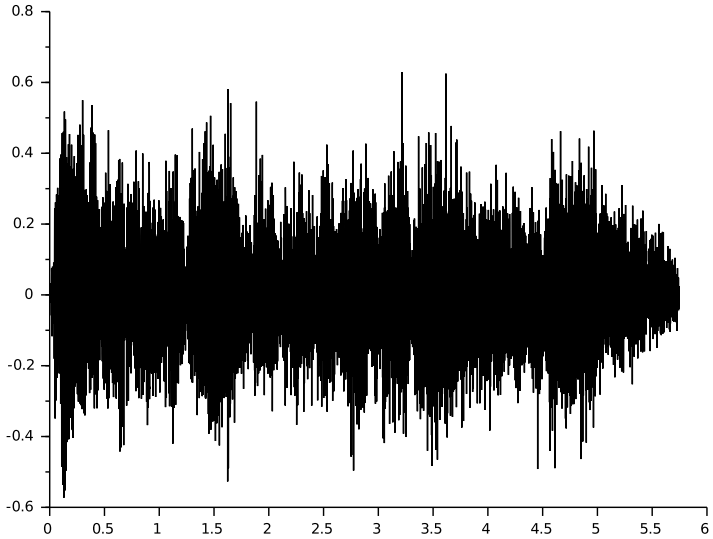
Exemple de dégradation par sous-échantillonnage : 16

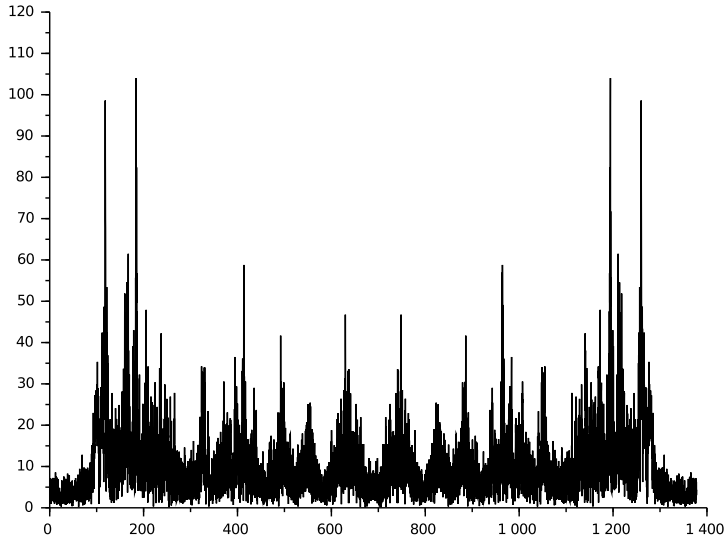




[ext]

Exemple de dégradation par sous-échantillonnage : 32





[ext]

en python

- FFT :
  - `numpy.fft.rfft` *Compute the one-dimensional discrete Fourier Transform for real input.*
  - `numpy.fft.fftfreq(n, d=1.0)` *Return the Discrete Fourier Transform sample frequencies.*
- spectrogram : `librosa.display.specshow`