Densité spectrale de puissance

Version 2020

Lumi - P. Cladé

oct. 19, 2020

L'objectif de ce TD est double : 1/ à l'aide de deux exemples mieux comprendre la notion de densité spectrale de puissance 2/ savoir utiliser l'informatique pour faire du traitement du signal (filtre, transformée de Fourier).

Note : Pour ce TD, nous vous conseillons d'utiliser un notebook jupyter. Nous tracerons des graphiques et utiliserons numpy, le notebook commencera donc par les lignes suivantes

```
import matplotlib.pyplot as plt
import numpy as np
```

Le sujet de TD se trouve sur la page `https://github.com/clade/Lumi2019`, dans le repertoire `densite_spectrale_de_puissane` On y trouvera aussi un template sous forme de notebook (fichier `dsp.ipynb`).

1 Avant propos

Lorsque l'on digitalise un signal, on parle de taux d'échantillonnage ($1/\Delta t$ où Δt est le délai entre deux mesures), le nombre N de points, la durée T de la mesure (on a $T = N\Delta t$). On note t_i l'instant de la mesure i (on a $t_i = t_0 + i\Delta t$). Pour un signal $x(t)$, on note $x_i = x(t_i)$.

Un point important lors du traitement numérique de signaux et de convertir les formules continues en formules discrètes.

1.1 Transformée de Fourier continue

Voici quelques rappels :

— Transformée de Fourier continue

$$\tilde{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-2i\pi ft} \mathrm{d}t$$

— Transformée de Fourier inverse

$$x(t) = \int_{-\infty}^{\infty} \tilde{x}(f) e^{2i\pi ft} \mathrm{d}f$$

1.2 Transformée de Fourier Discrète

On peut transformer l'intégrale par une somme de Riemann :

$$\tilde{x}(f) = \sum_{j=0}^{N-1} x_j e^{-2i\pi f j \Delta t} \Delta t$$

Dans la pratique, cette somme n'est calculée que pour des fréquence $f_k = \frac{k}{T}$. On obtient alors ce que l'on appelle la transformée de Fourier discrète dans laquelle il n'y aucun facteur dimensionné :

$$\tilde{x}_k = \sum_{j=0}^{N-1} x_j e^{-2i\pi \frac{kj}{N}}$$

Avec la relation suivante :

$$\tilde{x}(f_k) = \tilde{x}_k \Delta t$$

La transformée de Fourier discrète (Discret Fourier Transform - DFT) est souvent calculée en utilisant l'algorithme de FFT (Fast Fourier Transform). L'usage confond les deux acronymes et on utilise FFT pour désigné la DFT.

La DFT possède un inverse qui est calculé par :

$$x_j = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{x}_k e^{2i\pi \frac{kj}{N}}$$

1.3 DFT en Python

On utilisera les fonctions `fft` et sa réciproque `ifft` du module `numpy.fft`.

2 Densité spectrale du bruit d'une machine à laver

Nous avons enregistré le bruit d'une machine à laver le linge pendant la phase d'essorage. Il s'agit du fichier `machine_a_laver.wav`. On entend un son aigu correspondant au moteur. Le tambour tourne beaucoup plus lentement.

1. Quel est la fréquence typique de rotation du tambour d'une machine à laver le linge ?
2. On va lire les fichiers `wav` en utilisant la fonction `read` de la librairie *scipy.io.wavfile*. On enregistrera l'amplitude du son dans la variable *amplitude*

```
from scipy.io.wavfile import read
samplerate, amplitude = read('data/machine_a_laver.wav')
```

   Quel est le nombre N de points ? Que vaut Δt ? Quelle est la durée T de l'enregistrement ? Tracer l'amplitude du son pendant la dernière seconde.
3. On va utiliser la fonction `periodogram` du module `scipy.signal`. Le périodogramme est le nom de la méthode utilisée pour estimer la densité spectrale de puissance en se basant sur le transformée de Fourier (équation 1.22 du cours). Tracer la densité spectrale de puissance entre 10 et 50 Hz. On utilisera une échelle logarithme (fonction `plt.loglog`)

```
from scipy.signal import periodogram
freq, psd = periodogram(amplitude, samplerate, )
```

L'amplitude du son $x(t)$ peut s'écrire comme la somme d'un bruit de fond $b(t)$ et du signal lié à la rotation du tambour $s(t)$. Identifier sur le graph ce qui correspond au bruit de fond et ce qui correspond au signal. Quelle est la fréquence de rotation du tambour de la machine à laver.

4. On suppose que les unités de `amplitude` sont des volts (V). Quelle est l'unité de la densité spectrale de bruit ? Quelle est la valeur typique de la densité spectrale de bruit de $b(t)$ entre 10 et 100 Hz ?
5. La puissance moyenne de $x(t)$ est donnée par $\langle x^2(t)\rangle = \frac{1}{T}\int_0^T x^2(t)\mathrm{d}t$. Calculer numériquement cette valeur. Vérifier que la puissance moyenne est aussi donnée par $\int S_x(f)\mathrm{d}f$.
6. Le bruit $b(t)$ et le signal $s(t)$ ne sont pas corrélés. Nous avons donc la relation suivante : $S_x(f) = S_b(f) + S_s(f)$. En séparant S_x en deux parties (fin de la question 3), estimer la puissance moyenne de $b(t)$ et de $s(t)$. Quel est le rapport signal à bruit ?
7. Aproximativement, sur quelle bande passante faut-il filtrer le son pour avoir un rapport signal à bruit supérieur à 1 ?

3 Filtre en Python

Les algorithmes de FFT se trouvent dans le module `fft` de numpy. Il est possible de simplifier le calcul de la transformée de Fourier lorsque le signal est réél (puisque $\tilde{x}(-f) = \tilde{x}(f)^*$). On utilise alors les fonctions `rfft` et `irfft` pour effectuer la FFT. La fonction `rfftfreq` va donner les fréquences.

On crée le signal suivant

```
samplerate = 44100
signal = zeros(samplerate*3)
signal[samplerate:samplerate*2)] = 1
plot(signal)
```

1. Vérifier que l'on obtient le même signal après avoir appliqué `rfft` et `irfft`

```
import numpy as np
signal_tilde = np.fft.rfft(signal)
signal_2 = np.fft.irfft(signal_tilde)
plt.plot(signal_2)
```

2. Utiliser la fonction `rfftfreq`. Est-ce que ce qu'elle renvoie vous semble cohérent ?
3. On veut appliquer un filtre passe bas de fréquence de coupure $f_c = 3$ Hz

$$H(f) = \frac{1}{1 + j(f/f_c)}$$

Pour cela on définit la fonction python suivante

```
def pass_bas(signal, f_c, samplerate=44100):
    signal_tilde = np.fft.rfft(signal)
    freqs = np.fft.rfftfreq(len(signal), 1/samplerate)
    H = 1/(1+1J*(freqs/f_c))
    signal_2 = np.fft.irfft(H*signal_tilde)
    return signal_2
```

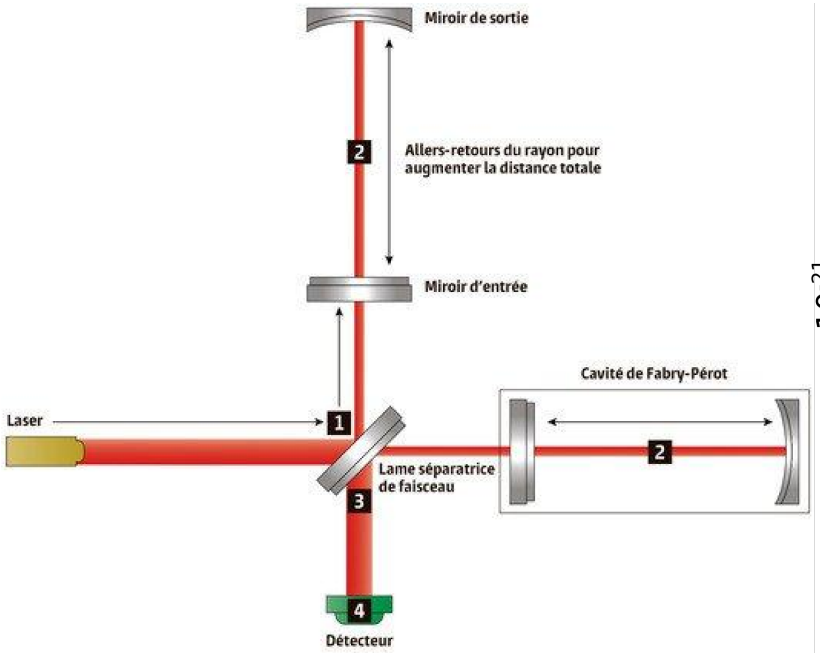
Tracer le signal filtré.

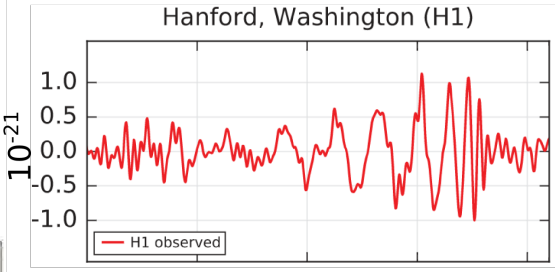
4. Faire de même avec un filtre passe haut en définissant la fonction `passe_haut` :

$$H(f) = \frac{j(f/f_c)}{1 + j(f/f_c)}$$

5. Filtrer le son de la machine à laver avec un filtre passe bas de fréquence 30 Hz. Peut-on voir le signal du tambour ? Quel est son amplitude RMS ?

4 Onde gravitationnelle





L'objectif de cette partie est d'analyser les signaux provenant des deux détecteurs d'onde gravitationnelle de LIGO (Hanforg et Linvingston). Le signal qui est enregistré est de celui de l'élongation relative (strain) des deux bras de l'interféromètre de Michelson.

1. En utilisant la fonction `loaddata` de la librairie `readligo`, extraire les données d'élongation relative et de temps des deux fichiers `H-H1_LOSC_4_V1-1126259446-32.hdf5` et `L-L1_LOSC_4_V1-1126259446-32.hdf5` :

```
from readligo import loaddata
filename_H1 = 'data/H-H1_LOSC_4_V1-1126259446-32.hdf5'
strain_H1, time_H1, chan_dict_H1 = loaddata(filename_H1, 'H1')
```

2. L'unité de temps est la seconde. Quelle est la durée de la mesure, le taux d'échantillonnage ?
3. Ces données sont centrées autour du moment où une onde gravitationnelle a été détectée, le 14 septembre 2015, vers 9h50. Tracer le signal pendant 10 secondes autour du passage de l'onde. Le passage de l'onde gravitationnelle a duré 100 ms environ. Peut-on voir ce signal ?
4. Tracer la densité spectrale de puissance entre 10 Hz et 2kHz. On utilisera la fonction `periodogram`. Puis on utilisera la fonction `welch` (toujours dans le module `scipy.signal`). La méthode de Welch effectue la moyenne de plusieurs périodogrammes pris consécutivement sur des durées plus courtes (argument `nperseg`). En diminuant la durée, on dimunie la résolution, mais en faisant la moyenne on améliore l'estimation de la DSP. On prendra une resolution de 1 Hz (i.e. on prendra `nperseg` égal au taux d'échantillonnage) :

```
f, psd = welch(strain_H1, samplerate, nperseg=samplerate)
```

5. Quel la valeur typique de la densité vers 100 Hz. Estimez la variance du signal après un filtre entre 30 et 300 Hz. Qu'en pensez-vous ?
6. Le problème est qu'il y a des pics de résonance qui représentent la majeur partie du bruit. Pour les supprimer, on va appliquer un filtre dont le gain en puissance sera inversement proportionnel à la densité spectrale de puissance.

Voici comment ce filtre sera appliqué

```python
from scipy.interpolate import interp1d

def whiten(strain, dt):
    freqs_welch, psd_welch = welch(strain, fs=1/dt, nperseg=int(1/dt))
    interp_psd = interp1d(freqs_welch, psd_welch)

    strain_tilde = np.fft.rfft(strain)
    N = len(strain)
    freqs = np.fft.rfftfreq(N, dt)

    gain = 1 / np.sqrt(interp_psd(freqs))
    gain = gain/gain.max()
    white_strain_tilde = strain_tilde * gain
    white_strain = np.fft.irfft(white_strain_tilde)
    return white_strain
```

7. Il reste encore du bruit à haute et basse fréquence. Appliquer un filtre passe bas de fréquence 300 Hz et un filtre passe haut de fréquence 20 Hz. Peut-on voir le signal ? Tracer le signal des deux détecteurs l'un à côté de l'autre, en changeant le signe d'un des deux signal. Quel est le délai entre les deux signaux ?
8. Sur cet événement, on peut facilement voir le signal sur chaque détecteur car il est supérieur au bruit après filtrage. Cependant, il est aussi possible de corréler les deux détecteurs. On peut alors appliquer un filtre passe bas de plus passe fréquence car après corrélation, le signal ne se moyenne plus à zéros. Effectuer le produit des deux signaux et filtrer avec un filtre passe bas de fréquence de coupure à 10 Hz. Donner un ordre de grandeur du bruit RMS et de l'amplitude du signal. Quel est le facteur d'amélioration sur le rapport signal sur bruit ?