Rapport Filtrage

Xavier Bourlot, 3IMACS-AE-C

30 mai 2020

Table des matières

1 Compte-Rendu **1**
1.1 Analyse du signal . . . 1
1.2 Filtrage . . . 1
1.3 Resultat . . . 2

2 Conclusion **3**

3 Annexes **3**
3.1 Audio . . . 3
3.2 Code . . . 3

1 Compte-Rendu

1.1 Analyse du signal

Après une écoute préliminaire du son, on entend des sifflements à 2 fréquences distinctes. Pour pouvoir cerner exactement l'emplacement sur le spectre de ces fréquences, on réalise un spectrogramme. Il permet d'effectuer plusieurs FFT sur des fenêtres de longueur définie, avec ou sans chevauchement. Le résultat est donc une FFT au cours du temps. On adapte les différents paramètres de la fonction pour pouvoir clairement discerner les sifflements. Le son étant enregistré en stéréo, on utilise uniquement la composante gauche su signal pour l'identification. On obtient le spectrogramme suivant :

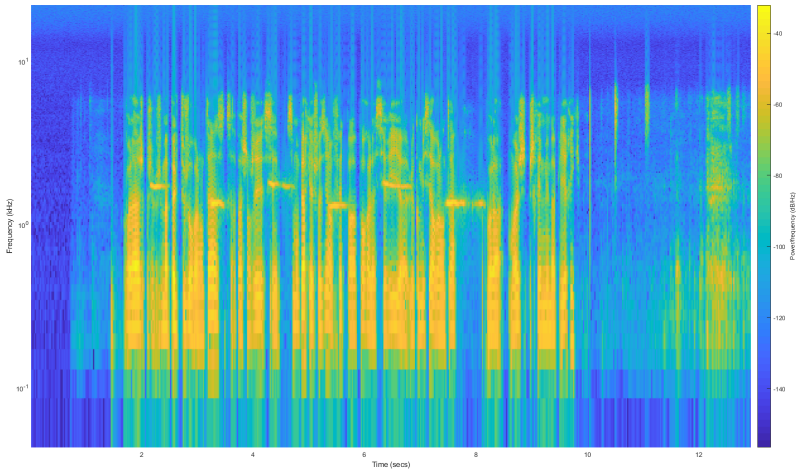

FIGURE 1 – Spectrogramme initial

On identifie les fréquences hautes et basses de chaque sifflement, qui sont les paramètres nécéssaires pour créer le filtre.

	F1(Hz)	F2(Hz)
Fmin	1209	1597
Fmax	1418	1849

1.2 Filtrage

On choisit d'utiliser deux filtres coupe-bande de type butterworth. On prendra soin de normaliser la fréquence par rapport à Fs=44100Hz. L'ordre du filtre est déterminé expérimentalement, d'après la réponse fréquentielle. On veut une atténuation suffisante pour que le sifflement soit du même niveau sonore que le bruit de fond. Ainsi, sur le spectrogramme, on veut "passer du orange au bleu",

c'est à dire de -40dB à -130dB environ. Il faut donc une atténuation d'environ 90dB. Un ordre 4 semble suffisant.

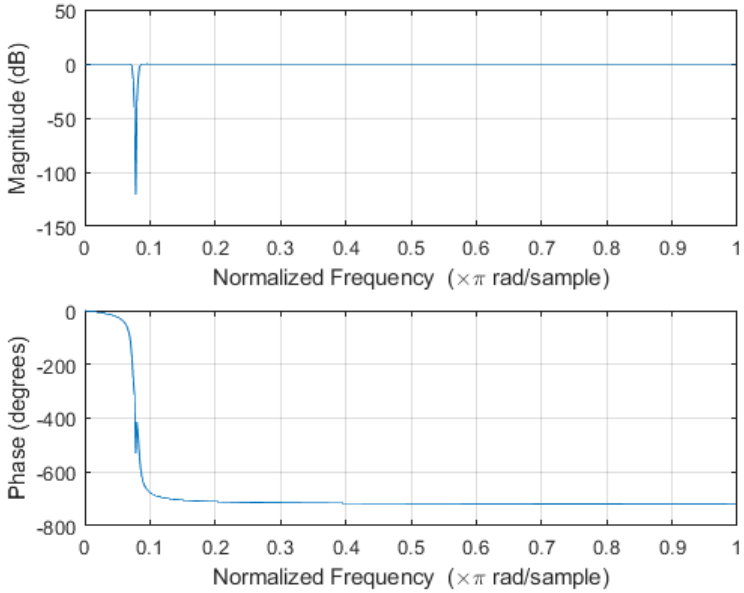

FIGURE 2 – Filtre coupe-bande pour F2

On applique successivement ces filtres aux 2 canaux audio.

1.3 Resultat

Afin d'observer le résultat, on réalise de nouveau un spectrogramme avec les mêmes paramètres que précédemment.

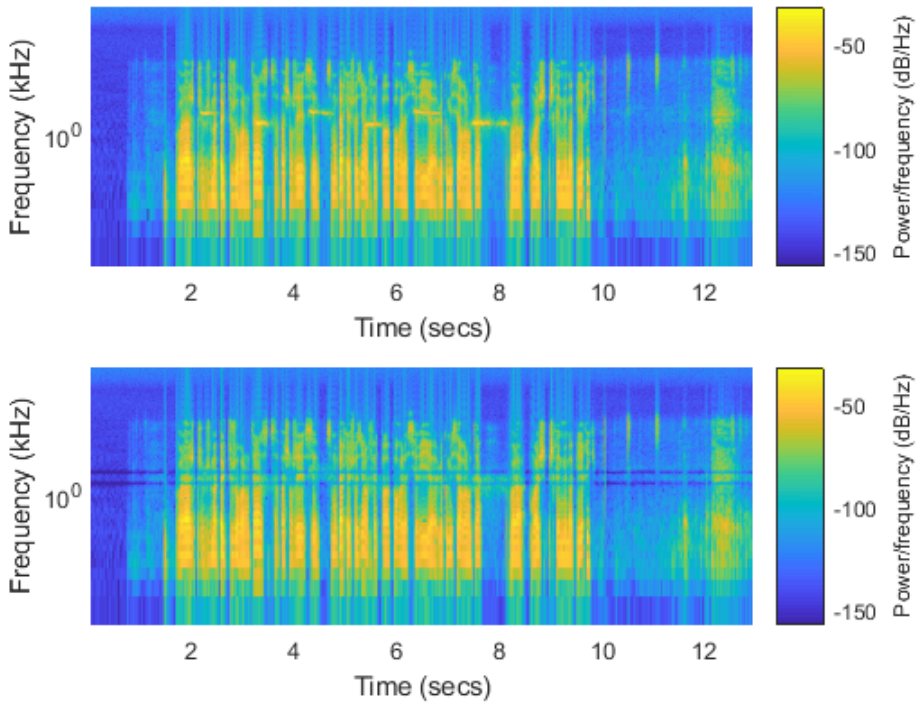

FIGURE 3 – Spectrogrammes avant / après filtrage

On observe 2 lignes sombres, correspondant chacune à un filtre. Les sifflements sont bien contenus dans les bandes de coupures, et le signal alentour est faiblement affecté par les filtres car les bandes de coupure sont suffisamment étroites.

2 Conclusion

Ainsi, le signal a pu être filtré avec succès. J'ai particulièrement apprécié que ce projet se déroule en autonomie totale, car il m'a forcé à rechercher par moi-même des solutions et à comprendre les manipulations effectuées. Pouvoir entendre le résultat permet également de bien se rendre compte du potentiel de ces méthodes.

3 Annexes

3.1 Audio

Fichier WAV initial : Fichier WAV résultat :

3.2 Code

```
   breaklines
clear all
clc
%chargement du fichier audio
[data,Fs]=audioread('valentin-bruite.wav');
y1=data(:,1);
y2=data(:,2);

%plot spectrogramme avant filtrage (canal gauche)
figure(1)
subplot(2,2,1:2);
title("Spectre avant filtrage");
spectrogram(y1,1000,0,1024,Fs,'yaxis');
ax = gca;
ax.YScale = 'log';

%frequences de coupure basses et hautes relevees experimentalement sur le
%spectrogramme
f1=[1209,1418];
f2=[1597,1849];

%creation d'un filtre coupe-bande pour chaque frequence indesirable (ordre
%4 pour plus de rejection)
[b1,a1] = butter(4,f1/(Fs/2),'stop');%attention a la normalisation des frequences
[b2,a2] = butter(4,f2/(Fs/2),'stop');

%application des 2 filtres aux 2 canaux
dataOutL=filter(b1,a1,y1);
dataOutL=filter(b2,a2,dataOutL);
dataOutR=filter(b1,a1,y2);
dataOutR=filter(b2,a2,dataOutR);
dataOut=[dataOutL,dataOutR];

%plot spectrogramme apres filtrage (canal gauche)
subplot(2,2,3:4);
title("Spectre apres filtrage");
spectrogram(dataOut(:,1),1000,0,1024,Fs,'yaxis');
ax = gca;
ax.YScale = 'log';

%plot des reponses frequentielles des 2 filtres
figure(2)
title("Reponse filtre coupe-bande 1");
freqz(b1,a1);
figure(3)
title("Reponse filtre coupe-bande 2");
freqz(b2,a2);

%export du resultat en .wav
audiowrite('res.wav',dataOut,Fs);
```