Rapport Filtrage Numérique TP2

Cameron Bray Xavier Bourlot

3IMACS-AE-C
30 mai 2020

Table des matières

1 Introduction

Dans ce TP, nous nous proposons d'identifier des consonnnes (ch,j,f,v,s,z) à partir d'un enregistrememement. L'objectif est d'écrire un programme permettant de trouver pour chaque son la consonne associée la plus probable. Cet exercice nous permettra d'approfondir nos connaissances du logiciel Matlab et des filtres numériques.

2 Analyse du signal

2.1 Analyse préliminaire

Après une écoute préliminaire de l'enregistrement, il apparait nécessaire de fragmenter la bande son en segments contenant chacun une consonne. Pour ce faire, 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 différentes gammes de fréquences associées à chaque son. On peut notamment augmenter le nombre de points de la FFT pour augmenter la résolution en fréquence, et diminuer la taille des fenêtres pour mieux discerner les changements en temporel. On obtient le spectrogramme suivant :

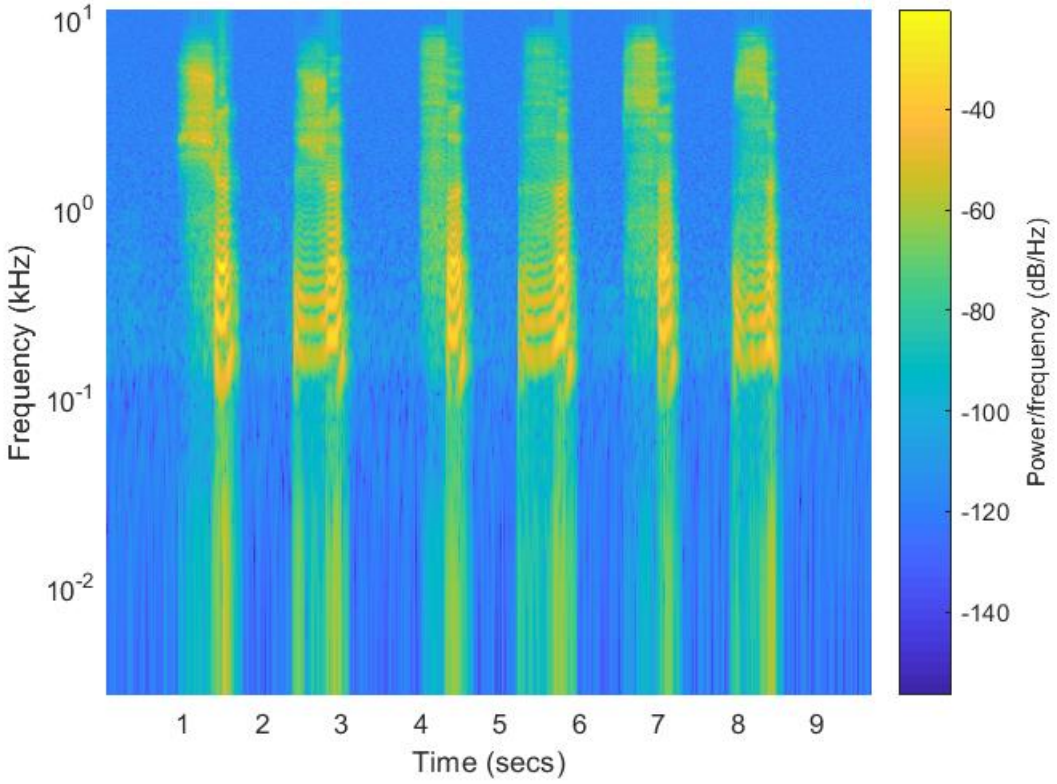

FIGURE 1 – Spectrogramme complet

2.2 Segmentation du signal

On note en premier lieu la présence du son [ə] à la fin de la prononciation de chaque consonne. Ce son étant commun à tous les enregistrements et contenant une large gamme de fréquences, il sera plus aisé d'identifier les consonnes uniquement sur la partie initiale du son. On relève donc en temporel les dates de début et fin de chaque consonne.
De plus, on remarque trois gammes de fréquences distinctes, qui permettent de discriminer les consonnes. En effet, chaque consonne possède plus ou moins d'énergie dans chaque bande, et la combinaison de chaque bande permet d'identifier de manière unique chaque consonne.

Gammes de fréquences	F1	F2	F3
Fmin (Hz)	3750	1680	120
Fmax (Hz)	8000	3650	1100

Consonne	'ch'	'j'	'f'	'v'	's'	'z'
Début (s)	.9524	2.404	3.991	5.215	6.553	7.937
Fin (s)	1.361	2.766	4.286	5.646	6.939	8.322
Fréquences	F1,F2	F1,F2,F3	$\emptyset$	F3	F1	F1,F3

TABLE 1 – Gammes de fréquences discriminantes et délimitations temporelles pour chaque consonne

3 Filtrage

On souhaite isoler l'énergie présente pour chaque consonne dans chaque bande de fréquence F1 à F3. Pour ce faire, on applique des filtres passe bande sur le signal dans la gamme choisie. Cela permet de ne retenir que le signal d'interêt, et ensuite de caluler son énergie. On considère que le signal se confond du bruit de fond au delà de -100dB.
On choisit des filtres de type elliptiques car les bandes de fréquences sont rapprochées et il est nécéssaire d'avoir un gabarit de filtre presque vertical pour obtenir une séparation suffisante.
Les paramètres du filtre sont l'ordre, l'atténuation entre la bande passante et la bande de coupure, et l'oscillation dans la bande passante. On choisit 150dB d'atténuation, 10dB de ripple et un ordre 10. On ajuste les paramètres visuellement grâce à la réponse fréquentielle superposée des trois filtres. Ainsi, un ordre 10 pour la bande de fréquence la plus étroite est un critère trop contraignant, on ajuste donc l'ordre à 6 pour ce filtre.
On minimise l'ordre dans un souci de limiter le nombre de coefficients du filtre (ici 21) et le temps de calcul.
La réponse fréquentielle des filtres est la suivante :

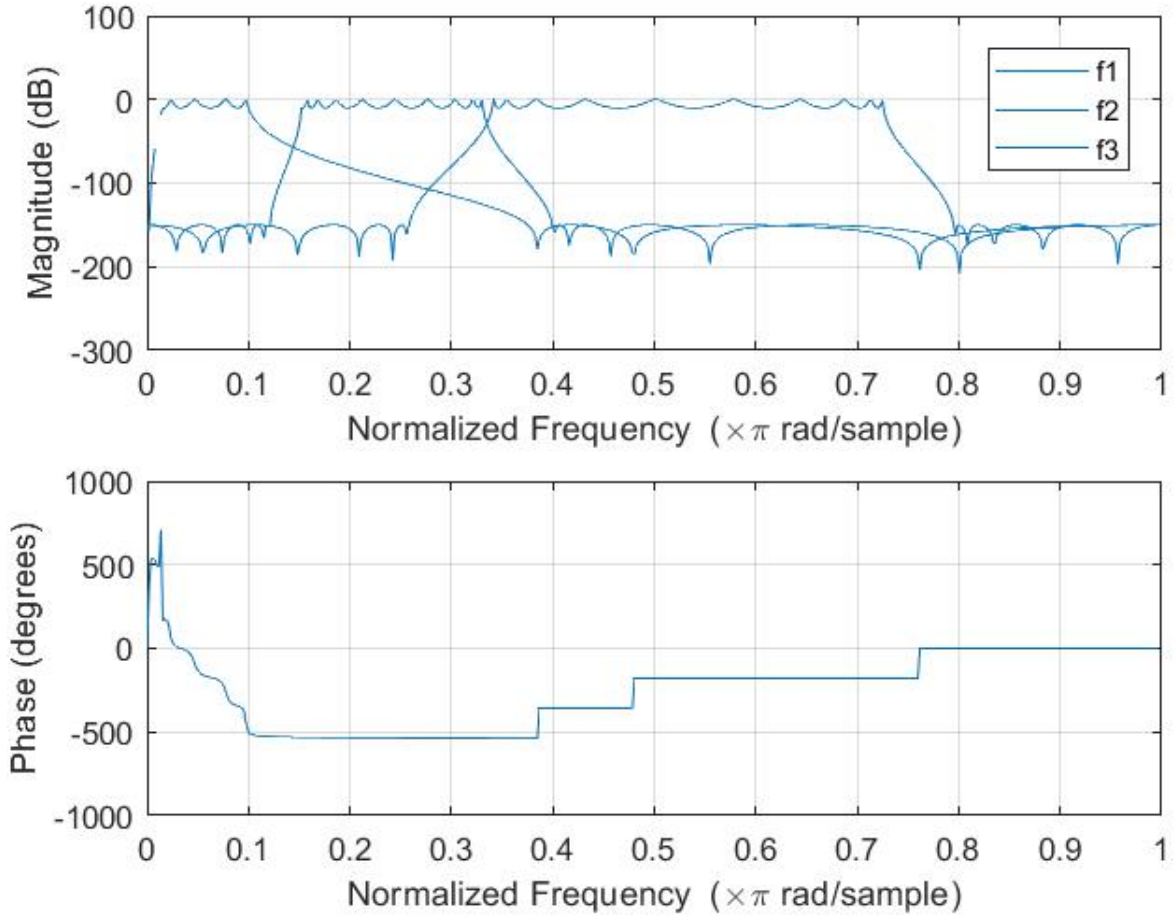

FIGURE 2 – Filtres passe-bande pour F1, F2 et F3

On observe que les bandes délimitées par les filtres ne se recoupent pas, ils nous permettront donc de calculer l'énergie présente dans chaque bande séparément.

4 Décision

L'énergie contenue dans un signal $x(t)$ vaut $E_s = \sum_{n=-\infty}^{+\infty} |x(t)|^2$. On calcule l'énergie contenue dans chacune des bandes. Il faut ensuite déterminer si la bande contient de l'énergie ou non. On fixe un seuil à 0.012 par observation des résultats de calculs de puissances. C'est le seuil qui permet de différencier un signal présent d'un bruit de fond. On a maintenant une réponse binaire (0 ou 1) pour chaque bande pour chaque consonne.
Il suffit alors d'associer chaque combinaison de 0 et 1 par bande à une consonne par une série de *if .. else if .. else*, selon la table de vérité suivante :

Consonne	'ch'	'j'	'f'	'v'	's'	'z'
F1	✓	✓	∅	∅	✓	✓
F2	✓	✓	∅	∅	∅	∅
F3	∅	✓	∅	✓	∅	✓

TABLE 2 – Relation gamme de fréquences / consonne

5 Conclusion

Ainsi, nous avons pu identifier avec succès les différentes consonnes, en comparant la présence d'énergie dans trois bandes caractéristiques. On peut cependant regretter une part manuelle non négligeable du traitement de l'enregistrement. Par exemple, la segmentation du début de chaque son aurait pu être automatisée en détectant le debut et la fin de l'émission, puis en sélectionnant un pourcentage fixe de la durée du son, correspondant à la partie "consonne" du son, en éliminant la partie [ə]. Le seuil d'énergie pour prendre la décision est déterminé expérimentalement, il ne tient donc pas compte d'éventuels changements de volume globaux. Il faudrait ajuster dynamiquement ce seuil en fonction de l'énergie totale du son par exemple.

A Annexes

A.1 Code Matlab

```
clc
clear all;
close all;

%import du fichier audio
[data,Fs]=audioread('consonnes_h_tp2.wav');

%representation temporelle du signal
figure;
plot(data)

%spectrogramme permettant de determiner les gammes de frequences discriminantes et les bornes temporelles de chaque consonne
figure;
spectrogram(data,1000,500,8192,Fs,'yaxis');
ax = gca;
ax.YScale = 'log';

%bode des 3 filtres combines
figure;
order=10
att=150%passband to stopband attenuation in dB
passripple=10;%passband ripple
%F1
F1=[3750 8000];
[b1 a1]=ellip(order,passripple,att,F1/Fs*2,'bandpass');
freqz(b1,a1);
hold on;

%F2
F2 =[1680 3650];
[b2 a2]=ellip(order,passripple,att,F2/Fs*2,'bandpass');
freqz(b2,a2);
hold on;

%F3
F3 =[120 1100];
[b3 a3]=ellip(order-4,passripple,att,F3/Fs*2,'bandpass');
freqz(b3,a3);
legend('f1','f2','f3')


%bornes temporelles et segments du vecteur son initial de chaque consonne
%ch 1,2
x1=[0.9524 1.361];
data1=data(floor(x1(1)*Fs):floor(x1(2)*Fs));
%j 1,2,3
x2=[2.404 2.766];
data2=data(floor(x2(1)*Fs):floor(x2(2)*Fs));
%f
x3=[3.991 4.286];
data3=data(floor(x3(1)*Fs):floor(x3(2)*Fs));
%v 3
x4=[5.215 5.646];
data4=data(floor(x4(1)*Fs):floor(x4(2)*Fs));
%s 1
x5=[6.553 6.939];
data5=data(floor(x5(1)*Fs):floor(x5(2)*Fs));
%z 1,3
x6=[7.937 8.322];
data6=data(floor(x6(1)*Fs):floor(x6(2)*Fs));

figure
answer(evaluate(data1,a1,b1,a2,b2,a3,b3));
evaluate(data2,a1,b1,a2,b2,a3,b3)%123
evaluate(data3,a1,b1,a2,b2,a3,b3)%0
evaluate(data4,a1,b1,a2,b2,a3,b3)%3
evaluate(data5,a1,b1,a2,b2,a3,b3)%1
evaluate(data6,a1,b1,a2,b2,a3,b3)%13


function answer(p)
    if(p(1) && p(2) && p(3))
        'jjjjjjjjjjj'
    elseif (p(1) && p(2))
        'chchchchc'
    elseif (p(1)&&p(3))
        'zzzzzzzzz'
    elseif (p(1))
        'ssssssss'
    elseif (p(3))
        'vvvvvvvvv'
    else
        'fffffffffff'
    end
end

%fonction appliquant les filtres pour determiner la presence ou non d'un signal dans chaque bande
function p=evaluate(x,a1,b1,a2,b2,a3,b3)%
    x1=filter(b1,a1,x);
    %calcul de puissance dans la bande F1
    p(1)=sum(abs(x1))/length(x);

    %calcul de puissance dans la bande F2
    x2=filter(b2,a2,x);
    p(2)=sum(abs(x2))/length(x);

    %calcul de puissance dans la bande F3
    x3=filter(b3,a3,x);
    p(3)=sum(abs(x3))/length(x);

    %resultat en 0 ou 1 si la puissance depasse un seuil min
    for i=1:3
        if(p(i)>0.012)
            p(i)=1;
        else
            p(i)=0;
        end
    end
end
```