École de technologie supérieure

Université du Québec

Rapport première étape de projet

Présenté à M. Marcel GABREA

Dans le cadre du cours

*SYS835 – Processeur numérique du signal et ses applications*

PAR

Eric LACERTE LACE23038502

Philippe LAVOIE LAVP05067200

MONTRÉAL, LE 8 NOVEMBRE 2017

# Introduction

Ce document décrit l’implémentation et la simulation de l’algorithme de suppression du bruit en temps réel élaboré par McAulay et Malpass[[1]](#footnote-1). Tout d’abord nous décrirons l’implémentation de ce dernier à l’aide du langage Matlab. Nous ferons également un bref survol théorique pour chacune des parties implémentées. Nous terminerons avec les résultats de la simulation et une analyse des résultats obtenus.

# Implémentation

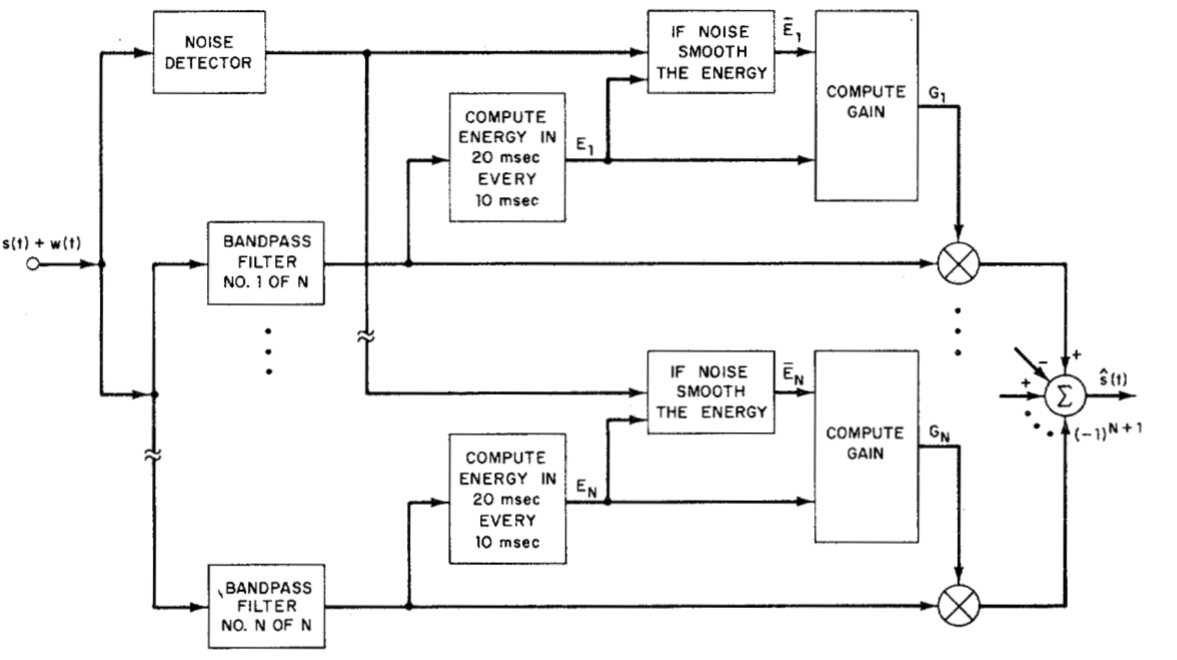


Figure Schéma Bloc Filtre de Suppression de bruit[[2]](#footnote-2)

## Détecteur de bruit

Le signal est divisé en trame de 20 ms. Nous devons déterminer pour chaque trame si elle contient du bruit ou de la voix. Pour ce faire nous utilisons une implémentation de l’algorithme de détection du bruit décrit dans l’article [2].

L’algorithme utilise un histogramme de 4s de l’énergie de chaque trame pour déterminer de façon adaptative le niveau d’énergie correspondant à la frontière entre le bruit et le bruit et la voix.

L’algorithme de l’article détermine qu’un signal n’est pas du bruit si son énergie dépasse 16 bits. Ceci est probablement dû à la quantification utilisée dans système décrit. Pour pouvoir utiliser l’algorithme tel quel, nous réduisons l’échelle de nos échantillons à 8 bits lors du calcul de l’énergie. Notre implémentation se trouve dans la fonction noisedetector ( *noisedetector.m )*.

## Filtres Passe-bande

Le système utilise 19 filtres Butterworth de 2e ordre pour séparer le signal en bandes de fréquences et déterminer le niveau de bruit moyen et calculer le gain à appliquer dans chaque bande. Dans l’implémentation Matlab, nous générons les coefficients en utilisant les fréquences centrales et bandes passantes de l’échelle de *Bark* [3] plutôt que les celles utilisées dans l’article, puisque la fréquence d'échantillonnage des signaux que nous traitons est différente de celle du système décrit dans [1] (8 kHz vs 7.575 kHz).

## Calcul de l’énergie

L’énergie est la somme des carrés de chaque échantillon de la trame analysée. L’énergie est calculée pour chaque bande *n* sur les *N* échantillons de la trame selon l’équation:

## 

## Détermination du niveau de bruit

Lorsque le détecteur de bruit détermine qu’une trame ne contient que du bruit, le niveau de bruit moyen dans chaque bande *n* est déterminé par la fonction de lissage récursive sur 1s (50 trames de 20 ms) suivante:

## Détermination du gain

Ce calcul est illustré dans le code Matlab à la section *Détermination du gain*.

Le gain de suppression est déterminé à l’aide d’une fonction de Bessel modifiée[[3]](#footnote-3) :

où

Nous aurons besoin de déterminer le paramètre gn(m), celui-ci étant le ratio de la différence entre l’énergie de la trame actuelle du canal et l’énergie du bruit de la trame précédente sur l’énergie de la trame actuelle du canal. Plus ce ratio est élevé, plus la trame actuelle peut être considérée comme de la voix. Donc plus ce paramètre est élevé, plus l’atténuation sera faible. Dans le code Matlab, gn(m) est désigné par *ch\_meas\_parms* dans *main.m.*

Le paramètre ξ est arbitraire et définit au début du code MatLab par l’utilisateur. Il doit être compris entre 1 et 20.

La calcul du gain est effectué par la fonction *func\_suppress\_curve(eps, v\_parm)* du fichier *func\_suppress\_curve.m*. Cette dernière étant l’implémentation de la fonction de Bessel modifiée.

## Application du gain et reconstitution du signal

Après avoir déterminé le gain à appliquer au signal du canal, un lissage est effectué afin d’éviter les changements trop brusques du signal. Ces brusques sauts pourraient altérer la qualité du signal perçue par l’usager. Ce lissage est décrit par l’équation suivante :

où

Cette équation est implémentée par la fonction *calc\_smooth\_gain*  du fichier *calc\_smooth\_gain.m.*

Une fois le gain lissé, celui-ci est appliqué sur le signal du canal. Tel qu’expliqué précédemment, ce gain sera beaucoup plus faible pour les trames contenant du bruit.

Par la suite, nous recombinons les signaux des différents canaux avec un déphasage de 180° entre eux. Ceci est exprimé dans le code de *main.m* par une alternance d’addition et de soustraction des signaux de canaux.

# Simulation et Résultats

Nous avons utilisé un extrait sonore (*car.wav*) comme entrée, tiré du site de Signalogic,[[4]](#footnote-5) afin de valider l’implémentation et mesurer l’efficacité de l’algorithme. Cet extrait, échantillonné à 8 kHz, est une conversation avec la présence d’un bruit de fond.

À l’aide d’un script (*analysis\_data.m*) que nous avons développé, nous pouvons visualiser un signal sur l’échelle du temps et la réponse fréquentielle de ce dernier. Cela permet d’analyser les performances de notre algorithme.

## Signal original d’entrée

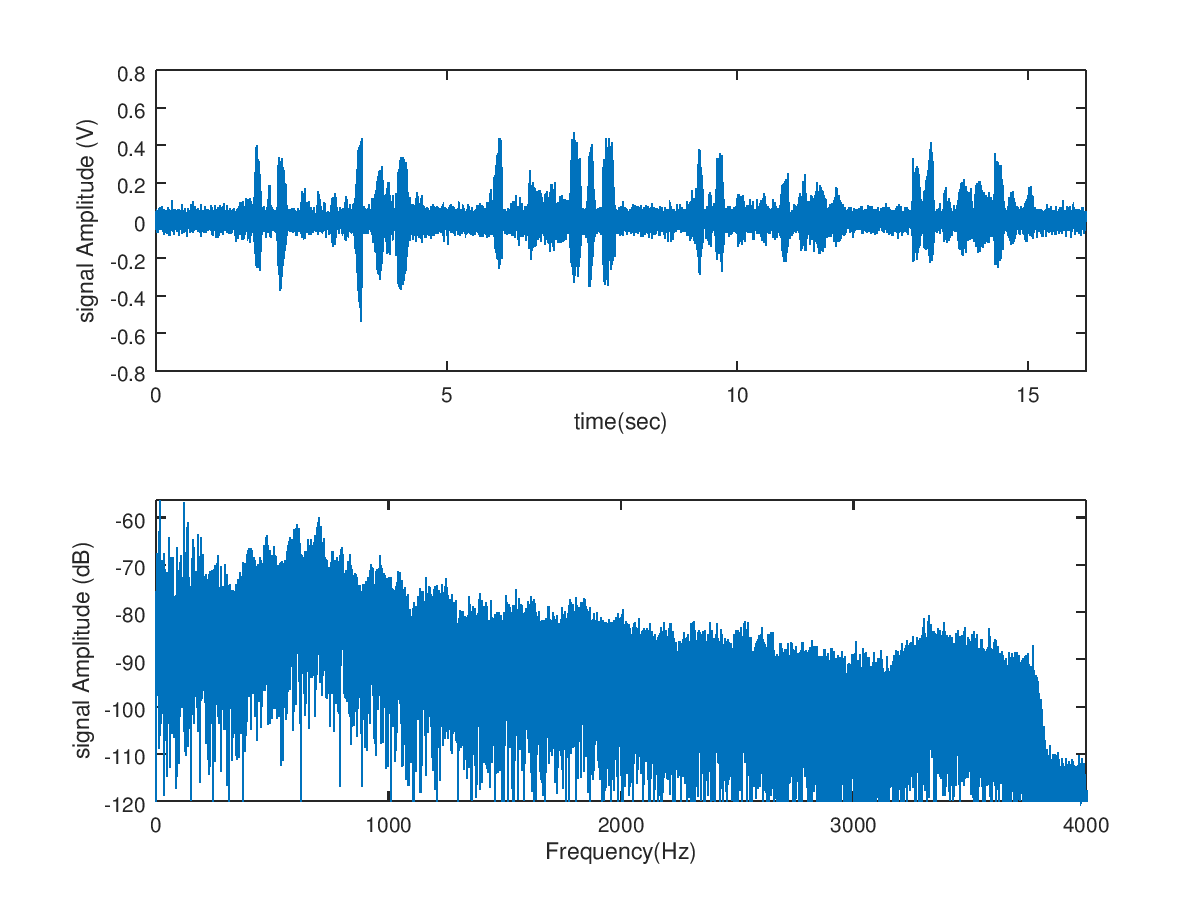


Figure Signal original, car.wav

## Signaux résultants

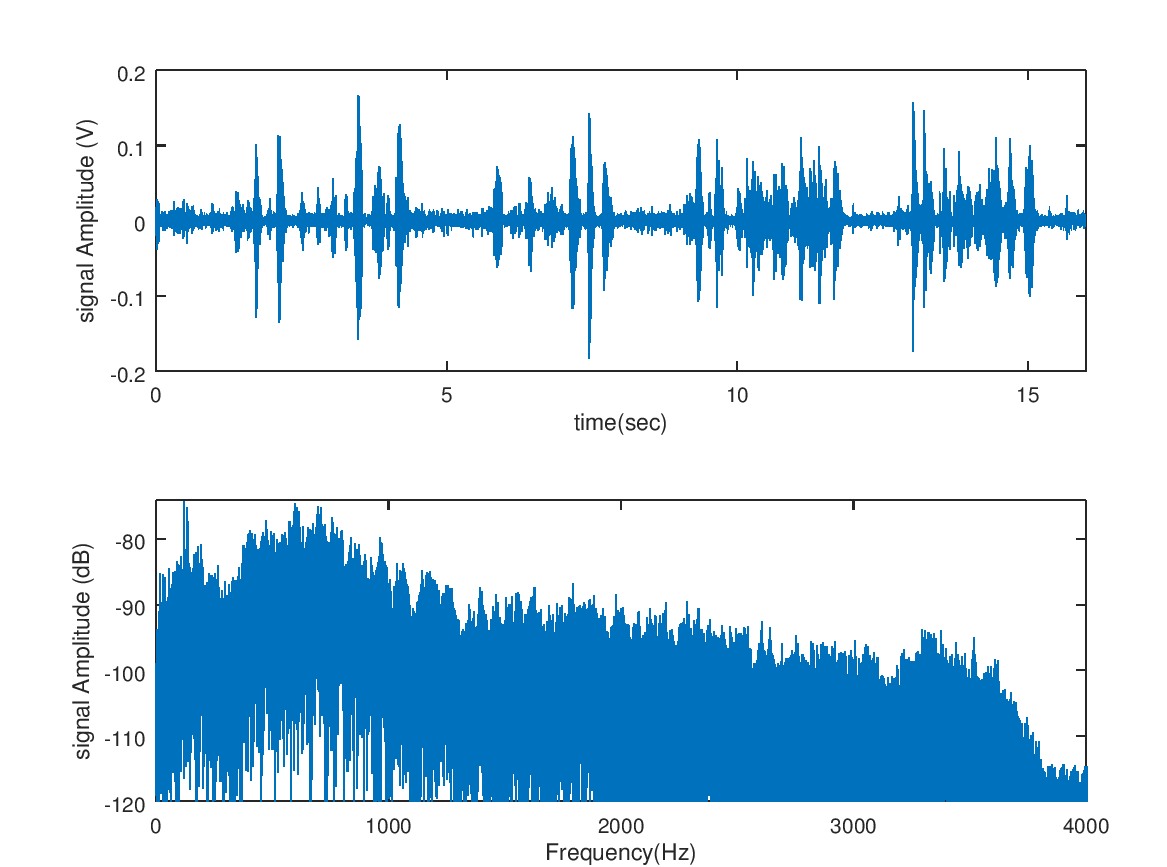


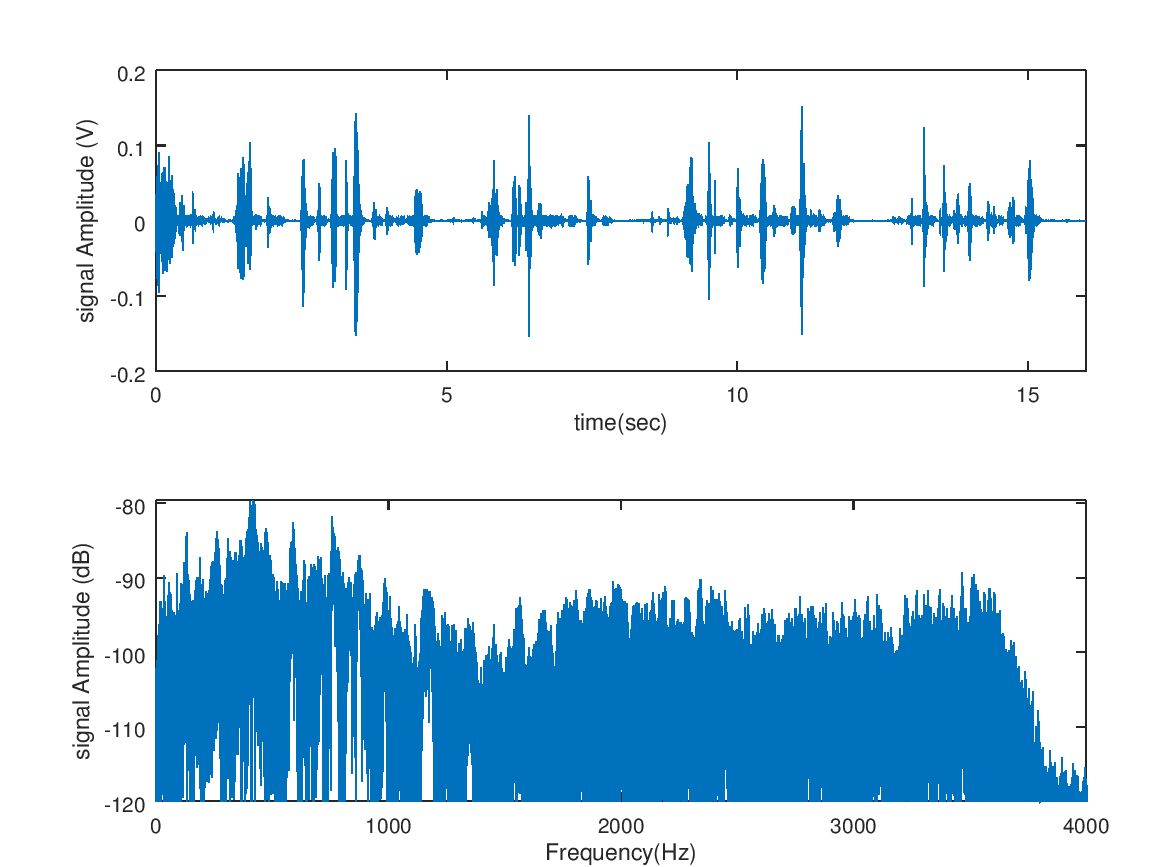
Figure Signal résultant (α = 0.4 ξ = 5)

Figure Signal résultant (α =0.4 ξ =15)

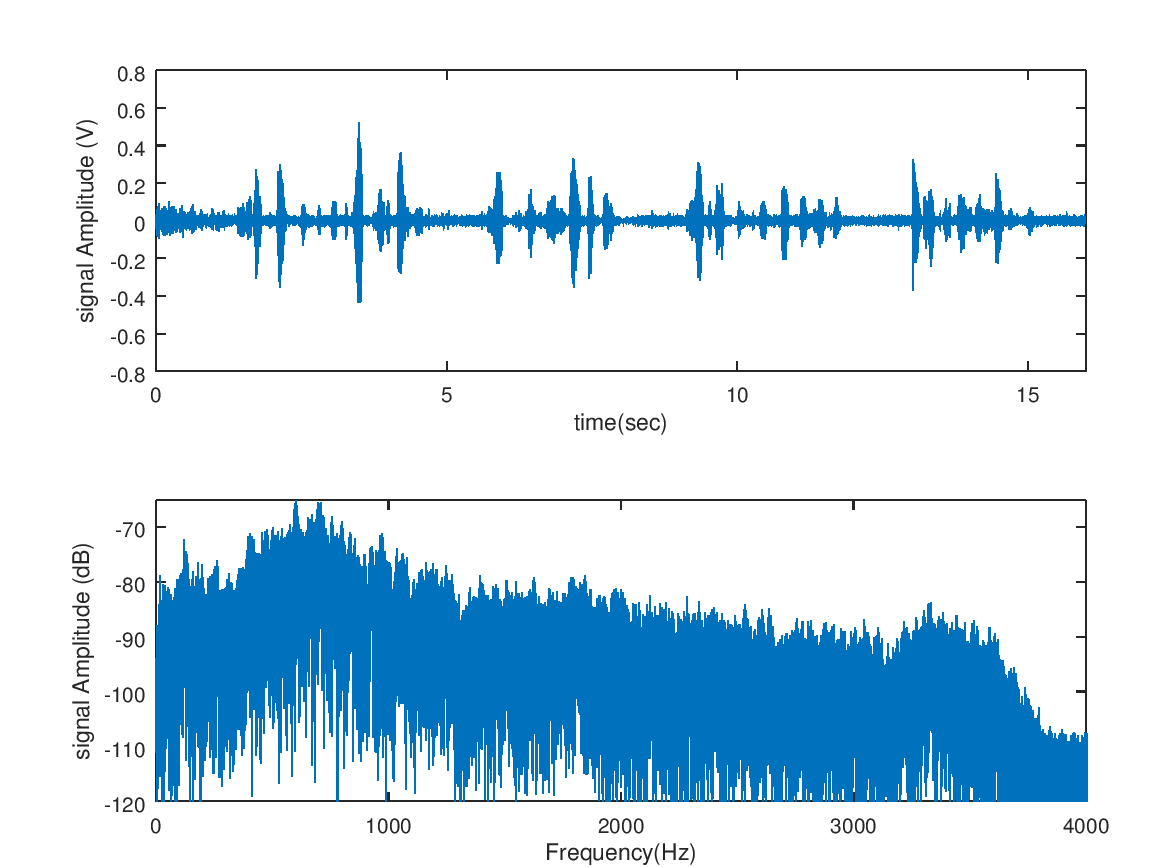


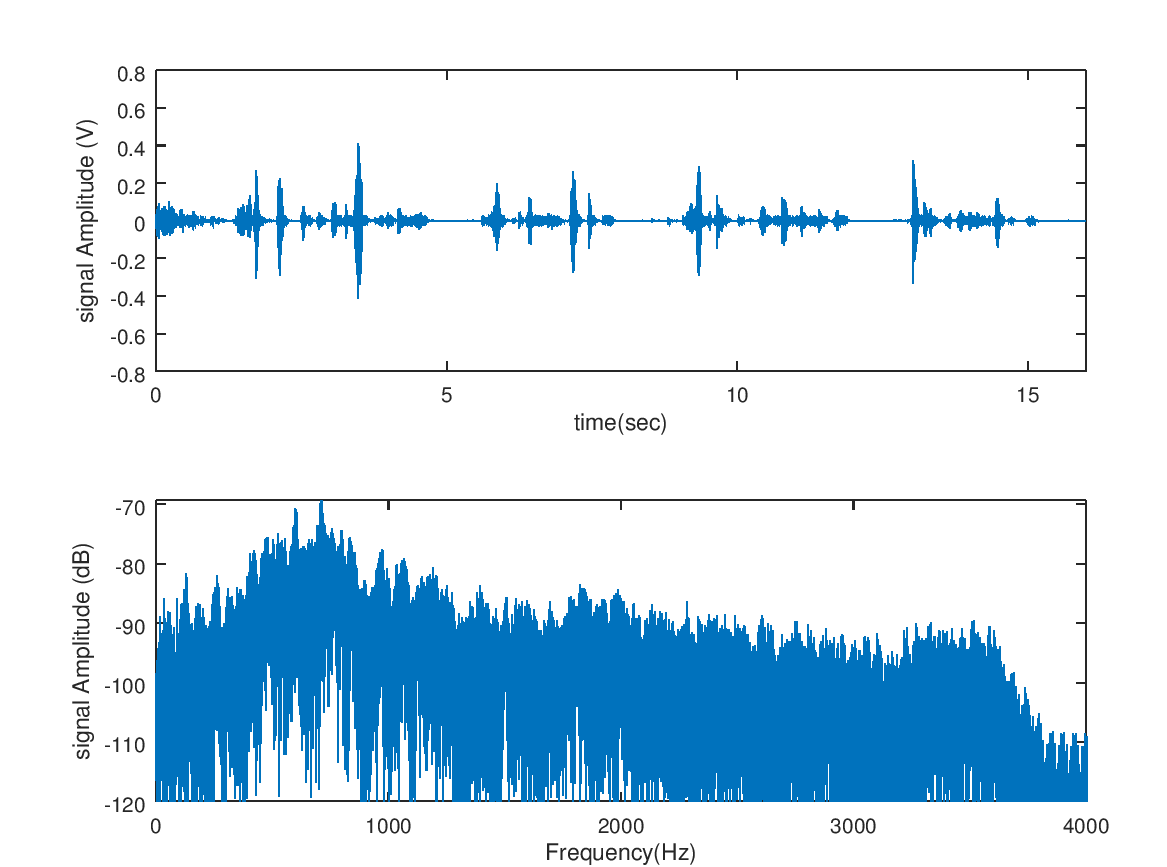
Figure Signal résultant (α = 0.1 ξ = 5)

Figure Signal résultant (α = 0.1 ξ = 15)

## Filtres Passe Bande des canaux

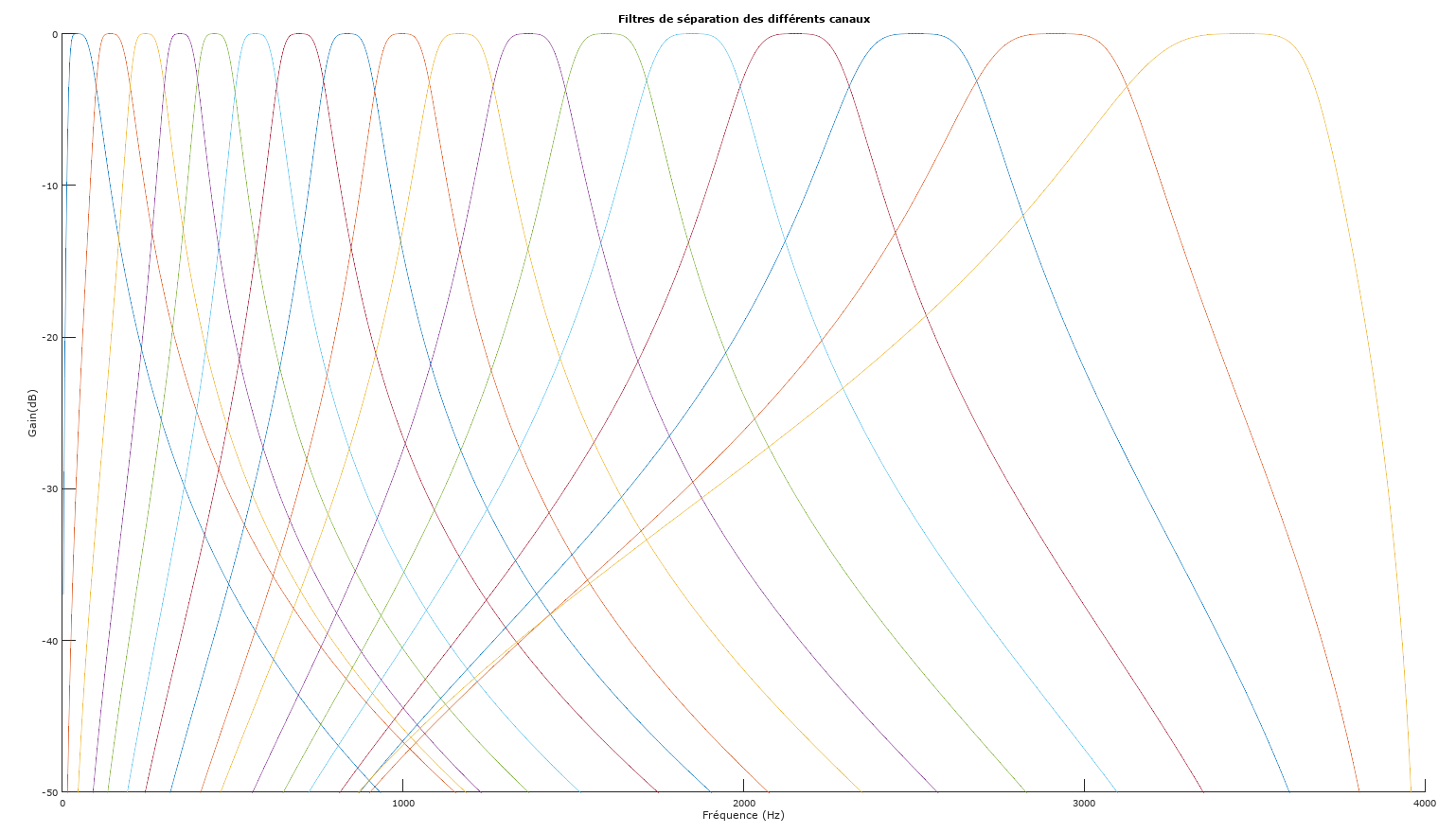
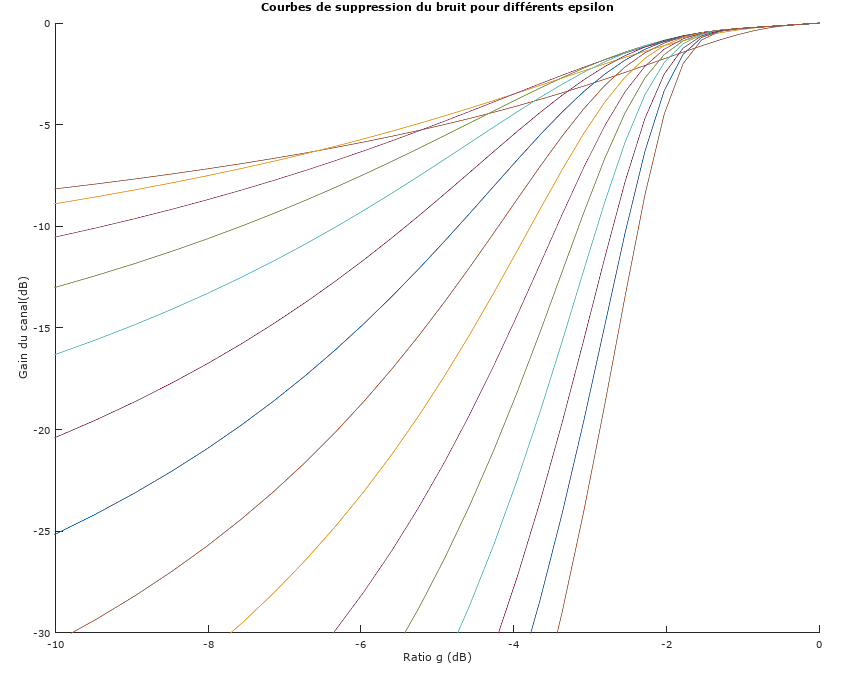


Figure Filtres de séparation des différents canaux

## Courbe de suppression du bruit



ε=1

ε=20

Figure Courbe de suppression du bruit pour différentes valeurs de ξ

# Conclusion et Analyses

Suite à l’implémentation et la simulation à l’aide de Matlab, nous constatons que l’algorithme fonctionne correctement. Nous avons constaté qu’il y a deux paramètres arbitraires que nous pouvons configurer afin d’optimiser la performance de l’algorithme, soit α et ξ .

Le paramètre α nous permet d’adoucir ou d’accentuer l’impact des trames précédentes dans le calcul du bruit. De plus, ceci nous permet une forme de filtre afin d’amoindrir l’impact des brusques variations du bruit de fond.

Le paramètre ξ permet de contrôler l’agressivité du gain de suppression du bruit. Plus ce paramètre est élevé plus l’atténuation du bruit sera élevée. Cependant, une valeur trop élevée engendra une dégradation du signal de voix.

Il est intéressant de noter, tel que mentionné dans l’article de McAulay et Malpass, que nous sommes en mesure d’annihiler complètement le bruit de fond mais cela engendre une distorsion dans le signal de voix. Il faut bien configurer les paramètres afin de trouver un compromis acceptable entre la distorsion de la voix et le niveau du bruit de fond. Les meilleures performances ont été observées pour des valeurs de α 0,1 et ξ de 5.

# Références

[1] Robert J. McAulay and Marilyn L. Malpass, (1980) A REAL-TIME NOISE SUPPRESSION FILTER FOR SPEECH ENHANCEMENT AND ROBUST CHANNEL VOCODING

[2] Robert J. McAulay and Marilyn L. Malpass, (1980) SPEECH ENHANCEMENT USING A SOFT-DECISION NOISE SUPPRESSION FILTER

[3] Échelle de Bark https://en.wikipedia.org/wiki/Bark\_scale

1. Robert J. McAulay and Marilyn L. Malpass, (1980) A REAL-TIME NOISE SUPPRESSION FILTER FOR SPEECH ENHANCEMENT AND ROBUST CHANNEL VOCODING [↑](#footnote-ref-1)
2. Robert J. McAulay and Marilyn L. Malpass, (1980) SPEECH ENHANCEMENT USING A SOFT-DECISION NOISE SUPPRESSION FILTER [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. http://www.signalogic.com/index.pl?page=codec\_samples [↑](#footnote-ref-5)