ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
ANTOINE MUNOZ

REHAUSSEMENT D'UN SIGNAL DE LA PAROLE ALTÉRÉ PAR UN BRUIT CONVOLUTIF ET ADDITIF À L'AIDE DE LA TRANSFORMÉE EN ONDELETTES.

MONTRÉAL, LE 4 JANVIER 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Christian Gargour, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Marcel Gabréa, codirecteur
Département de génie électrique à l'École de technologie supérieure

M. Jean-Marc Lina, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Christian Perron, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 2 DÉCEMBRE 2005

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

SOMMAIRE

Cette étude porte sur la conception et la simulation d'un système de rehaussement de la parole avec bruits convolutifs et additifs, à l'aide de la transformée en ondelettes dans le but de l'appliquer sur un processeur dédié au traitement numérique du signal.

Les méthodes de rehaussement de la parole considérées ici, sont celles qui visent la réduction du bruit convolutif et additif générés par les microphones utilisés pour capter la parole ou par ceux résultant de la réverbération. Le but étant de reconstruire un signal aussi proche que possible de l'original.

La littérature existante a été étudiée. Le choix s'est porté sur une méthode adaptée et modifiée à nos besoins pour estimer la parole entachée de bruits additifs et convolutifs à l'aide l'analyse en composantes indépendantes avec utilisation de la transformée en ondelettes. La simulation s'est faite sous MATLAB. Sa mise en œuvre a été réalisée en langage C.

REHAUSSEMENT D'UN SIGNAL DE LA PAROLE ALTÉRÉ PAR UN BRUIT CONVOLUTIF ET ADDITIF À L'AIDE DE LA TRANSFORMÉE EN ONDELETTES

Antoine Munoz

SOMMAIRE

Cette étude porte sur la conception et la simulation d'un système de rehaussement de la parole en présence des bruits convolutifs et additifs, à l'aide de la transformée en ondelettes dans le but d'effectuer une réalisation à l'aide d'un processeur dédié au traitement numérique du signal.

Les méthodes de rehaussement de la parole qui seront considérées dans ce travail sont celles qui visent la réduction du bruit convolutif et additif tel que celui qui est généré par le ou les microphones utilisés pour capter la parole ou celui résultant de la réverbération provenant du milieu ambiant. Le but étant de reconstruire un signal aussi proche que possible de l'original.

La transformée en ondelettes est un outil puissant que plusieurs chercheurs ont déjà utilisée et elle est mise à contribution dans cette étude. Le projet de recherche proposé ici, consiste donc à appliquer différentes méthodes convenant au type de débruitage mentionné plus haut.

Une étude de la littérature existante a été effectuée. Le choix s'est porté sur une méthode plus particulièrement adaptée à nos besoins. Cette méthode consiste principalement en l'estimation de la parole entachée de bruits additifs et convolutifs par une analyse en composantes indépendantes avec utilisation de la transformée en ondelettes . La simulation de cette méthode a été effectuée avec certaines adaptations et modifications à l'aide du logiciel MATLAB et des signaux acquis dans différents types d'environnement. Sa mise en œuvre a été réalisée par une programmation en langage C en vue d'être implantée sur un processeur dédié au traitement numérique du signal.

ENHANCEMENT OF SPEECH EMBEDDED IN CONVOLUTIVE AND ADDITIVE NOISE WITH WAVELET APPLICATION

Antoine Munoz

ABSTRACT

This work deals with the design and the simulation in view of a possible implementation on a digital signal processor of a speech enhancement system in presence of additive and convolutive noise using the wavelet transform

The methods which have been studied here are those which aim to reduce additive and convolutive the noise generated by microphones used to collect audio signals as well as the noise generated by the environment of these microphones in order to recover a signal as close as possible to the original one.

The wavelet transform is a powerful tool which has been used by a large number of researchers for the purpose mentioned above. The research project developed here consists mainly in applying different methods suitable for this type of denoising.

Different methods covering our subject of interest and reported in the literature have been studied. One of these methods has been chosen for its suitability to our purposes. It, consists of the estimation of the speech embedded in a reverberant and noisy environment by independent component analysis and wavelets. We have tested this approach by computer simulation with the Matlab software with the inclusion of some modifications and adaptations. We have then translated our algorithm in C language for an eventual implementation on an embedded DSP system for a real world utilization.

TABLES DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET SIGLES

ACI	analyse en composantes indépendantes
BSE	abréviation anglaise pour EAS; Blind sources extraction
BSS	abréviation anglaise pour SAS; Blind sources separation
CCS	Code Composer Studio; logiciel de développement pour DSP avec IDE
DSK	DSP starter kit ; ensemble matériel et logiciel de développement pour DSP
DSP	abréviation anglaise pour digital signal processing ou processor mais aussi densité spectrale de puissance , à nuancer selon le contexte
EAS	extraction aveugle de sources
FFT	abréviation anglaise pour fast Fourier transform : transformée de Fourier rapide
IDE	integrated development environment; logiciel incorporant plusieurs utilitaires nécessaires à la programmation d'unité programmable dans un cadre de travail sur ordinateur convivial
MFLOPS	abréviation anglaise pour million of floating-point operations per second
MOS	abréviation anglaise pour mean opinion score, échelle qualitative de la perception auditive
RIF	filtre numérique à réponse impulsionnelle finie
RSB	rapport de puissance du signal sur le bruit exprimé en décibel
SAS	séparation aveugle de sources
TFD	transformée de Fourier discrète
TOC	transformée en ondelette continue
TOD	transformée en ondelette continue dyadique

INTRODUCTION

Le rehaussement de la parole est un sujet de recherche très actif actuellement dans plusieurs domaines de l'activité humaine incluant entre autres, le scientifique, l'industriel et le médical. Le présent mémoire s'insère dans cette recherche en abordant la sélection d'une méthode originale venant de chercheurs de ce domaine où est mise à contribution la transformée en ondelettes.

La méthode choisie est développée avec quelques modifications sans toutefois déroger de ces principales fonctionnalités visées. Son ensemble est vérifié sur ses théories et principes par simulation dans un premier volet sous environnement Matlab.

Dans son second volet, les algorithmes bâtis sous simulation sont émigrés en langage C de programmation essentiellement dans le but de concrétiser sa mise en œuvre sur un processeur dédié au traitement numérique du signal.

Les résultats pratiques obtenus lors de la mise en œuvre en langage de programmation C pour DSP ainsi que ceux recueillis par la théorie sous simulation Matlab seront étudiés et comparés afin d'évaluer l'efficacité de la méthode choisie.

CHAPITRE 1

LE REHAUSEMENT DE LA PAROLE

Le rehaussement de la parole vise essentiellement l'intégrité et la préservation optimales de l'identité et des spécificités de la source sonore recherchée malgré ses altérations subies et provoquées par des éléments sonores perturbateurs qui lui sont étrangers mais présents dans son environnement.

Les applications où l'on retrouve la nécessité du rehaussement de la parole sont multiples et variées. À titre d'exemple, mentionnons les domaines des communications où la parole en général est mise en jeu dans sa reconnaissance ou son identification notamment dans les applications du multimédia, des radios, des communications commerciales, industrielles et militaires.

On peut également déduire que les solutions proposées pour le rehaussement de la parole peuvent s'exporter vers des domaines similaires mettant en jeu des signaux de nature sonore, comme dans le domaine médical, plus spécifiquement, l'échographie entraînant une plage fréquentielle du sonore plus étendue embrassant les ultrasons comme les infrasons.

Avant de présenter succinctement la nature de la parole, on doit définir la classification des signaux sonores perturbateurs. Ces sont généralement les signaux exclus du ou des signaux recherchés. Ils portent alors l'appellation de bruit qui se classifie selon l'effet perturbateur qu'il produit sur les ou le signal recherché, en l'occurrence ici, la parole.

L'environnement entourant la parole d'un locuteur peut générer des bruits non corrélés et additifs, ils ont donc un effet de superposition sur le signal de la parole comme c'est souvent le cas du bruit ambiant.

Le bruit convolutif est celui qui affecte le spectre du signal de la parole lorsqu'elle subit un filtrage linéaire.

Les causes sont principalement dues à la réverbération: son effet peut être modélisé par un filtre linéaire qui dépend de l'architecture, du contenu de la salle dont les surfaces réfléchissantes retransmettent le signal altéré dans ses composantes fréquentielles, et de la position du locuteur par rapport au microphone car plusieurs chemins se créent introduisant sur le signal recherché différents délais ou retards de ses copies introduisant ainsi des interférences.

Ensuite viennent les caractéristiques spectrales spécifiques d'un locuteur : la moyenne à long terme de la réponse fréquentielle du conduit vocal varie d'un locuteur à un autre à cause des différences physiologiques des conduits vocaux.

Finalement s'ajoute l'équipement d'enregistrement: si on utilise différents microphones, on peut constater des modifications tout au long du spectre. Ce changement dû au microphone peut être raisonnablement modélisé par un filtre linéaire.

Le bruit convolutif est une cause majeure de la dégradation des performances. Les mesures de distorsions spectrales utilisées pour la reconnaissance de la parole sont fortement affectées par ce type de bruit.

Si le modèle du canal de transmission peut être représenté par un filtre linéaire alors les signaux enregistrés peuvent être raisonnablement modélisés et l'implication des bruits additifs et convolutifs sera fondée sur ce modèle. Ainsi la représentation d'un signal observé y(t) affecté de ces bruits donne la relation suivante :

$$y(t) = s(t) * h(t) + n(t) \tag{1.1}$$

où s(t) est le signal de la parole d'origine sans distorsion, n(t) est le bruit additif et h(t) le bruit convolutif.

1.1 Les caractéristiques de la parole

La parole peut être considérée comme un signal sonore engendré par le système vocal humain faisant participer un ensemble d'organes constituant le conduit vocal. Il s'agit des muscles pulmonaires, des poumons, des bronches avec la trachée qui aboutissent aux cordes vocales, éléments vibratoires essentiels pour la génération des sons, ensuite le conduit se termine par un ensemble de cavités à volume variable dont certaines se caractérisent par une grande souplesse ou par une rigidité importante. Elles sont constituées du larynx, du pharynx, par la suite, le conduit se sépare au voile du palais pour aboutir sur les deux dernières cavités nasale et buccale. Cette dernière d'une grande souplesse est sans contredit la plus versatile. Elle joue aussi un rôle prépondérant dans la formation des sons par une mise en œuvre complexe de la coordination simultanée des muscles linguaux, maxillaires et buccaux. La figure 1 illustre les organes de la phonation.

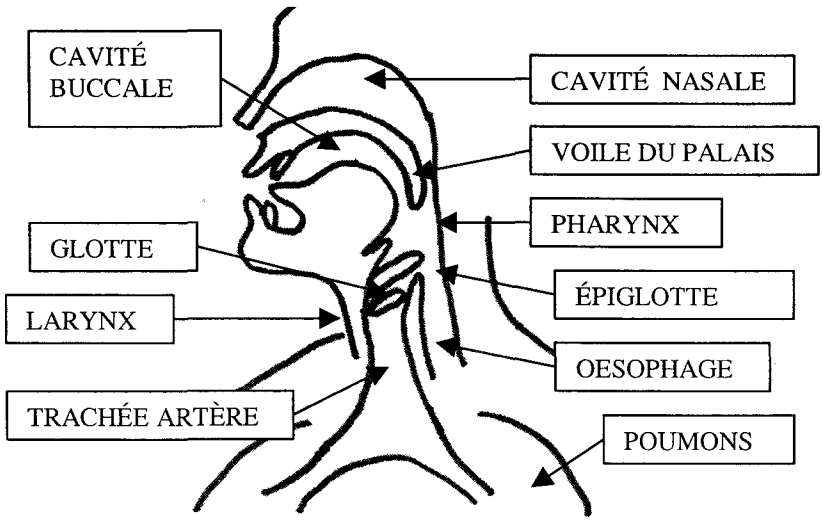

Figure 1 Les organes de la phonation selon [1]

Les sons représentant la parole sont considérés selon le point de vue du traitement du signal comme étant globalement un processus aléatoire. Mais si on fait une analyse segmentée sur le signal de la parole, on retrouve généralement un aspect pseudo-stationnaire ou quasi-périodique qui est caractéristique lorsqu'elle n'est pas parsemée d'éléments purement aléatoires.

Ce sont ces dernières considérations qui ont amené à identifier deux classes de sons issus de la parole. Il s'agit des sons voisés et non voisés.

La première classe à forte périodicité, englobe l'ensemble des voyelles plus quelques consonnes et sont issues principalement des vibrations des cordes vocales, donnant une fréquence fondamentale de la voix communément appelée pitch, plus les harmoniques de celle-ci engendrées par les cavités en aval des cordes vocales par effet de résonance, ces dernières composantes périodiques sont appelées les formants de la voix.

La seconde classe est remarquable par son absence totale de périodicité et s'apparente aisément à la nature aléatoire du bruit blanc, et se situe sur la partie haute du spectre vocal.

La voix présente une portée fréquentielle pouvant aller autour de 6 octaves en étendue. Cependant parmi les signaux vocaux, le ton de la voix est situé dans une bande restreinte du spectre vocal humain où ses limites sont fixées sur différentes valeurs selon les chercheurs rencontrés. À titre d'exemple, selon [2] le ton de la voix ou sa fréquence fondamentale f0 se situe dans les fréquences basses du spectre de la voix parlée, c'est-à-dire généralement entre 100 et 150 Hz pour une voix masculine, entre 200 et 300 Hz pour une voix féminine, et pour une voix enfantine, elle se situe entre 300 et 450 Hz.

Après avoir décrit brièvement la nature des signaux de la parole, nous allons passer en revue les techniques usuelles employées pour rehausser celle-ci en présence du bruit.

1.2 Les techniques de rehaussement de signal

En général, la majorité des méthodes de rehaussement traite principalement le bruit additif et plus rarement celui de convolution dont l'élimination ou la réduction s'avère être une tâche plus élaborée. Mais avant d'aborder le bruit convolutif, nous allons résumer les méthodes qui éliminent le bruit additif puisque celui-ci est toujours présent en pratique. La référence [3] présente les principales méthodes de rehaussement de la parole que nous allons résumer ici dans leurs grandes lignes.

La première méthode opère dans le domaine spectral. Le signal temporel observé est projeté dans le domaine fréquentiel par transformée de Fourier (en général avec la FFT) pour déterminer la puissance à laquelle on soustrait la densité spectrale de puissance du bruit mesurée ou connue lors des moments où la parole est absente. Le résultat est remis dans le domaine temporel par la transformée de Fourier inverse qui nous donne le signal rehaussé et nettoyé du bruit additif.

La seconde méthode opère tant dans le domaine temporel que fréquentiel, et emploie intensivement le filtrage optimal adaptatif qui ajuste les coefficients de sa réponse impulsionnelle de telle sorte que l'on obtienne en sortie le signal rehaussé à partir du signal observé en entrée du filtre. Pour se faire, le procédé nécessite la connaissance statistique de l'estimation du signal observé comme celle du bruit et s'appuie généralement sur le critère de l'erreur moyenne quadratique minimale de l'estimation.

La troisième méthode est l'annulation adaptative du bruit. Elle est également basée principalement sur le filtrage adaptatif qui nécessite souvent à son entrée un signal de référence qui dans notre cas est le bruit environnant non corrélé au signal recherché, nous obtenons alors à sa sortie une estimation du bruit qui est soustraite ou comparée au signal observé prélevé sur le deuxième canal. Le résultat donne le signal rehaussé. Cette méthode se base également souvent sur le critère de l'erreur moyenne quadratique

minimale de l'estimation et emploie un grand nombre d'algorithmes qui y sont associés. Elle s'opère le plus souvent dans le domaine temporel.

La quatrième et dernière des principales méthodes s'appuie sur les caractéristiques fréquentielles de la voix et à sa nature périodique pour éliminer sinon réduire le bruit perturbateur. Cette méthode contrairement aux autres mentionnées précédemment, s'affranchit de la nécessité d'une connaissance à priori de l'environnement bruité. Elle dépend essentiellement de la détermination de la fréquence fondamentale ou pitch de la voix étudiée. Dès que cette information est acquise, on procède différemment selon le choix du domaine emprunté, temporel ou fréquentiel.

Dans le domaine temporel, on introduit, après un délai égal à la période fondamentale, le signal observé dans un filtre adaptatif qui ajuste ses coefficients pour obtenir le signal estimé qui est comparé à une période près, le résultat est le bruit rejeté et constitue l'erreur estimée qui vient modifier les coefficients selon le critère de l'erreur moyenne quadratique minimale de l'estimation.

Dans le domaine fréquentiel, on emploie un filtre adaptatif en peigne qui va rechercher la fréquence fondamentale et ses harmoniques tout en éliminant le restant qui constitue essentiellement la partie bruitée.

Dans le travail présenté ici, nous utilisons certains éléments issus des méthodes qui viennent d'être mentionnées.

Nous avons retenu intentionnellement le bruit convolutif pour la fin, car les méthodes élaborées pour le contrer n'existent que depuis une décennie pour les signaux à large bande et depuis environ deux décennies pour les signaux à bande étroite.

Les références [4] et [5] traitent du sujet en abordant les mélanges convolutifs. Les méthodes décrites sont complexes et exigent souvent un calcul intensif et laborieux souvent associés aux statistiques d'ordre élevé.

Ce que nous retenons ici, c'est qu'il est admis [5] qu'un mélange convolutif peut s'approcher d'un mélange instantané lorsque les signaux considérés sont à bande étroite. C'est cette dernière caractéristique qui est mise à profit dans la solution retenue que nous verrons plus loin en détail.

1.3 La détermination de la fréquence fondamentale

La fréquence fondamentale f0 est une caractéristique essentielle dans notre application choisie, aussi les approches les plus classiques de la détermination de la fréquence fondamentale sont maintenant rappelées avant de présenter la méthode employant la transformée en ondelettes qui sera d'ailleurs expliquée plus en détail au chapitre 3.

Ces approches sont définies en deux classes; la première classe opère dans le domaine temporel, dans ce cas on cherche à estimer l'intervalle entre deux instants consécutifs ou survient la fermeture glottique de la parole. La seconde classe exploite la stationnarité segmentée du signal de la parole pour relever sa périodicité et emploie alors l'analyse spectrale fenêtrée, la recherche de la fréquence fondamentale dans ce cas, s'opère dans le domaine fréquentiel.

Mentionnons les quelques principales méthodes utilisées dont fait mention la référence [6]:

- Méthode basée sur l'analyse spectrale
- Méthode basée sur l'autocorrélation
- Méthode du Cepstrale
- Méthode du LPC (linear predictive coding)

- Méthode du SIFT (simplified inverse filtering technique)
- Méthode du AMDF (average magnitude difference function)

Parmi cette dernière liste nous montrerons quelques exemples à savoir les méthodes de la spectrale , de la corrélation ainsi que de la cepstrale et de l'AMDF.

La détermination de la fréquence fondamentale f0 peut se faire au moyen d'une analyse spectrale segmentée par transformée de Fourier sur le signal de la parole à analyser. La partie spectrale qui nous intéresse se situe en bas de 600 Hz. La f0 est la plus petite fréquence avec un maximum local parmi le contenu de ses harmoniques ayant des maxima locaux.

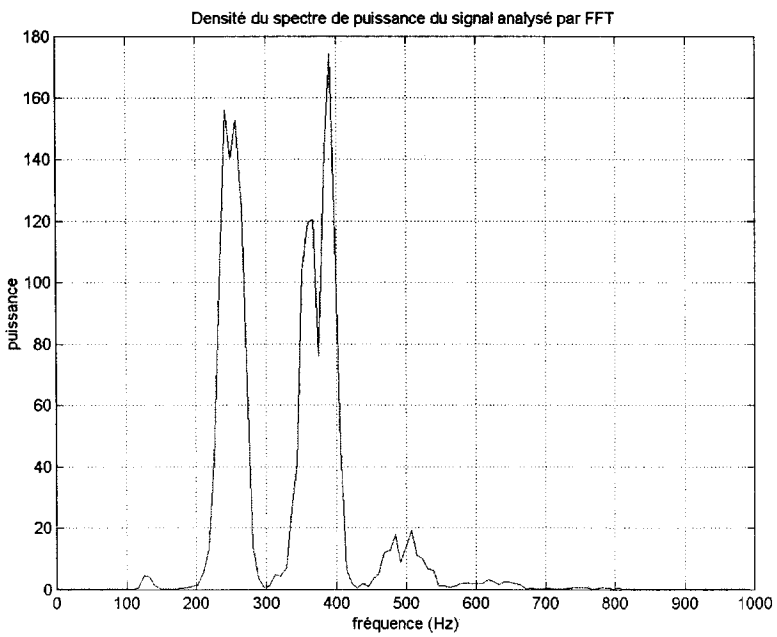

Figure 2 Détermination de la f0 par FFT

La figure 2 illustre un exemple avec le mot anglophone « we » où on peut voir le premier maximum (la f0) aux environs de 130 Hz avec d'autres maxima plus importants

de ses harmoniques qui sont définis comme les formants de la voix et se situent aux environs de 260 Hz (2f0), 390 Hz (3f0) et 520 Hz (4f0).

Il est important de remarquer l'importance de la résolution fréquentielle. Dans notre exemple elle est fixée à (8000/1024) Hz, soit le rapport de la fréquence d'échantillonnage sur le nombre d'échantillons traités par la fenêtre ou le segment analysé. Si cette résolution est trop grande, la f0 peut ne pas apparaître dans certains cas comme ici dont le maximum est relativement petit et on peut alors la confondre avec une de ses harmoniques aux maxima plus importants.

La seconde méthode repose sur le principe de la corrélation du signal à analyser avec une version de lui-même qui est plus ou moins retardée, il s'agit plus précisément de l'autocorrélation. Elle possède une propriété qui est ici mise à profit ; tout signal à caractère périodique va donner une autocorrélation également périodique. Le temps séparant les maxima consécutifs de l'autocorrélation donne la période du signal analysé.

La figure 3 montre un exemple d'autocorrélation avec le même signal analysé précédemment et illustré sur la figure 3a. La figure 3b donne le résultat de l'autocorrélation, la figure 3c est sa copie avec les premiers échantillons retirés qui ne tient pas compte du premier maximum autour de zéro de l'abscisse. La figure 3d donne le calcul de la f0 sur les deux premiers maxima consécutifs situés autour du $60^{ième}$ et $125^{ième}$ échantillons donnant respectivement comme f0, 130 Hz et 125 Hz.

On remarque un troisième maximum donnant une fréquence supérieure à 1 KHz en dehors du domaine spectrale de la fréquence fondamentale humaine. Plus on va dans la partie rétardée du résultat de l'autocorrélation de la figure 3c, plus les maxima diminuent et se confondent aux maxima des contributions bruités du signal. Il est donc important de fixer judicieusement un seuil au dessus duquel on considère les maxima retenu pour fin du calcul de la f0. En général le premier maximum fixe la valeur de la f0.

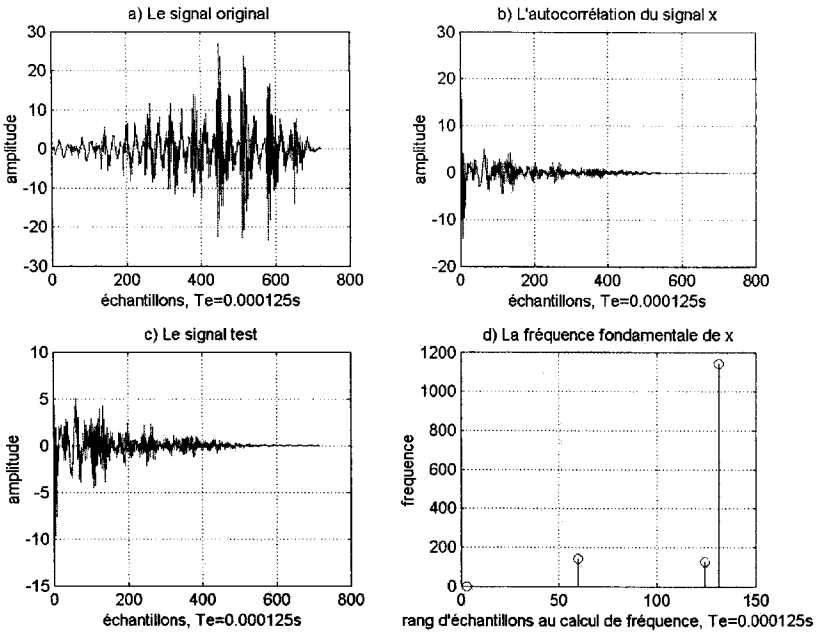

Figure 3 Détermination de la f0 par autocorrélation

La méthode de l'AMDF qui est le sigle anglais pour Average Magnitude difference function est une méthode dérivée de l'autocorrélation, elle est aussi définie comme l'autocorrélation par différence. Contrairement à l'autocorrélation, on va chercher les positions et les écarts entre les minima. Comme l'autocorrélation, c'est principalement le premier minimum qui fixe la période de la f0. La figure 4 donne les résultats de cette méthode.

La figure 4a représente le signal analysé. La figure 4b donne en graphique le résultat de l'AMDF le premier minimum après l'origine en abscisse se trouve autour du 65^{ième} échantillon, son calcul est montré sur la figure 4c où la f0 est déduite à 123 Hz. Ensuite les autres minima donnent respectivement des f0 de 133 Hz, 100 Hz, et 195 Hz.

Le dernier minimum sur la figure 4c est sujet aux mêmes contraintes vues pour l'autocorrélation, dès qu'on s'éloigne de l'origine de l'abscisse les minima ou les

maxima pour l'autocorrélation sont difficiles à bien déceler ou isoler des contributions bruités du signal.

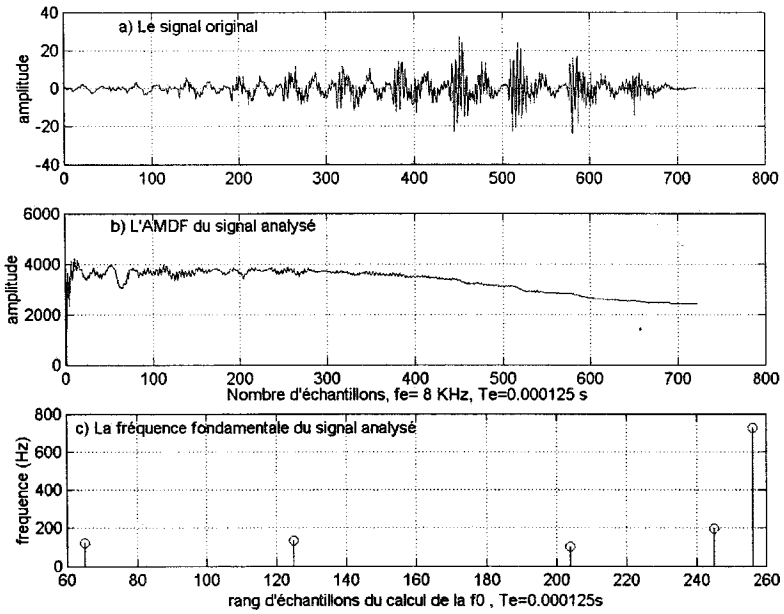

Figure 4 Détermination de la f0 par l'AMDF

La méthode cepstrale selon [7] est dérivée de la méthode d'analyse spectrale par transformée de Fourier segmentée. Il s'agit de faire la transformée de Fourier sur un signal segmenté. On calcule ensuite le logarithme de ce dernier résultat puis on lui applique la transformée de Fourier inverse pour revenir dans le domaine temporel. La figure 5 illustre l'analyse du signal par la méthode cepstrale pour déterminer la f0.

La figure 5a représente le signal analysé dans le domaine temporel. La figure 5b est la transformée de Fourier du signal dans le domaine fréquentiel. La figure 5c est l'application du logarithme sur la figure 5b dans le domaine fréquentiel. Finalement la

figure 5d est le résultat de la transformée inverse de Fourier sur le graphique de la figure 5c. Cette dernière opération nous a ramené dans le domaine temporel.

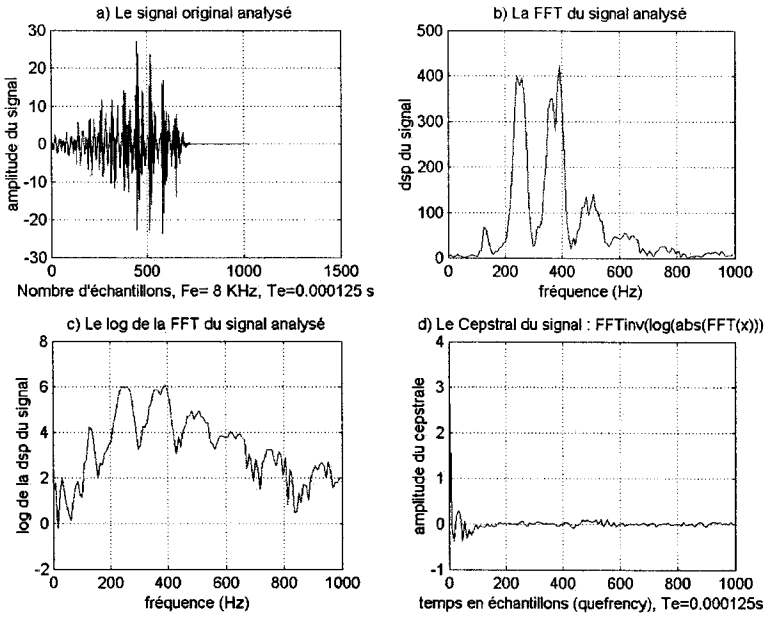

Figure 5 Détermination de la f0 par cepstrale

L'abscisse (en ms) appelée quefrence de la figure 5d est divisée en deux sous-domaine temporel où la frontière se situe autour des 2 ms. Cette frontière se traduit en terme d'échantillon autour du $16^{ième}$. La zone inférieure à cette limite constitue pour la voix la contribution du conduit vocal autrement dit les formants ou la partie fréquentielle au delà de 500 Hz (1/2 ms). La zone supérieure à cette limite constitue la contribution de l'onde glottique, autrement dit les fréquences en bas de 500 Hz, où la fréquence fondamentale de f0 se retrouve.

Le premier maximum que l'on observe dans la zone supérieure nous donne la f0 étant l'inverse de la valeur temporelle où se trouve ce maximum. Dans notre cas nous

devrions observer un maximum autour du $65^{ième}$ échantillon. On constate dans le cas présent que cette méthode a de la difficulté à détecter la f0.

Nous verrons plus loin, au chapitre 3 la transformée en ondelettes utilisée pour la détection de la f0 de notre application qui traitera également le signal étudié ici. De plus d'autres comparaisons avec les méthodes classiques seront étudiées et présentées par d'autres auteurs.

CHAPITRE 2

LA TRANSFORMÉE EN ONDELETTES CONTINUE

Il existe plusieurs méthodes basées sur les analyses spectrales pour étudier la nature évolutive sur le temps ou l'espace de différents phénomènes physiques, chimiques et biologiques.

La méthode la plus répandue est l'analyse de Fourier qui emploie les propriétés de la transformée de Fourier pour fournir l'information fréquentielle et le déphasage d'un signal variant dans l'espace ou le temps. Cette méthode bien que largement utilisée dans différents domaines scientifiques et industriels a montré certaines limitations dans un nombre d'applications où les signaux en cause s'écartaient plus ou moins de la nature du signal stationnaire et prédictif propice à l'analyse de Fourier.

Lorsque l'étude d'un phénomène implique la mesure de signaux quasi stationnaires ou non stationnaires dont les variations temporelles ou spatiales sont à fortes discontinuités, la transformation de Fourier est remplacée par celle à fenêtre glissante. S'il s'agit de signaux quasi stationnaires à discontinuités ponctuelles alors la transformée en ondelettes devient plus appropriée pour ces types de signaux.

La transformation en ondelettes, présentée par Mallat [8] , Goswami et Chan [9], permet un ensemble très diversifié de méthodes que ce rapport ne pourra aborder entièrement. Cependant il est utile dès à présent de mentionner la distinction fondamentale de principe qui existe entre les transformations en ondelettes et de Fourier.

La transformation de Fourier permet l'analyse et la synthèse d'un signal à partir de ses composantes élémentaires que sont les sinusoïdes modulées en fréquences et en phase.

La transformation en ondelettes permet d'analyser et de synthétiser un signal à partir de composantes élémentaires que sont les ondelettes. Ces dernières sont généralement des

signaux oscillatoires d'énergie finie et de durée également finie dont l'intégrale est nulle. Les ondelettes sont caractérisées par deux paramètres. L'un est associé à l'échelle ou à l'étendue temporelle de l'ondelette, l'autre est associé à la translation temporelle (ou spatiale) de l'ondelette par rapport à un moment donné.

2.1 La transformée de Fourier à fenêtre glissante (TFFG)

Lorsqu'on veut localiser ou détecter un changement dans un signal, on a recours à l'analyse de la transformée de Fourier à fenêtre temporelle glissante (TFFG). L'idée de base de la TFFG est d'utiliser une fonction fenêtre $\varphi(t)$ de telle sorte que les largeurs des fenêtres tant temporelles $\Delta\varphi(t)$ que fréquentielles $\Delta F(\varphi(t))$ seront bornées ou à support compact. Cela permet d'obtenir la réponse fréquentielle à un temps donné. Le principe d'incertitude d'Heisenberg limite la borne inférieure du produit de la largeur des fenêtres temporelles et fréquentielles; il doit être supérieur à ½. Le produit sera égal à ½ seulement si la fonction fenêtre est gaussienne. Un des exemples de la TFFG est la transformée de Gabor qui utilise une fonction gaussienne. La fenêtre d'analyse étant fixe, ce procédé ne peut localiser (dans le temps) toutes les gammes de fréquences de façon égale.

2.2 Transformée en ondelettes continue (TOC)

Comme dans le cas de la TFFG, une des caractéristiques maîtresses de la transformée en ondelettes continue est de pouvoir localiser un phénomène transitoire ou un changement brusque dans un signal. Contrairement à la TFFG, la fenêtre d'analyse s'allonge s'il s'agit de localiser les basses fréquences et se rétrécit pour résoudre les hautes fréquences. L'ondelette est une fonction localisée dans le temps et contenant des oscillations. Une ondelette ne peut être choisie arbitrairement, car après la transformation, il faut pouvoir reconstituer le signal. Il faut aussi que les fonctions d'intérêt puissent être représentées par une combinaison d'ondelettes dilatées et décalées. La condition suffisante d'admissibilité est que la moyenne temporelle d'une ondelette soit nulle et d'énergie finie (et supérieure à 0). Cela signifie que le spectre

d'une ondelette est similaire à un filtre passe haut puisque la moyenne temporelle est nulle. La transformée continue en ondelettes est donnée par:

$$W_{\psi}f(b,a) = \int_{-\infty}^{\infty} f(t)\overline{\psi_{b,a}(t)}dt \tag{2.1}$$

où l'ondelette s'exprime par :

$$\psi_{b,a}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{2.2}$$

Remarquons la présence du trait haut signifiant le complexe conjugué, car certains types d'ondelettes sont complexes. Si on substitue l'équation 2.2 dans l'équation 2.1, on constate qu'il s'agit d'une corrélation du signal $f(t)$ avec la fonction $\Psi(t)$, celle-ci étant faite à l'échelle a. On peut aussi dire que l'équation 2.1 représente la convolution de $f(t)$ avec la fonction $\Psi_{b,a}(t)$ involuée, c'est à dire $\Psi_{b,a}(-t)$.

Les paramètres b et a sont des paramètres de translation et de dilatation (ou contraction) respectivement. La transformation engendre donc une fonction à deux variables. En fait, on peut considérer que la transformation en ondelettes continue est constituée par les sorties d'une suite continue de filtres empilés les uns sur les autres sur l'axe vertical des échelles alors que le temps est sur l'axe horizontal du plan. Chacun des filtres est la transformée de Fourier de l'ondelette à l'échelle appropriée selon notre position dans l'axe.

Dans la pratique, la convolution du signal avec l'ondelette peut être effectuée à l'aide d'une transformée de Fourier discrète (TFD) du signal, suivie d'une multiplication par la TFD de l'ondelette discrétisée, et suivie finalement par une transformée de Fourier

inverse de ce produit. Ces opérations sont faites pour chaque échelle *a*. La transformée de Fourier inverse selon les auteurs Goswami et Chan [9] étant :

$$W_n(a) = \sum_{k=0}^{N-1} F_k \psi(aw_k) e^{i2\pi kn/N} \tag{2.3}$$

Où la DFT du signal est :

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-i2\pi kn/N} \tag{2.4}$$

Pour assurer une comparaison des transformées entre les échelles, il y a une normalisation préalable de l'ondelette pour obtenir une énergie unitaire à chaque échelle.

Le résultat de la transformée en ondelettes est un coefficient d'ondelette à une échelle *a* et à une translation *n* données. Les valeurs des coefficients représentent la contribution relative d'une ondelette avec une contraction donnée et une translation donnée à la composition du signal.

La transformée en ondelettes inverse est mathématiquement donnée par :

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} db \int_{-\infty}^{\infty} \frac{1}{a^2} \left[W_\psi f(b,a) \right] \psi_{b,a}(t) da \tag{2.5}$$

Poursuivant l'analogie avec les bancs de filtres, on peut considérer la transformée inverse conne étant une banque de filtres superposés sur l'axe vertical et permettant de récupérer le signal d'origine si le produit du filtre de transformation avec celui de synthèse donne 1.

Plus l'ondelette est contractée dans le domaine du temps plus son spectre est dilaté dans celui de la fréquence. L'avantage principal de la transformée en ondelettes continue est d'obtenir un produit de largeur des fenêtres (temps * fréquence) constant. Cela permet de localiser un changement si, à un instant donné, on considère la valeur des coefficients d'ondelettes sur l'axe vertical. L'ondelette temporelle se contracte au fur et à mesure que l'on remonte l'axe vertical, ce qui permet une localisation adéquate de toutes les fréquences à tout instant.

L'interprétation d'une transformée en ondelettes continue est différente de celle donnée par la TFFG. Puisque la transformée de Fourier d'une ondelette est équivalente à un filtre passe-haut, une sinusoïde pure sera représentée par une bande plus ou moins large selon la largeur de la fenêtre fréquentielle, au lieu d'une seule ligne comme dans la TFFG. De plus l'axe vertical n'est pas un axe de fréquence mais plutôt un axe d'échelle. Il est possible de relier l'échelle et la fréquence si on connaît le type d'ondelette utilisée, donc son spectre. En outre, la localisation d'une discontinuité temporelle, qui contient toutes les fréquences, sera moins bonne aux basses fréquences, la fenêtre temporelle étant plus large.

La caractérisation de signaux avec cette méthode peut se faire de deux manières. L'expérience de l'usager lors de l'examen du plan de la transformation, ou des tests statistiques sur la puissance des coefficients par rapport à ceux obtenus avec du bruit blanc. Bien que cette approche soit possible, nous centrons notre propos sur la décomposition du signal dans le cadre de ce projet.

CHAPITRE 3

LA MÉTHODE DU REHAUSEMENT DE LA PAROLE

La recherche sur le projet s'est basée sur le document en référence [10] des auteurs Barros, Rutkowski, Itakura et Ohnishi dont le titre est : « Estimation of speech embedded in a reverberant and noisy environment ». Dans ce document on montre un système pour rehausser la parole représentant la plus grande énergie baignée dans un environnement de bruits additifs et convolutifs où s'ajoutent d'autres sources vocales ou d'autres perturbations. Le système illustré à la figure 6, est décrit par 4 blocs de processus dont les fonctions principales sont les suivantes :

- Deux microphones distancés l'un de l'autre, captent l'ensemble des signaux présents. Il y a donc deux canaux dont les signaux à traiter sont $X_1(t)$ et $X_2(t)$.
- Le bloc SIF (speech instantaneous frequency) permet à partir des signaux $X_1(t)$ et $X_2(t)$ de déterminer le ton de la voix (pitch) ou encore la fréquence fondamentale f0 de la parole qui sera nécessaire pour les autres blocs en aval. La méthode se base sur un algorithme utilisant une analyse spectrale et un filtre avec une ondelette de type Gabor pour estimer la fréquence f0 en employant la transformée de Hilbert pour déduire la fréquence instantanée de la parole.
- Un banc de filtres adaptatifs de type passe-bande centrés sur un nombre fini d'harmoniques de la f0 décomposent en sous-bandes les signaux venant des deux canaux. Le traitement du signal travaillé ainsi en bandes étroites permet de réduire voir supprimer les effets convolutifs inhérents à ce genre d'application. Chaque sous-bande en sortie d'un passe-bande aura un signal $r_{i,k}(t)$ où l'indice i et k sont respectivement le canal et la sous-bande associée à la fréquence fondamentale f0 ou à une de ses harmoniques kf0.
- Le bloc ACI (Analyse en Composantes Indépendantes) est constitué de plusieurs fonctions. À partir de chaque $r_{i,k}(t)$ une transformée de Hilbert est faite pour obtenir l'enveloppe du signal $\hat{A}_{i,k}(t)$ qui va permettre de moduler une pure harmonique à la fréquence de f0 ou kf0 selon la sous-bande. Ce signal synthétisé

$Z_{i,k}(t)$ doit se substituer à $r_{i,k}(t)$ et pour se faire, il doit recouvrer la phase de ce dernier au moyen d'un filtre adaptatif de Wiener. Nous obtenons alors $x_{i,k}(t)$ un signal synthétisé avec la correction de phase remplaçant $r_{i,k}(t)$. Pour chaque sous-bande, un traitement statistique et adaptatif va chercher les éléments indépendants pour ne garder que ce qui sont liés à f0 ou à kf0 donc les plus susceptibles d'être les constituants de la parole représentée par f0 ou kf0. On reconstitue de la sorte la parole recherchée épurée des éléments étrangers en sommant la contributions de toutes les sous-bandes des deux canaux.

- Le dernier bloc agit principalement en tant qu'algorithme de coordination entre les différents blocs mentionnés en maintenant à jour les éléments psycho-accoustiques.

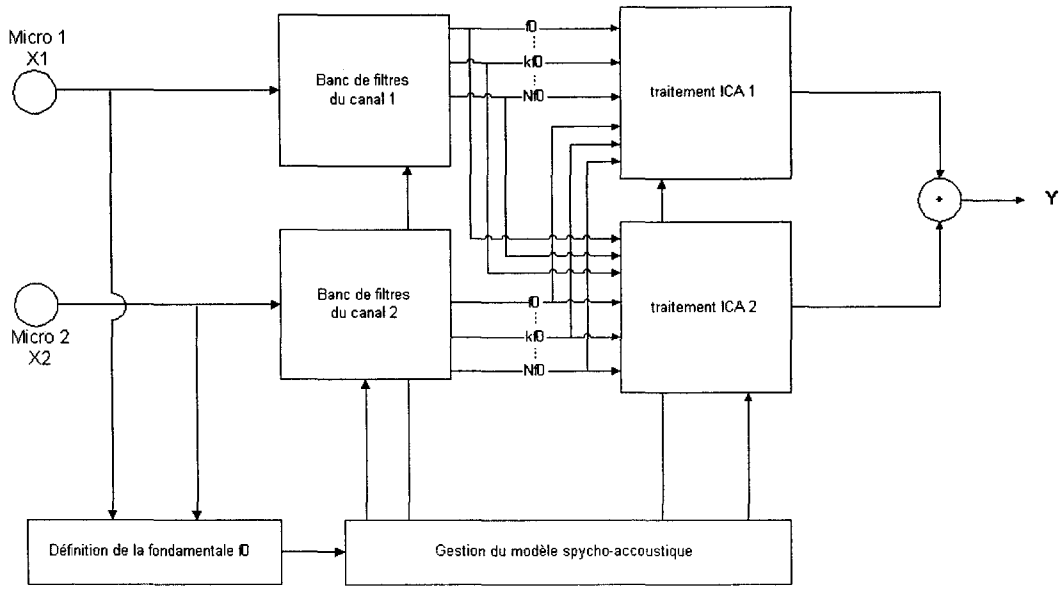

Figure 6 Schéma synoptique du rehaussement de la voix selon [10]

Le document [10] bien que présentant les grandes lignes du projet, reste toutefois très succinct sur les détails de sa réalisation et quelquefois certains éléments sont totalement absents des publications. Nous avons donc choisi de partir sur les bases du système

suggéré tout en adoptant des solutions originales aux niveaux des différents blocs fonctionnels lorsque la situation l'exigeait.

3.1 La détermination de la fréquence fondamentale par ondelettes

L'algorithme de détection et de détermination de la fréquence fondamentale f0 de la parole que nous avons privilégié, suite aux études précédentes, est basé sur l'utilisation des ondelettes.

Les méthodes faisant appel aux propriétés des ondelettes s'appuient sur le fait que la parole est un signal non stationnaire quasi-périodique avec d'importantes discontinuités. Ce genre de signal est réputé plus propice à l'analyse aux moyens d'ondelettes qui possèdent un comportement plus robuste vis-à-vis du bruit additif et convolutif. Parmi les méthodes présentées ici, nous avons retenu certains de leurs éléments pour mettre en œuvre notre méthode qui détermine la f0. Nous allons maintenant détailler leurs principaux éléments.

Nous sommes partis de la référence [11] des auteurs Kadambe et Boudreaux-Bartels qui emploient une transformée en ondelettes de type dyadique (TOD) car son facteur d'échelle est une grandeur à la base deux qu'on peut ajuster en intervenant sur sa puissance appartenant aux entiers relatifs (voir équation 3.1). Cette ondelette est selon les auteurs [11] propice pour les f0 hautes et basses et présente une robustesse au bruit. Elle est proposée avec une performance considérée supérieure aux méthodes classiques basées sur les méthodes de l'autocorrélation et de la cepstrale.

La durée variable de la période de f0, soit 1/f0 peut se située entre 1,35 ms et 40 ms et dépend largement de facteurs psychologiques et physiologiques du locuteur. Une représentation temps-échelle de la transformée en ondelettes de type dyadique (TOD) devra localiser la fermeture glottique et déduire l'intervalle de cet événement répétitif qui définie la période de f0. La TOD du signal x(t) est définie par :

$$TOD_x\left(b,2^j\right)=\frac{1}{2^j}\int_{-\infty}^{+\infty}x(t)\,g^*\left(\frac{t-b}{2^j}\right)dt=x(t)*g^*_{2^j}(t) \tag{3.1}$$

et se calcule en employant un facteur d'échelle a = 2^j qui est discrétisé sur la séquence dyadique. Ensuite, la fonction d'ondelette complexe conjuguée *g*(t)* doit satisfaire les conditions mentionnées ci-après. Cette fonction s'exprime comme :

$$g_{2^j}(t) = \frac{1}{2^j}\, g\left(\frac{t}{2^j}\right) \tag{3.2}$$

D'un point de vue du traitement de signal, la TOD peut être vue comme le résultat de la sortie de bancs de filtres multibandes composés de passe-bandes en octave dont la réponse impulsionnelle est la fonction d'ondelette. La largeur de bande et la fréquence centrale de chaque passe bande sont proportionnelles à $1/2^j$. Chaque échelle ainsi correspond à des bancs de fréquences qui permet de ressortir le contenu fréquentiel du signal aussi les fonctions d'ondelettes peuvent être assimiler à des filtres passe-bandes dans le domaine fréquentiel.

La TOD a comme propriétés la linéarité et l'invariance sur sa translation temporelle que partagent également les signaux vocaux dont la modélisation est souvent une combinaison linéaire avec une translation temporelle de sinusoïdes amorties. Si le signal x(t) ou ses dérivées possèdent des discontinuités alors la TOD de x(t) affichera en module des maxima locaux autour de ces discontinuités. Cette particularité prouvera son utilité dans la détection de la f0 puisque la fermeture glottique correspond à une variation brusque du débit de l'air et fait ressortir le caractère transitoire du signal de la parole.

Mallat qui est cité par les auteurs de la référence [11] a montré que dans notre genre d'application le choix du type d'ondelette est important. Si l'on choisit une fonction

d'ondelette g(t) qui est la première dérivée d'une fonction lisse c'est à dire dont la transformée de Fourier possède une énergie principalement concentrée sur la région des basses fréquences alors la TOD montrera des maxima locaux sur les variations brusques du signal et des minima locaux pour des variations lentes du même signal. De plus, on remarque lorsque survient une très forte variation du signal au moment précis t0, la TOD donne un maximum local et ce pour plusieurs échelles dyadiques consécutives à ce même t0. Cette dernière observation permet d'élaborer des algorithmes qui mettront en corrélation les maxima locaux de la TOD sur plusieurs échelles consécutives.

3.2 L'algorithme de la TOD pour la détection de la fondamentale f0

La TOD sur un segment du signal de la parole étudiée d'une longueur L ms est calculée à l'échelle $a = 2^i$, où i = ib, ib +1, ..., ih. Ici ib et ih sont respectivement les limites basse et haute que peut prendre i.

Pour chaque échelle 2^i, on trouve les maxima locaux par rapport à b de la TOD $(b, 2^i)$ qui dépasse un seuil donné T, ici son niveau correspond à 80% du maximum global de la TOD. Si la localisation des maxima locaux coïncide sur au moins deux échelles alors on suppose qu'il s'agit d'un moment de fermeture glottique. Par la suite nous estimons la période de la fréquence fondamentale f0 de la parole en mesurant l'intervalle temporel entre de tels maxima locaux consécutifs.

En théorie le nombre d'échelle est infini, par contre pour réduire le traitement du calcul nous limiterons ce nombre à une grandeur adéquate pour l'obtention de la TOD en se basant essentiellement sur la nature de la voix, également de durée finie. Dans ce contexte le nombre d'échelle dyadique suffisant [11] pour estimer la période de f0 à partir de la TOD est fixé à trois. Les échelles vont de 2^3 à 2^5 et sachant que l'échelle est inversement proportionnelle à la fréquence, les échelles choisies des ondelettes doivent couvrir le domaine spectral contenant la plage des fréquences fondamentales possibles.

La plus grande échelle tient compte de la volonté de retenir la bande des basses fréquences où se situe la f0. Ainsi nous calculons la TOD en commençant par l'échelle inférieure et nous doublons à chaque itération jusqu'à l'obtention de l'échelle supérieure. Évidemment le fait d'avoir uniquement trois échelles réduit la complexité du calcul de la TOD.

Les auteurs Kadambe et Boudreaux-Bartels [11] ont comparé les performances entre différentes méthodes d'estimation de la f0. La méthode d'estimation de la période de la f0 employant la TOD a été comparée aux méthodes les plus courantes employant l'analyse cepstrale et l'autocorrélation du signal.
La méthode d'analyse cepstrale d'un signal périodique aboutit à la même périodicité que le signal considéré. Le cepstral d'un signal x(t) est défini selon [11] comme :

$$c_x(t)=\left[\int_0^{\infty} \log\left|X(\omega)\right|^2 \cos(\omega t)\, d\omega\right]^2 \tag{3.3}$$

Quant à l'autocorrélation Rx(τ) du signal x(t), elle est définie par :

$$R_x(\tau)=\int_{-\infty}^{+\infty} x^*(t)\, x(t+\tau)dt \tag{3.4}$$

L'autocorrélation d'un signal périodique donne également une périodicité identique à celle du signal considéré.

Les deux méthodes déduisent la f0 en mesurant la durée entre les maxima consécutifs qui donne la période de la f0. C'est en général le premier maximum après celui placé à l'origine de l'abscisse qui donne la valeur prépondérante de la f0.

La comparaison s'est fait sur la précision de l'estimation de la période de la f0, sur la robustesse du traitement pour un signal dans un environnement bruité, sur la complexité

du calcul exigée par l'algorithme et finalement sur la facilité de choisir différentes segmentations du signal de la parole pour accomplir le traitement.

Lorsque f0 appartient au domaine des basses fréquences, les trois méthodes s'équivalent dans la précision de l'estimation de la période du ton de la voix. On constate lors des variations en sauts discontinus de la période de la fondamentale que la TOD donne un excellent résultat alors que les autres méthodes présentent leur pire performance. Cela est dû au fait que ces dernières méthodes font l'hypothèse de la stationnarité du signal dans la fenêtre d'analyse pour fixer la valeur de f0 et par conséquent donne une moyenne de la période estimée dans un segment du signal où les non-stationnarités sont noyées.

Toujours selon les auteurs Kadambe et Boudreaux-Bartels [11], lorsque le signal analysé est bruité avec des rapports signal/bruit (RSB) allant de 0 dB à –18 dB, c'est la TOD qui présente la meilleure précision (erreur relative de $\leq 2\%$ pour un RSB de –18 dB) sur l'estimation de la période. Les autres méthodes affichent leur meilleure performance pour un RSB à 0 dB avec des erreurs relatives de 17% et 52% respectivement pour la méthode du cepstrum et de l'autocorrélation, lorsque le RSB atteint –18 dB les résultats se dégradent allant jusqu'à 77 % et 80 %.

3.3 La complexité du calcul dans les méthodes abordées

La méthode la moins exigeante en terme d'opérations et par conséquent la plus rapide d'exécution est l'autocorrélation, ensuite vient après la TOD. Ces deux méthodes impliquent principalement des sommes de produits. La plus complexe et exigeante en terme d'opérations, est la méthode de la cepstrale car elle demande une série d'opérations élaborées dont une transformée de Fourier rapide suivie du calcul du logarithme de la puissance spectrale pour finalement se terminer par une transformée de Fourier inverse.

3.4 Le choix de la segmentation du signal de la parole

La segmentation ou le fenêtrage du signal de la parole analysée dans les méthodes observées doit être considéré avec précaution. Selon le choix de la méthode, plus particulièrement dans l'autocorrélation et la cepstrale, son importance est majeure. Ces deux méthodes font l'estimation moyenne de la période de f0 sur un segment de longueur L échantillons et pour cela il leur faut au moins détecter deux principaux événements pour pouvoir définir une période sur une segmentation donnée. Si la segmentation est trop courte, l'algorithme ne pourra évaluer avec précision la période et si la segmentation est trop grande les non-stationnarités du signal seront camouflées par une moyenne des événements périodiques évolutifs.

Cependant dans le cas de la TOD la longueur de la segmentation est moins importante, car elle procède sur la reconnaissance de l'instant de la fermeture glottique uniquement. Avec un signal synthétisé dont la 1/f0 = 10 ms, différentes segmentations ont été testées où L valait 40, 25.6, et 12.8 ms, l'erreur relative de l'estimation de la période donnait moins de 2% pour les premières valeurs de L et moins de 6% pour la dernière. Cette dégradation de la précision avec la diminution de L est due aux effets de bord. Dans le cas d'une ondelette non causale, l'algorithme ne détient pas suffisamment de données pour adéquatement calculer la TOD lorsque le segment est plus petit que la moitié de la longueur de l'ondelette LW, il faut opter pour le recouvrement partiel des segmentations au moins l'équivalent à $L_W/2$ pour se libérer de cette limitation.

3.5 Étude de la parole non synthétisée

Deux voix, féminine et masculine, ont été analysées par les auteurs [11] à l'aide d'une fréquence d'échantillonnage fe = 16 KHz, la période de la f0 était estimée en segmentant la parole par une fenêtre rectangulaire de longueur L = 32 ms où la TOD est réalisée. Pour éviter les effets de coupure sur la convolution, le calcul se fait sur un bloc inclus

dans le segment dont les limites temporelles sont [LW/2, L - LW/2]. Le segment suivant pris en charge est décalé par un saut S = LW/2.

Les voix ont aussi été soumises aux deux autres méthodes avec les mêmes paramètres de saisie. Lorsqu'il s'agissait de la voix masculine, les trois méthodes donnaient une estimation de la période de f0 similaire. Dans le cas de la voix féminine, sur les parties de signaux vocaux, l'autocorrélation et la TOD rendaient un résultat similaire cependant la méthode du cepstrum donnait une estimation plus changeante. Sur les parties de signaux non vocaux les méthodes d'autocorrélation et de la cepstrale comportaient des erreurs quand venait le temps de distinguer les parties vocales des non vocales surtout autour des 200 à 220 ms et 260 à 300 ms. La méthode de la cepstrale rencontre des difficultés de traitement lorsque les parties vocales de la voix féminine contiennent un contenu spectral riche en harmoniques de la f0, d'où son choix ardu à déterminer f0.

La TOD offre également une performance supérieure pour déterminer la f0 indépendamment de la nature féminine ou masculine de la voix. La méthode de la cepstrale se comporte bien avec des voix masculines où la f0 appartient au domaine spectral des basses fréquences de la voix. L'autocorrélation se comporte bien avec des voix féminines où la f0 appartient au domaine spectral des hautes fréquences de la voix.

Par contre, l'autocorrélation comme la cepstrale n'arrivent pas à détecter les variations aussi facilement que la TOD qui contrairement aux autres méthodes n'assume pas à priori dans son traitement la stationnarité ou quasi-stationnarité à l'intérieur de la fenêtre d'analyse.

3.6 Conclusion sur les performances de la TOD

La TOD représente selon les auteurs Kadambe et Boudreaux-Bartels [11] et [12] un excellent choix comme méthode pour la détection de la fondamentale de la parole. Ses points forts par rapport aux autres méthodes approchées sont les suivants :

- Elle ne fait pas l'hypothèse de la stationnarité ou de la quasi-stationnarité dans le signal analysé
- Elle offre une estimation très précise dans un signal bruité (erreur relative maximale de 2 % pour un RSB de –18 dB en essai de simulation)
- Elle localise parfaitement le début de toute période de fondamentale f0 et les successives dans un segment donné devenant ainsi un choix idéal pour des applications faisant intervenir des méthodes de synchronisation.
- Elle est idéale pour une grande gamme de période de fondamentale traitant également les voix féminines ou masculines.
- Sa mise en œuvre est relativement simple en terme de calcul et de traitement numérique car elle se contente de deux ou trois échelles d'ondelettes pour obtenir ses résultats.

3.7 Les variantes de l'algorithme de la TOD

La plupart des experts [11] et [12] qui adoptent la solution de la TOD sont d'avis que le type d'ondelette le plus propice à ce genre d'application est le B-spline. D'autres auteurs tels que Qiu, Koh et Yang de la référence [13] ont proposé une ondelette à leur avis plus facile à construire et cependant s'apparentant et conservant les caractéristiques de la famille des splines. Cette fonction est impaire et bien localisée dans le domaine fréquentiel et temporel. Elle permet donc d'avoir :

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{3.5}$$

L'approche de ces derniers auteurs [13] est substantiellement différente de l'algorithme présenté au début [11]. De plus ils ne se contentent pas d'utiliser la TOD dans le domaine temporel uniquement mais aussi dans le fréquentiel.

Il faut remarquer que même l'emploi de la TOD dans le domaine temporel est modifié. Plutôt que de soumettre en analyse un segment de la parole sur trois différentes échelles

dyadiques par exemple, il débute l'analyse du segment à l'échelle 2^j où j = 1 et c'est le résultat obtenu du traitement de la TOD qui est alors soumis de nouveau à celle-ci mais cette fois à l'échelle incrémentée, soit j +1, le résultat suivant est alors traité par la TOD à l'échelle suivante j = 3. Le résultat final est supérieur avec cet algorithme qui élimine rapidement les détails du signal analysé avec l'emploi récursif de la TOD sur les résultats successifs plutôt que sur le signal original.

La fondamentale peut aussi être déterminée dans le domaine fréquentiel à l'aide de la TOD. La fréquence fondamentale est mesurée par la distance séparant deux crêtes spectrales adjacentes. A partir d'un segment de voix on extrait son logarithme spectral qui est lui traité par la TOD cette fois sur une seule échelle j = 3.

Dans le cas temporel ou fréquentiel le facteur de translation de l'ondelette reste ici inchangé soit $\tau = 0$.

Les auteurs G.A. Shelby, C.M. Cooper et R.R. Adhami [14] ont suggéré de redresser le signal à analyser c'est-à-dire ne retenir que la partie positive du signal, cela réduit la quantité d'information à traiter sans pour autant nuire au traitement de la TOD pour déduire la période de f0.

3.8 Notre choix d'ondelettes

Le présent document a gardé les échelles de la première méthode [11]. Nous n'avons pas retenu l'approche des auteurs [10] sur la méthode choisie pour la détection de la f0 car selon [15] la transformée de Hilbert pour trouver la fréquence instantanée dans un milieu bruité n'est pas une méthode robuste.

Cependant nous garderons la transformée de Hilbert pour chercher l'enveloppe d'un signal dans la partie ICA vue plus loin. Nous avons choisi l'algorithme de la deuxième

méthode dans le domaine temporel [13] pour trouver la f0, par contre les choix de l'ondelette de ces auteurs n'ont pas été retenu.

Bien que respectant le critère du choix de l'ondelette qui demande une ondelette s'apparentant au signal à analyser, nous avons préféré la première dérivée d'une gaussienne comme illustrée sur la figure 7 et recommandée par [13].

Elle s'apparente effectivement très bien au signal de la fermeture glottique de la voix. L'idée du redressement [14] du signal a été retenue dans l'évaluation des maxima locaux des fenêtres sur signal à analyser par la TOD.

Cette ondelette choisie va être contractée ou dilatée selon l'échelle retenue. Comme suggéré par les auteurs [11], [13] nous avons gardé trois échelles dyadiques de telle sorte qu'elles s'insèrent dans la bande spectrale de la fréquence fondamentale de la voix humaine.

La figure 8 donne l'occupation du domaine spectrale de l'ondelette pour les trois échelles choisies.

Nous avons repris l'exemple du signal utilisé par les méthodes classiques de détection de la fréquence fondamentale f0 vues au chapitre 2. les résultats sont montrés sur la figure 9, où nous pouvons voir comment la f0 est déterminée sur le signal analysé.

La figure 9a est le signal analysé qui est traité par la transformée en ondelettes sur les trois échelles consécutives. Le résultat est montré sur la figure 9c où l'on remarque des maxima claires sans ambiguïté. L'algorithme par la suite relève la durée entre chaque maximum donnant la période de la f0. la figure 9b est le résultat de ce calcul.

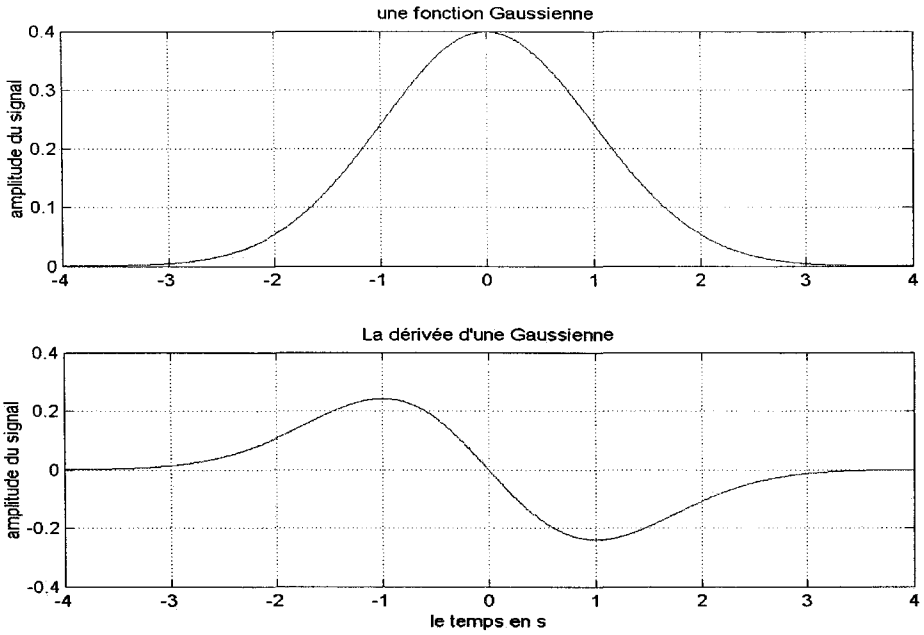

Figure 7 La Gaussienne et sa dérivée comme ondelette mère

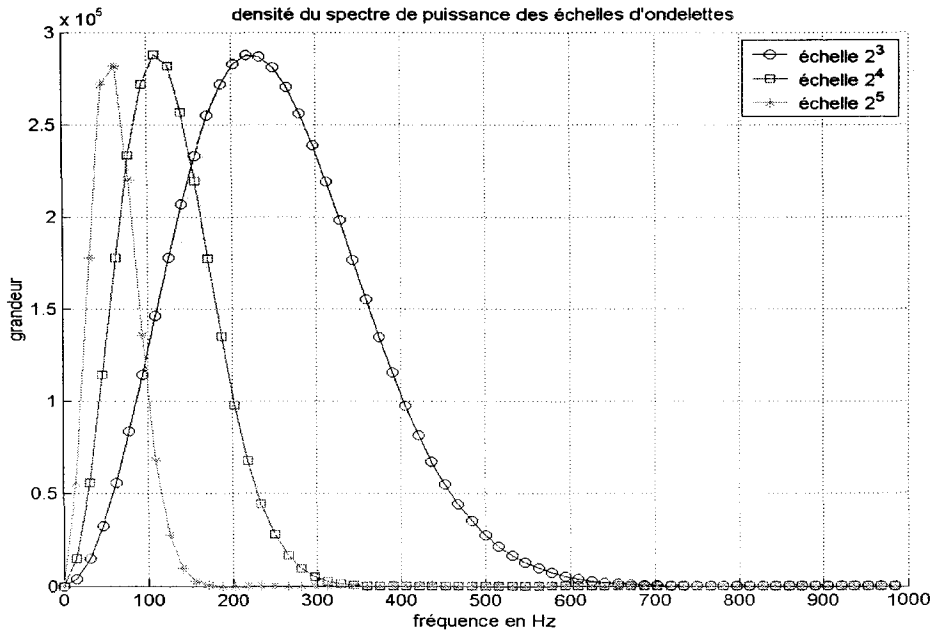

Figure 8 Domaine spectrale des trois échelles d'ondelette

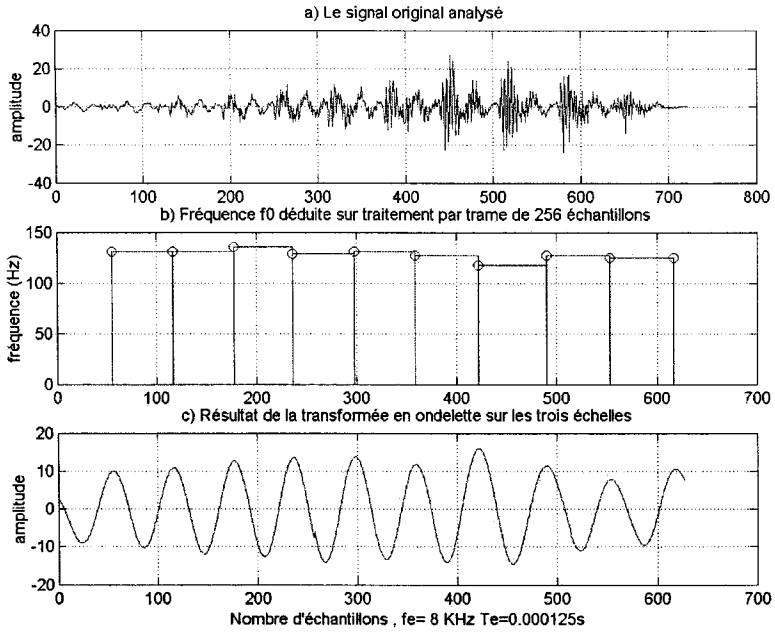

Figure 9 Détermination de la f0 par transformée en ondelettes

Chaque trait vertical de la figure 9b correspond à un maximum de la figure 9c et entre deux maxima on les relie par un trait horizontal dont la projection sur l'ordonnée donne la fréquence de la f0 qui varie légèrement entre 120 Hz et 135 Hz.

Les grandes lignes de l'algorithme sont montrées avec les principales opérations sur la figure 10. Cet algorithme est mis en œuvre plus en détail dans les programmes tant de simulation que d'application en langage C qu'on peut consulter respectivement dans les annexes I et II.

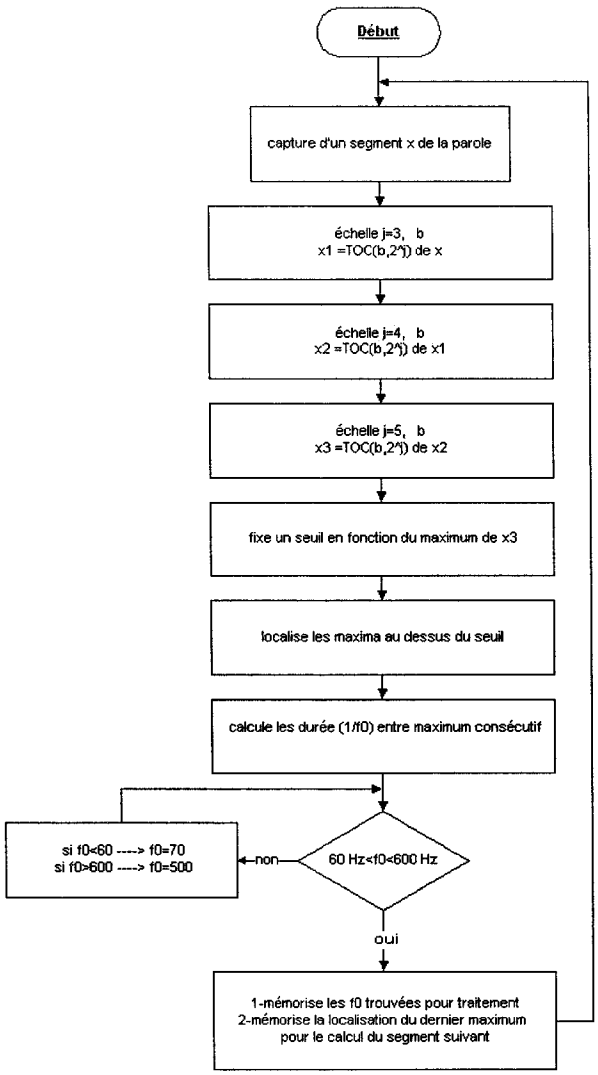

Figure 10 L'algorithme de détection de la f0 avec TOC sur trois échelles

3.9 le banc de filtres de type passe-bande adaptatif

La démarche consiste à recourir au principe des liens harmoniques qu'on retrouve particulièrement dans tout son voisé dont la nature bien que non-stationnaire puisse être partitionnée pour relever les éléments stationnaires ou plus précisément quasi-périodiques. Pour se faire, on utilise un banc de filtres de type passe-bande centrés sur la fréquence fondamentale f0 et ses harmoniques. Nous obtiendrons ainsi selon la f0

mesurée, un nombre de sous-bandes du signal étudié reliées aux harmoniques de la f0 et restant à l'intérieur de la fréquence de Nyquist.

C'est par ce contenu fréquentiel en harmoniques que le système auditif humain est selon la théorie retenue [10], le plus susceptible de distinguer les sons voisés issus d'un milieu bruité. Il est à remarquer que la largeur de bande des filtres du banc est dans l'ordre de grandeur de la fréquence centrale des passe-bandes. Cela confère à cette phase du traitement du signal un comportement similaire à celui qu'on retrouve dans le système auditif humain de la cochlée [10] pour ce qui est de la non-linéarité d'échelle de la réponse aux stimuli sonores.

3.10 Construction des passe-bandes du banc de filtres

Les filtres de type passe-bande sont élaborés selon l'information de la référence [10] qui veut simuler la réponse du système auditoire humain où la bande passante de chaque filtre est proportionnelle à la f0. les filtres vont s'adapter à la fréquence fondamentale f0 du moment de la façon suivante : le premier passe-bande va égaler sa fréquence centrale à celle de la f0, qui est fixée approximativement sur la moyenne géométrique des fréquences de coupure du passe-bande. En définissant par α la fréquence de coupure basse et β la fréquence de coupure haute, nous pouvons résumer ainsi les particularités du premier passe-bande :

$$f0 = \sqrt{\alpha\,\beta} \tag{3.6}$$

$$\beta = 2\,\alpha \tag{3.7}$$

$$LB = \beta - \alpha = \alpha \tag{3.8}$$

$$\alpha = \frac{f0}{\sqrt{2}} \quad et \quad \beta = \sqrt{2}\,f0 \tag{3.9}$$

Le passe-bande suivant fera coïncider sa fréquence de coupure basse à celle de coupure haute du premier passe-bande autrement dit $\beta_0 = \alpha_1$. Dès lors la fréquence de coupure haute du passe-bande suivant est déduite de la relation (3.7), il en est de même de la fréquence centrale par la relation (3.6) qui est aussi une harmonique de la f0. Finalement la largeur de bande du passe-bande suivant sera le double de la largeur de bande du premier.

Ces relations sont conservées pour tous les passe-bandes qui se suivent. Le nombre de sous-bandes sera égal au nombre de passe-bandes permises par le domaine fréquentiel que la fréquence de Nyquist délimite.

Le banc de filtre est lié à la fréquence fondamentale f0 et sa structure se présente dans le tableau I :

Tableau I

Structure du banc de filtre

Les n bande-passantes	La fréquence centrale (Hz)	α coupure basse	β coupure haute	LB= β- α Largeur de bande
0	f0	$\alpha_0 = f0/2^{1/2}$	$\beta_0 = 2^{1/2} f0$	$LB_0 = \alpha_0$
1	2f0	$2\alpha_0$	$2\beta_0$	$2LB_0$
2	4f0	$4\alpha_0$	$4\beta_0$	$4LB_0$
3	8f0	$8\alpha_0$	$8\beta_0$	$8LB_0$
n	2^n f0	$2^n \alpha_0$	$2^n \beta_0$	$2^n LB_0$

Le nombre n de passe-bande du banc de filtre, peut être déduit connaissant la fréquence fondamentale f0 et la fréquence de Nyquist fNyq puisque la contrainte de Nyquist impose que :

$$2^n \beta_0 \leq fNyq$$

$$n \leq \frac{\ln\left(\frac{fNyq}{f0\sqrt{2}}\right)}{\ln 2} \qquad (3.10)$$

3.11 Le filtre numérique RIF pour les passe-bandes

Le filtre numérique RIF à réponse impulsionnelle finie (appelé également non récursif) est le type qui sera retenu pour notre application car malgré une demande substantielle en calcul comparativement aux autres filtres numériques, il offre deux qualités incontournables qui sont une stabilité intrinsèque alliée à la garantie d'absence de distorsion de phase lorsque ses coefficients sont agencés dans un ordre symétrique. Les lignes qui suivent présentent brièvement les principales propriétés du filtre RIF. La section actuelle traitant de l'élaboration des filtres en général et du banc de filtres en particulier s'est surtout appuyée sur la référence [16] du traitement numérique des signaux.

L'équation aux différences (EAD) d'un filtre RIF consiste à modéliser la sortie en terme des entrées présentes et passées, de là sa nature non récursive.

$$S(n) = b_0 E(n) + b_1 E(n-1) + ... + b_N E(n-N) \tag{3.11}$$

La transformée en Z de l'EAD prend la forme suivante :

$$\frac{S(z)}{E(z)} = \sum_{k=0}^{N} b_k \, z^{-k} = H(z) = b_0 + b_1 \, z^{-1} + b_2 \, z^{-2} + ... + b_N \, z^{-N} \tag{3.12}$$

Où les b_k sont les coefficients de la réponse impulsionnelle finie du filtre, et N l'ordre du filtre. Si on multiplie H(z) par un facteur unitaire z^N / z^N on obtient la relation intéressante suivante :

$$\frac{S(z)}{E(z)} = \sum_{k=0}^{N} b_k \, z^{-k} = H(z) = \frac{b_0 \, z^N + b_1 \, z^{N-1} + b_2 \, z^{N-2} + ... + b_N}{z^N} \tag{3.13}$$

Elle montre que tous les pôles sont nuls et appartiennent au cercle unitaire du domaine de z c'est-à-dire $|z| < 1$. Cette caractéristique suffit à confirmer la stabilité de la

fonction de transfert H(z). Dans l'expression de H(z) l'ordre de la fonction est spécifié par N.

La réponse en fréquence d'un filtre donne par sa transformée de Fourier continue inverse la réponse impulsionnelle idéale non causale à symétrie paire infinie et de déphasage nul (voir équation 3.15) cette dernière ne peut être concrétisée dans une mise en œuvre réelle. Toutefois une approximation de cette dernière permet de concrétiser sa réalisation en faisant intervenir deux opérations. La première consiste à la rendre finie par sa troncature ou fenêtrage, la seconde provoque une translation temporelle ou retard pour la rendre causale. C'est ainsi qu'après ces opérations nous obtenons la réponse impulsionnelle causale H(z) de l'équation 3.27 qui introduit un déphasage. Cependant ce dernier est linéaire car la symétrie est préservée dans la translation temporelle.

Lorsque l'ordre N de H(z) est paire le nombre de ses termes est impair et la symétrie se déplace autour d'un point central N/2. La transformée de Fourier continue de H(z) devient alors dans ce cas $H(e^{j\omega})$ où son module est réel et son déphasage alors vaut $\theta(\omega) = - \omega N/2 \pm k\pi$.

Le déphasage est donc une fonction linéaire de la fréquence. Cette dernière caractéristique est essentielle pour le traitement des signaux vocaux, car le délai de groupe (voir équation 3.14) devient constant quelle que soit la fréquence. Ainsi on est assuré de l'absence de distorsion de phase.

$$\frac{d\,\theta(\omega)}{d\,\omega} = -\frac{N}{2} \tag{3.14}$$

3.12 La réponse impulsionnelle d'un passe-bande

En partant de l'exigence de la réponse fréquentielle idéale et par l'entremise de la transformée de Fourier inverse continue nous obtenons la réponse impulsionnelle idéale d'un passe-bande $h_{INC}(m)$ où l'abréviation inc signifie idéale et non causale.

$$h_{INC}(m) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} h_{INC}(e^{j\omega})\, e^{jm\omega}\, d\omega \tag{3.15}$$

Où

$$h_{INC}(e^{j\omega}) = \begin{cases} 1 & si\ \omega_1 \leq |\omega| \leq \omega_2 \\ 0 & si\ \omega_1 > |\omega| > \omega_2 \end{cases} \tag{3.16}$$

Alors l'intégration se résume en deux parties

$$h_{INC}(m) = \frac{1}{2\pi}\left[\int_{-\omega_2}^{-\omega_1} e^{jm\omega}\, d\omega + \int_{\omega_1}^{\omega_2} e^{jm\omega}\, d\omega \right] \tag{3.17}$$

Son résultat est :

$$h_{INC}(m) = \begin{cases} \dfrac{\omega_2 - \omega_1}{\pi} & si\ m = 0 \\ \dfrac{\sin m\omega_2 - \sin m\omega_1}{m\pi} & si\ m \neq 0 \end{cases} \tag{3.18}$$

Sa transformée en Z s'exprime ainsi :

$$h_{INC}(z) = \sum_{m=-\infty}^{+\infty} h_{INC}(m)\, z^{-m} \tag{3.19}$$

Sa nature infinie ne peut aboutir à sa réalisation concrète. Il faut donc la rendre finie au moyen d'une troncature de son domaine autour de sa symétrie en la multipliant par une fonction de fenêtrage finie qui pour les besoins de la démonstration sera une fonction fenêtre rectangulaire. Une fenêtre plus performante sera retenue dans notre application et expliquée plus loin. Ainsi nous obtenons :

$$h_{NC}(m) = h_{INC}(m)\ fen(m) \tag{3.20}$$

Où

$$fen\,(m) = \begin{cases} 1 & si\ -M \le m \le +M \\ 0 & autrement \end{cases} \tag{3.21}$$

La réponse impulsionnelle est devenue $h_{NC}(m)$ en remplaçant sa nature idéale infinie par sa forme tronquée mais encore non causale. Maintenant pour rendre la réponse impulsionnelle causale il faut introduire un délai ou retard de M.

La transformée en Z de $h_{NC}(m)$ s'exprime ainsi :

$$h_{NC}(z) = \sum_{m=-M}^{+M} h_{NC}(m)\, z^{-m} \tag{3.22}$$

En introduisant un retard de M la transformée en Z devient causale et s'exprime par :

$$h_C(z) = z^{-M} h_{NC}(z) = \sum_{m=-M}^{+M} h_{NC}(m)\, z^{-(m+M)} \tag{3.23}$$

Pour mieux formaliser $h_C(z)$ le changement de variable n = m+M est souhaité, ce qui donne :

$$h_C(z) = \sum_{n=0}^{+2M} h_{NC}(n-M)\, z^{-n} \tag{3.24}$$

La borne supérieure de la sommation de $h_C(z)$ étant généralement admise comme l'ordre de la transformée alors si N=2M l'expression devient :

$$h_C(z) = \sum_{n=0}^{N} h_{NC}(n-\frac{N}{2})\, z^{-n} \tag{3.25}$$

Puisque

$$h_C(n) = h_{NC}(n-\frac{N}{2}) \tag{3.26}$$

La transformée en Z de la réponse impulsionnelle devenue concrète et réalisable est donc

$$h_C(z) = \sum_{n=0}^{N} h_C(n)\, z^{-n} \tag{3.27}$$

Les transformations nécessaires pour rendre la réponse impulsionnelle idéale et abstraite en une réponse impulsionnelle concrète et réalisable n'ont pas altéré la symétrie paire de la réponse impulsionnelle. Cependant la transformation pour aboutir à la causalité a déplacé le centre de symétrie avec les conséquences qui suivent.

La réponse impulsionnelle non causale $h_{NC}(m)$ définie par son domaine tronqué $-M \leq m \leq +M$ préserve la symétrie paire centré sur $h_{NC}(0)$ d'où la propriété suivante : $h_{NC}(m) = h_{NC}(-m)$. Lorsque le facteur en z du terme centré de sa transformée en Z est mis en évidence et que l'on fait la transformée de Fourier on constate comme pour la réponse idéale que la réponse impulsionnelle non-causale présente un déphasage nul.
Mais pour la réponse impulsionnelle causale $h_C(m)$ son domaine s'est vu décalé par l'ajout d'un retard M redéfinissant ainsi son domaine comme $0 \leq m \leq +2M$. Pour garder la cohérence des changements de variables vus plus haut la réponse impulsionnelle causale devient $h_C(m)$ définie par son domaine $0 \leq n \leq N$.

La symétrie paire se voit ainsi décalée pour être centré sur h_C (N/2) ce qui implique alors la propriété suivante : h_C (n+N/2) = h_C (-n+N/2), cette dernière propriété sera mise à profit lors de la mise en œuvre des filtres dans le DSP, effectivement le calcul sera réduit de moitié puisqu'il suffit de calculer uniquement la moitié des termes de la réponse impulsionnelle. Maintenant si on fait la transformée de Fourier de la transformée en Z de $h_C(n)$ tout en mettant en évidence le facteur en z du terme central de symétrie, on obtient après simplifications l'expression suivante :

$$H_C(e^{j\omega}) = \sum_{n=0}^{N} h_C(n)\, e^{-jn\omega} = \pm P(\omega) e^{-j\frac{N}{2}\omega} \tag{3.28}$$

où ± P(ω) est une fonction réelle de forme polynomiale donnant le module de l'amplitude de $h_C(e^{j\omega})$ et s'exprime comme suit :

$$P(\omega) = \sum_{n=0}^{N/2} C_n \cos(n\omega) \tag{3.29}$$

En tenant compte du signe du polynôme combiné avec l'argument de $h_C(e^{j\omega})$ on constate l'introduction d'un déphasage dans la réponse impulsionnelle causale $h_C(e^{j\omega})$. Son expression générale montre bien sa linéarité en fréquence.

$$\theta(\omega) = -\frac{N}{2}\omega \pm k\pi \qquad k = \{0, 1, 2, ...\} \tag{3.30}$$

3.13 La fonction de fenêtrage

La troncature de la réponse impulsionnelle idéale se fait telle que vue précédemment par un fenêtrage, en multipliant la réponse impulsionnelle idéale par une fonction fenêtre. Dans la démonstration plus haut, on utilise la fenêtre rectangulaire définie en (3.21). Cette dernière est cependant rarement utilisée dans la pratique car contrairement aux autres types de fenêtres elle ne minimise aucunement les effets déformateurs de la troncature sur la réponse impulsionnelle appelés phénomènes de Gibbs. Ces derniers sont des ondulations superposées à la synthèse de la fonction $h_C(e^{j\omega})$ et ceci, indépendamment du nombre de coefficients employés.

Le fait d'augmenter les coefficients diminue l'amplitude des oscillations devenues plus rapides sur les zones de continuité de la fonction sans toutefois réduire l'amplitude des oscillations qui convergent vers les points de discontinuité de la fonction.

C'est précisément pour réduire ces effets que nous cherchons des fenêtres plus performantes. Il existe plusieurs types de fenêtres connues et communément utilisées. Le choix d'une fenêtre va dépendre de l'atténuation maximale recherchée, de l'ordre requis et en relation avec lui de la largeur de la fenêtre. Les plus utilisées sont les fenêtres de Hamming, de Von Hann, de Blackman et de Kaiser.

3.14 La fenêtre de Kaiser

La mise en œuvre des filtres RIF emploie la fenêtre de Kaiser qui contrairement aux autres fenêtres s'ajuste avec une plus grande flexibilité à l'aide de deux paramètres qui sont sa longueur L ou nombre de termes de la fenêtre et β le second paramètre, nécessaire pour évaluer la fonction modifiée de Bessel d'ordre zéro du premier genre $I_0(x)$. Cette fonction va donner principalement la silhouette de la fenêtre et l'ajustement de la raideur de la bande de transition séparant la bande passante de l'atténuée et fixant l'atténuation maximale voulue.

Selon la valeur de β, on peut retrouver les autres fenêtres, ainsi lorsque $\beta = 0$ nous obtenons la fenêtre rectangulaire, quand $\beta = 5.44$ la fenêtre s'apparente très bien à celle de Hamming.

La simplicité de mise en œuvre des autres fenêtres leur permet de s'affranchir de la complexité et du coût élevé en calcul itératif qu'exige la souplesse de la fenêtre de Kaiser. Cet aspect ne sera pas à négliger lors de la mise en œuvre du DSP dans l'éventualité de contraintes en temps réel importantes, à moins de placer au préalable dans une table le résultat des calculs itératifs qui accélère le procédé ce qui a été notre choix.

La démarche pour construire la fenêtre de Kaiser est prise de la référence [16], il s'agit d'une approche d'optimisation alliée à un calcul basé sur des relations empiriques. Les étapes sont les suivantes :

- On débute en choisissant le coefficient d'ondulation minimale requise qu'il soit dans la bande passante δ_P ou atténuée δ_A autrement dit $\delta = \min(\delta_P, \delta_A)$ pour trouver son évaluation en dB telle que $A = -20 \log(\delta)$.
- On fixe la largeur de la bande de transition désirée séparant la bande passante de la bande atténuée de telle sorte que $\Delta\omega = \omega_{LA} - \omega_{LP}$

L'ordre N du filtre RIF sera déduit de la relation suivante :

$$N \geq \frac{A-8}{2.285\,\Delta\omega} \tag{3.31}$$

Il est à remarquer que suite à ce calcul approché, N retenu sera toujours un nombre pair afin que le nombre de termes de la fenêtre comme ceux de la réponse impulsionnelle tronquée soit impair L = N + 1 pour assurer la symétrie centrale des coefficients et garantir un déphasage linéaire.

La valeur de β est déterminée par les relations fixées selon la plage impliquée de l'atténuation A exprimée en dB.

$$\beta = \begin{cases} 0 & si\ A < 21 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & si\ 21 \leq A \leq 50 \\ 0.1102(A-8.7) & si\ A > 50 \end{cases} \tag{3.32}$$

Une fois β et N connus, le calcul des coefficients de la fenêtre de Kaiser causale $W_{KC}(n)$ sont déterminés par la relation suivante :

$$W_{KC}(n) = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1-\left(\dfrac{n-N/2}{N/2}\right)^2}\right]}{I_0(\beta)} & si \quad 0 \leq n \leq N \\ 0 & autrement \end{cases} \tag{3.33}$$

$I_0(x)$ est la fonction modifiée de Bessel d'ordre zéro du premier genre, elle est évaluée de la façon suivante :

$$I_0(x) = 1 + \sum_{k=1}^{\infty}\left[\frac{(x/2)^k}{k!}\right]^2 \tag{3.34}$$

En pratique une précision satisfaisante est atteinte en ne requérant tout au plus que 20 à 25 termes de la sommation de cette expression.

La figure 11 illustre l'allure d'une fenêtre de Kaiser, dont les paramètres sont à titre d'exemple ; $\delta_{min} = 0.01$, A= 40 dB, et $\Delta\omega = 0.0897$, l'ordre ainsi obtenu est N=158

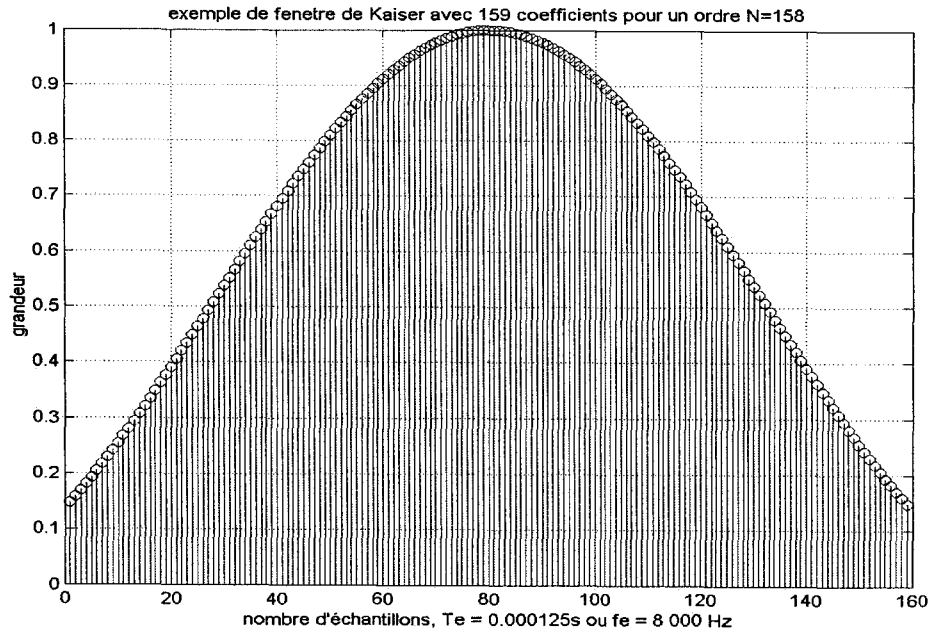

Figure 11 Fenêtre de Kaiser

3.15 Les signaux issus du banc de filtres

Le banc de filtres de type passe-bande centrés sur la fréquence fondamentale f0 et ses harmoniques donne en sortie de chaque filtre une sous-bande du signal observé c'est-à-dire $r_{i,k}(t)$. Puisque le traitement ne veut retenir que l'harmonicité propre de la voix choisie. Il va chercher l'enveloppe de chaque sous-bande de signal $\hat{A}_{i,k}(t)$ pour s'en servir comme signal modulant une porteuse à la fréquence centrale du filtre passe-bande concerné. Selon la sous-bande en question cette fréquence centrale est soit la fondamentale f0 ou une de ses harmoniques.

Pour se faire, ce dernier traitement est réalisé à l'aide d'une transformée de Hilbert dont le principe est vu en détail plus loin et qui à partir de chaque sous-bande en entrée, extrait en sortie l'enveloppe de chaque sous-bande qui va moduler sa porteuse respective. Se faisant, le signal obtenu au cours du traitement $Z_{i,k}(t)$ a perdu l'information de la phase originale. Le processus de recouvrer l'information de la phase fait appel à la théorie de Wiener qui suit, pour donner le signal synthétisé avec le déphasage corrigé $x_{i,k}(t)$.

3.16 Les filtres de Wiener

Développés à partir de concepts temporels plutôt que fréquentiels, ces filtres sont conçus pour minimiser l'erreur quadratique moyenne entre leur signal en sortie et un signal désiré. Ils sont optimaux au sens du critère de l'erreur quadratique moyenne et nous verrons que dans ce cas les coefficients du filtre sont liés à la fonction d'autocorrélation du signal d'entrée et à l'intercorrélation entre les signaux d'entrée et de sortie désirée. Cette partie du document a souvent puisé dans les références [17], [18], [19], [20], et [21], [22], [23], [24].

Le problème d'estimation linéaire est le suivant : x(n) correspond au signal qui nous intéresse mais n'est pas directement accessible. Seul y(n) l'est suite au résultat du passage de x(n) dans un système linéaire.

On veut retrouver x(n) à partir de y(n). Une solution consiste à filtrer y(n) de telle sorte que la sortie du filtre soit la plus proche possible de x(n). On peut mesurer l'erreur de l'estimation par e(n) définie par :

$$e(n) = x(n) - \hat{x}(n) \tag{3.35}$$

Évidemment, plus l'erreur e(n) sera faible, plus l'estimation sera bonne. On cherche donc un filtre qui minimisera l'erreur. Minimiser $e^2(n)$ revient à minimiser une fonction quadratique dérivable. Par ailleurs, étant donné que les signaux intéressants sont

aléatoires, la fonction coût qui sera à minimiser est l'erreur quadratique moyenne souvent connue sous son appellation anglaise MSE (mean square error) définie par :

$$\xi(n) = E[e^2(n)] \tag{3.36}$$

Le filtre optimal de Wiener est un filtre qui minimisera la MSE.

3.17 Filtre de Wiener de type RIF

Le filtre adaptatif se construit autour du calcul d'un filtre RIF. Soit h, le filtre que nous recherchons et N la longueur de sa réponse impulsionnelle dont la forme matricielle est :

$$\mathrm{h} = [h_0 \quad h_1 \quad \ldots \quad h_{N-1}]^T \tag{3.37}$$

Le signal estimé est alors :

$$\hat{x}(n) = \sum_{k=0}^{N-1} h_k y(n-k) \tag{3.38}$$

ou en introduisant la notation matricielle pour y(n) on a :

$$\hat{x}(n) = \mathrm{h}^T \mathrm{y}(n) \Leftrightarrow \hat{x}(n) = \mathrm{y}^T(n)\mathrm{h} \tag{3.39}$$

avec

$$\mathrm{y} = [y(n) \quad y(n-1) \quad \ldots \quad y(n-(N-1)]^T \tag{3.40}$$

Avec l'hypothèse que les signaux x(n) et y(n) sont stationnaires, on arrive à la fonction coût suivante :

$$\xi = E\left[\left(x(n) - \mathrm{h}^T \mathrm{y}(n)\right)^2\right] \Leftrightarrow \xi = E\left[x^2(n) - 2\mathrm{h}^T \mathrm{y}(n)x(n) + \mathrm{h}^T \mathrm{y}(n)\mathrm{y}^T(n)\mathrm{h}\right] \tag{3.41}$$

$$\xi = E\left[x^2(n)\right] - 2\mathrm{h}^T \Phi_{yx} + \mathrm{h}^T \Phi_{yy} \mathrm{h} \tag{3.42}$$

où Φyy est la matrice d'autocorrélation de taille NxN est :

$$\Phi_{yy} = E[\mathrm{y}(n)\mathrm{y}^T(n)] \tag{3.43}$$

le vecteur d'intercorrélation Φyx de taille N est défini par :

$$\Phi_{yx} = E[\,\mathrm{y}(n)x(n)] \tag{3.44}$$

La fonction coût MSE ξ(n) montre qu'elle dépend de la réponse impulsionnelle du filtre RIF h. Pour en obtenir le minimum, on cherche les conditions d'annulation de la dérivée de la fonction coût par rapport au variables que sont les N points de la réponse impulsionnelle du filtre. On a donc une dérivée partielle totale c'est-à-dire un gradient à annuler.

La dérivée de la fonction coût par rapport au $j^{ième}$ point de la réponse impulsionnelle donne :

$$\frac{\partial \xi}{\partial h_j} = E\left[\frac{\partial}{\partial h_j}\left\{e^2(n)\right\}\right] = E\left[2e(n)\frac{\partial e(n)}{\partial h_j}\right] \tag{3.45}$$

En explicitant l'équation e(n), on obtient :

$$\frac{\partial \xi}{\partial h_j} = E\left[2e(n)\frac{\partial}{\partial h_j}\left\{x(n) - \mathrm{h}^T \mathrm{y}(n)\right\}\right] \tag{3.46}$$

Le fait que la sortie du filtre $h^T y(n)$ peut s'écrire comme une somme de N produits dont un seul contient le terme h_j, nous permet d'écrire l'expression suivante :

$$\frac{\partial \xi}{\partial h_j} = E\left[2e(n)\frac{\partial}{\partial h_j}\left\{h_j \mathrm{y}(n-j)\right\}\right] \Leftrightarrow \frac{\partial \xi}{\partial h_j} = E\left[-2e(n)y(n-j)\right] \tag{3.47}$$

Pour avoir les conditions d'annulation de cette équation sur tous les j={0, ..., N-1}, on utilise un ensemble de N équations qui s'écrit sous forme matricielle par la notion du vecteur gradient comme suit:

$$\nabla \xi = \begin{bmatrix} \partial\xi/\partial h_0 \\ \partial\xi/\partial h_1 \\ \cdot \\ \partial\xi/\partial h_j \\ \cdot \\ \partial\xi/\partial h_{N-1} \end{bmatrix} = -2E\begin{bmatrix} y(n)e(n) \\ y(n-1)e(n) \\ \cdot \\ y(n-j)e(n) \\ \cdot \\ y(n-N+1)e(n) \end{bmatrix} = -2E\left\{\begin{bmatrix} y(n) \\ y(n-1) \\ \cdot \\ y(n-j) \\ \cdot \\ y(n-N+1) \end{bmatrix} e(n)\right\} = -2E[\mathrm{y}(n)e(n)] \tag{3.48}$$

En utilisant les équations de l'erreur et de la fonction de transfert du filtre pour remplacer e(n) on obtient :

$$\nabla \xi = -2E\left[y(n)(x(n) - \mathrm{y}^T(n)\mathrm{h})\right] = -2E\left[\mathrm{y}(n)x(n)\right] + 2E\left[\mathrm{y}(n)\mathrm{y}^T(n)\right]\mathrm{h} \tag{3.49}$$

Cette équation devient en introduisant le vecteur d'intercorrélation et la matrice d'autocorrélation:

$$\nabla \xi = -2\Phi_{yx} + 2\Phi_{yy}\mathrm{h} = 0 \tag{3.50}$$

La réponse impulsionnelle optimale h_{opt} annule cette équation d'où :

$$\mathrm{h}_{opt} = \Phi^{-1}{}_{yy}\, \Phi_{yx} \tag{3.51}$$

Ce filtre de Wiener de type RIF permet d'obtenir une erreur quadratique minimale entre x(n) désirée et son estimé donnée par :

$$\xi_{\min} = E\left[x^2(n)\right] - h_{opt}^T \Phi_{yx} \tag{3.52}$$

3.18 Algorithme LMS pour le filtrage adaptatif

La mise en œuvre d'un filtre (estimateur) optimal de Wiener demande la connaissance des caractéristiques du signal, de la matrice d'autocorrélation et du vecteur d'intercorrélation, du bruit et de la fonction de transfert du canal. Cela implique également que ces caractéristiques soient stables au cours du temps, ce qui n'est pas le cas en pratique. La matrice d'autocorrélation et le vecteur d'intercorrélation comme la fonction coût sont inconnus généralement dans les cas pratiques.

On va approcher le filtre optimal de Wiener en utilisant une boucle de retour et un algorithme de minimisation : c'est ce que l'on appelle le filtrage adaptatif illustré sur la figure 12. Dans ce cas, on remplacera la connaissance des fonctions de corrélation par une phase d'apprentissage permettant de modifier itérativement la réponse impulsionnelle du filtre, en affectant les coefficients du filtre.

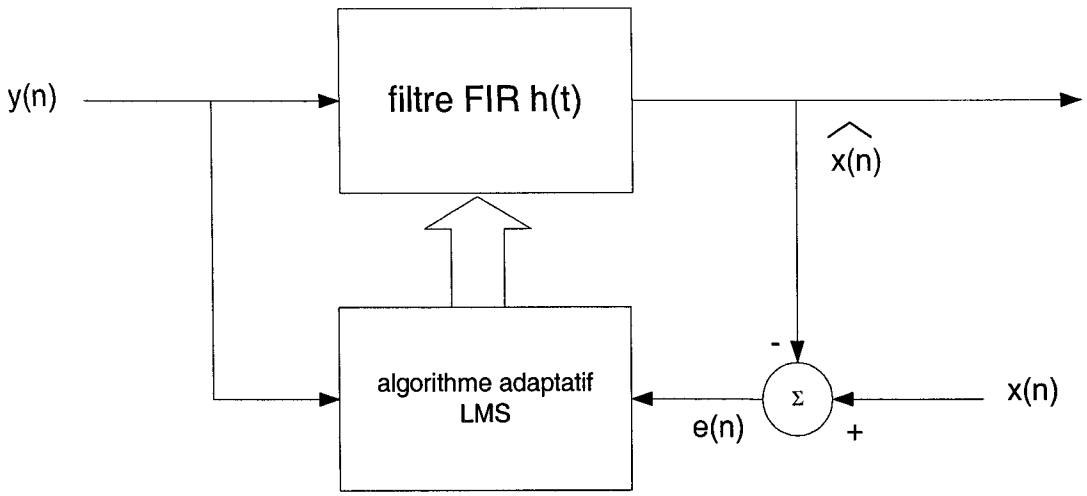

Figure 12 Principe du filtrage adaptatif

Le filtrage adaptatif a pour objet d'approcher ces filtres optimaux. Pour cela, les coefficients de la réponse impulsionnelle du filtre sont adaptés en fonction de l'erreur par une boucle de retour comme le montre la figure 12.

Cette adaptation nécessite une séquence d'apprentissage et une stratégie de mise à jour des coefficients du filtre dont l'objectif est la minimisation d'une erreur. Pour cela, on utilisera des algorithmes d'optimisation. L'algorithme choisi ici pour sa simplicité est le LMS largement utilisé en filtrage adaptatif.

La réponse impulsionnelle d'un filtre adaptatif est donc variable dans le temps. Elle dépend du signal reçu, de la séquence d'apprentissage et de l'algorithme d'optimisation utilisé.

Le signal estimé est défini de la façon suivante :

$$\hat{x}(n) = \sum_{k=0}^{N-1} h_k(n-1)\, y(n-k) \Leftrightarrow \hat{x}(n) = \mathrm{h}^{\mathrm{T}}(n-1)\mathrm{y} \tag{3.53}$$

Il est calculé à l'instant (n) en utilisant la réponse impulsionnelle du filtre calculée précédemment au moment (n-1) par l'algorithme d'optimisation.

Il existe plusieurs techniques d'optimisation basées sur la descente en gradient qui peuvent être mise en œuvre. L'algorithme LMS (Least Mean Squares) dont le détail est donné ci-dessous est souvent utilisé dans les systèmes de filtrage adaptatif et c'est cette approche que nous retenons pour les besoins de notre application. On s'y réfère souvent comme l'algorithme du gradient stochastique.

3.19 Détail de l'algorithme LMS

Initialisation du filtre RIF h(0) = 0. À chaque échantillon n, les opérations suivantes sont accomplies :

$$\begin{aligned} \hat{x}(n) &= h^T(n-1)\ y(n) \\ e(n) &= x(n) - \hat{x}(n) \\ h(n) &= h(n-1) + 2\mu\ y(n)\ e(n) \end{aligned} \tag{3.54}$$

Dans notre application, y(n) est notre signal d'entrée représenté par la fréquence kf0 modulée par l'enveloppe $\hat{A}_{i,k}(t)$ obtenue par la transformée de Hilbert du signal $r_{i,k}(t)$ à la sortie du banc des filtres. Ces derniers signaux représentent selon la sous-bande k le signal désiré x(n) de notre algorithme par lequel se définit la séquence d'apprentissage qui permettra de retrouver la phase originale en sortie du filtre adaptatif.

L'emploi de l'algorithme du LMS se justifie par le fait qu'il est essentiel en traitement de signal de traiter autant faire se peut, les données observées directement, autrement dit en ligne. Ainsi, on évite d'avoir recours dans les calculs, à l'espérance mathématique comme cela a été vu plus haut pour les besoins d'analyse.

On va donc travailler sur h pour minimiser la fonction coût :

$$\xi(n) = E\{e^2(n)\} \tag{3.55}$$

en mettant à jour h tel que :

$$h(n+1) = h(n) - \mu \nabla \xi\ (n) \tag{3.56}$$

Il faut pour cela évaluer le gradient et on déduit ci-dessous son expression :

$$\begin{aligned} &\nabla \xi(n) = \nabla E\{e^2(n)\} = E\{\nabla\ e^2(n)\} = E\{2\ e(n)\ \nabla e^*(n)\} \\ &et \quad \nabla e^*(n) = \nabla(x(n) - \hat{x}(n)) = \nabla(x(n) - h^T y^*(n)) = -y^*(n) \end{aligned} \tag{3.57}$$

donc nous avons :

$$\nabla \xi(n) = \frac{\partial \xi(n)}{\partial h} = -2E\{e(n)\ y^*(n)\} \tag{3.58}$$

Ainsi la mise à jour de l'algorithme du LMS devient :

$$h(n+1) = h(n) - \mu \nabla \xi\ (n) = h(n) + 2\mu\ E\{e(n)\ y^*(n)\} \tag{3.59}$$

on élimine l'espérance mathématique en l'approximant par son estimée instantanée tout en se limitant au domaine réel, l'algorithme du gradient stochastique s'exprime finalement par :

$$h(n+1) = h(n) + 2\mu\ e(n)\ y(n) \tag{3.60}$$

Le facteur μ est le pas d'adaptation de l'algorithme et agit à titre de gain sur le gradient. Sa grandeur est cruciale dans l'atteinte de la convergence de l'algorithme. Dans le cas de signaux stationnaires, on démontre que la convergence est atteinte lorsque :

$$\lim_{n\to\infty} h_n = \Phi_{yy}^{-1}\,\Phi_{yx} \quad \Leftrightarrow \quad 0 < \mu < \frac{2}{\lambda_{\max}} \tag{3.61}$$

La borne supérieure de μ fait intervenir λ_{max} qui est la valeur propre maximale de la matrice d'autocorrélation Φ_{yy} qui comme on le sait est rarement connue en pratique. De plus, la grandeur de cette borne supérieure n'est jamais approchée. On lui préfère à la place de λ_{max} la trace de la matrice d'autocorrélation Φ_{yy} pour une borne plus sûre.

Comme la matrice d'autocorrélation Φ_{yy} se réfère généralement à des signaux stationnaires elle se présente sous le type Toeplitz. Alors sa trace s'évalue ainsi :

$$tr(\Phi_{yy}) = (p+1)\Phi_{yy}(0) = (p+1)E\left\{|y(n)|^2\right\} \tag{3.62}$$

où p est l'ordre du filtre, ou encore le nombre des éléments de la diagonale principale de la matrice d'autocorrélation Φ_{yy} . Encore ici l'espérance mathématique est substituée par son estimée instantanée.

Jusqu'à présent le pas d'adaptation est considéré constant bien que cette particularité ne soit pas idéale dans un contexte où la matrice d'autocorrélation Φyy est rarement connue et que les signaux ne sont pas toujours garantis stationnaires et peuvent varier en nature, on préfère alors un pas d'adaptation qui varie selon la puissance du processus traité. L'algorithme LMS est alors dit normalisé et on obtient :

$$\mu(n) = \frac{\beta}{L\|y(n)\|^2} \tag{3.63}$$

Où β du pas normalisé est un facteur qui se situe dans les limites telles que $0 < \beta < 2$. L représente l'ordre du filtre, ou encore le nombre des éléments de la diagonale principale de la matrice d'autocorrélation. La mise à jour de l'algorithme du gradient stochastique prend finalement la forme suivante:

$$h(n+1) = h(n) + \beta\,\frac{y(n)}{\varepsilon + L\|y(n)\|^2}\,e(n) \tag{3.64}$$

où ε est une petite valeur constante qui assure que le quotient ne s'emballe pas et maintienne la convergence advenant que y(n) du dénominateur tend vers zéro.

Dans notre application [10], y(n) est le signal d'entrée synthétisé par la fréquence kf0 modulée par l'enveloppe $\hat{A}_{i,k}(t)$ qui est obtenue par la transformée de Hilbert du signal $r_{i,k}(t)$ filtré à la sortie par le k passe-bande du banc des filtres. Ces derniers signaux représentent selon la sous-bande k le signal désiré x(n) de notre algorithme par lequel se définit la séquence d'apprentissage qui permettra de retrouver la phase originale en sortie du filtre adaptatif.

Donc la phase est retrouvée à partir du signal synthétisé en s'appuyant sur un filtre adaptatif de type Wiener [10]. Le processus est le suivant ;

En premier, le signal synthétisé se définit ainsi :

$$Z_{i,k}(t) = \begin{bmatrix} Z_{1,i,k} & Z_{-1,i,k} \end{bmatrix}^T \tag{3.65}$$

Où i est le canal propre au capteur (microphone) considéré, k la sous-bande du banc de filtres associée à la fondamentale f0 ou à une de ses harmoniques kf0 sachant que le signal synthétisé se construit plus particulièrement comme :

$$Z_{q,i,k}(t) = \hat{A}_{i,k}(t) e^{q\, j 2\pi f_0(t) t} \qquad où\ q = (1,-1) \tag{3.66}$$

Ce signal est appliqué à l'entrée du filtre de Wiener dont le signal estimé présent à la sortie devient :

$$\hat{x}_{i,k} = h_{i,k}^T Z_{i,k} \tag{3.67}$$

De façon plus détaillée et d'un point de vue matriciel, cette dernière relation s'écrit :

$$X = [Cp \;\; Cm]\begin{bmatrix} Zp \\ Zm \end{bmatrix} = Cp\,Zp + Cm\,Zm \tag{3.68}$$

$$Zp = A\,e^{j\omega} \quad et \quad Zm = A\,e^{-j\omega}$$

Le signal en sortie aura la forme suivante :

$$x = Cp\,A e^{j\omega} + Cm\,A e^{-j\omega} \tag{3.69}$$

Les coefficients du filtre de Wiener dans h(t) sont Cp et Cm et se trouvent à être l'un par rapport à l'autre son conjugué complexe. Ce sont eux, qui ajoutent la phase au signal synthétisé pour obtenir le signal estimé x(t) identique au signal désiré $r_{i,k}(t)$ par l'adaptation permise selon l'algorithme de Wiener.

La phase du signal synthétisé peut être déduite en fonction du temps en posant l'expression complexe des coefficients du filtre de Wiener h(t) ainsi :

$$Cp = a + jb \qquad \& \qquad Cm = a - jb \tag{3.70}$$

Alors le signal en sortie du filtre de Wiener devient :

$$\begin{aligned}
x &= Cp\,A(\cos\omega + j\sin\omega) + Cm\,A(\cos\omega - j\sin\omega) \\
x &= Cp\,A\cos\omega + Cm\,A\cos\omega + j(Cp\,A\sin\omega - Cm\,A\sin\omega) \\
x &= (Cp + Cm)A\cos\omega + j(Cp - Cm)A\sin\omega \\
x &= 2a\,A\cos\omega + j\,2jb\,A\sin\omega \\
x &= 2a\,A\cos\omega - 2b\,A\sin\omega \\
x &= 2A(a\cos\omega - b\sin\omega) \\
x &= 2A\sqrt{a^2+b^2}\,\cos(\omega - \Phi) \qquad où\;\; \Phi = tg^{-1}\left(\frac{b}{a}\right)
\end{aligned} \tag{3.71}$$

Ce signal représente par le filtre adaptatif de Wiener le signal filtré $r_{i,k}(t)$ désiré mais cette fois-ci reconstruit à partir d'un signal synthétisé par l'enveloppe $\hat{A}_{i,k}(t)$ modulant en

amplitude une pure harmonique basée sur la fréquence fondamentale kf0 corrigée par la contribution de la phase $\Phi(t)$.

Nous avons vu dans notre application [10] que l'emploi de la transformée de Hilbert du signal $r_{i,k}(t)$ filtré à la sortie d'un des k passe-bandes du banc de filtre était nécessaire pour élaborer l'enveloppe $\hat{A}_{i,k}(t)$. Cette enveloppe va moduler la fondamentale ou une de ses harmoniques pour synthétiser et se substituer aux signaux réels filtrés.

La transformée de Hilbert possède quelques propriétés intéressantes largement utilisées en traitement de signal et en communication, comme dans notre cas, elle est très souvent employée pour extraire l'enveloppe de signaux à bande étroite, c'est-à-dire des signaux dont la majorité de l'énergie est centrée autour d'une fréquence qui est dite porteuse dans le cas de la modulation d'amplitude. Avant d'aborder cet emploi, expliquons brièvement en quoi consiste la transformée de Hilbert.

Elle permet de transformer tout signal réel x(t) en un signal analytique z(t) sous sa forme complexe à fréquence uniquement positive. Ainsi :

$$z(t) = x(t) + j\, y(t) \tag{3.72}$$

Où

$$y(t) = TH\{x(t)\} \tag{3.73}$$

La partie imaginaire du signal analytique, y(t) est la transformée de Hilbert du signal réel autrement dit de la partie réelle du signal analytique qui a subit un décalage temporel d'un quart de cycle.

Ce décalage temporel ou déphasage peut être réalisé par un filtre passe-tout provoquant un déphasage constant de $\pi/2$ aux fréquences négatives et de $-\pi/2$ pour les fréquences

positives, à cet égard, il est appelé filtre de transformée de Hilbert puisque sa sortie est la transformée de Hilbert du signal appliqué à son entrée. En notation complexe, un décalage temporel d'un quart de cycle ou si l'on veut, un déphasage de ± π/2 se traduit ainsi :

$$e^{\pm j\pi/2} = \cos(\pi/2) \pm j\sin(\pi/2) = 0 \pm j = \pm j \tag{3.74}$$

Pour les signaux stationnaires à fréquences positives ou négatives, cette catégorie de signaux peut se traduire également comme :

$$x_+(t) = e^{+j\omega t} \quad et \quad x_-(t) = e^{-j\omega t} \tag{3.75}$$

Si on produit un déphasage de + π/2 aux fréquences négatives et de - π/2 aux fréquences positives comme le fait la transformée de Hilbert, nous obtenons :

$$y_+(t) = -je^{+j\omega t} \quad et \quad y_-(t) = +je^{-j\omega t} \tag{3.76}$$

Les signaux analytiques respectifs deviennent :

$$z_+(t) = x_+(t) + j\,y_+(t) \quad et \quad z_-(t) = x_-(t) + j\,y_-(t) \tag{3.77}$$

Une fois ces expressions simplifiées nous avons :

$$\begin{aligned} z_+(t) &= e^{+j\omega t} + j\left(-je^{+j\omega t}\right) \quad et \quad z_-(t) = e^{-j\omega t} + j\left(+je^{-j\omega t}\right) \\ z_+(t) &= e^{+j\omega t} - j^2 e^{+j\omega t} \quad et \quad z_-(t) = e^{-j\omega t} + j^2 e^{-j\omega t} \\ z_+(t) &= 2e^{+j\omega t} \quad et \quad z_-(t) = 0 \end{aligned} \tag{3.78}$$

Nous remarquons ainsi que les fréquences négatives sont annulées lorsque le signal devient analytique.

Jusqu'à maintenant, nous avons abordé la transformée de Hilbert dans le domaine temporel, cependant sa mise en œuvre dans notre application sur le processeur de traitement de signal préfère l'approche fréquentielle de la transformée de Hilbert qui s'avère plus aisée avec l'emploi de l'outil mathématique de la FFT (fast fourier transform) qui est la transformée de Fourier rapide.

Aussi la transformée de Hilbert dans le domaine fréquentiel est présenté ci-après. Si nous revenons à la représentation d'un signal analytique z(t) qui est défini selon:

$$z(t)= f(t)+ j\ \hat{f}(t) \tag{3.79}$$

Sa transformée de Fourier devient :

$$Z(\omega)= F(\omega)+ j\ \hat{F}(\omega) \tag{3.80}$$

Pour être analytique Z(ω) ne peut se situer que sur le côté positif du spectre où seules les fréquences positives se retrouvent. Par conséquent pour toutes fréquences négatives, nous avons :

$$Z(\omega)=0 \quad si \quad \omega<0 \tag{3.81}$$

Cela veut donc dire que :

$$F(\omega)=-j\ \hat{F}(\omega) \quad ou\ encore\ que \quad \hat{F}(\omega)= j\,F(\omega) \quad quand\ \omega<0 \tag{3.82}$$

Comme la fonction en fréquence dans la partie imaginaire est à symétrie impaire :

$$\hat{F}(\omega)=-j\,F(\omega) \quad quand\ \omega>0 \tag{3.83}$$

En résumé la transformée de Hilbert dans le domaine fréquentiel est définie de la façon suivante :

$$\hat{F}(\omega)=-j\,F(\omega) \quad quand\ \omega>0 \quad ou \quad \hat{F}(\omega)= j\,F(\omega) \quad quand\ \omega<0 \tag{3.84}$$

Sous sa forme concise la transformée de Hilbert s'exprime ainsi :

$$\hat{F}(\omega)=-j\,F(\omega)\,\mathrm{sgn}(\omega) \tag{3.85}$$

Rappelons ici les propriétés de la fonction signum sgn(ω) qui se définit ainsi :

$$sgn(\omega)=\frac{|\omega|}{\omega}=\begin{cases} 1 & si\ \omega>0 \\ -1 & si\ \omega<0 \end{cases} \tag{3.86}$$

En conclusion un signal analytique dans le domaine fréquentiel vaut :

$$
\begin{aligned}
&si\ \omega < 0\ alors \\
&Z(\omega) = F(\omega) + j\,\hat{F}(\omega) \\
&Z(\omega) = F(\omega) + j\,[jF(\omega)] \\
&Z(\omega) = F(\omega) + j^2\,F(\omega) \\
&Z(\omega) = F(\omega) - F(\omega) = 0
\end{aligned}
\tag{3.87}
$$

Et

$$
\begin{aligned}
&si\ \omega > 0\ alors \\
&Z(\omega) = F(\omega) + j\,\hat{F}(\omega) \\
&Z(\omega) = F(\omega) + j\,[-jF(\omega)] \\
&Z(\omega) = F(\omega) - j^2\,F(\omega) \\
&Z(\omega) = F(\omega) + F(\omega) = 2\ F(\omega)
\end{aligned}
\tag{3.88}
$$

Une fois le calcul dans le domaine fréquentiel obtenu, il suffit de faire la transformée de Fourier inverse ou une FFT inverse pour retrouver le signal analytique dans le domaine temporel. Le calcul du module de sa forme complexe nous donne généralement l'enveloppe lorsque le signal analytique est un signal de type à bande étroite ou de modulation d'amplitude.

3.20 Introduction sur L'ACI

La dernière partie du projet s'inspire de la démarche reposant sur la théorie de l'analyse en composantes indépendantes (ACI) [4], sans toutefois y adhérer totalement puisqu'elle a recourt également à d'autres aspects qui s'en dissocient à certains égards. Cependant la démarche appartient plus globalement aux méthodes dites de séparation et d'extraction aveugle de sources (SAS et EAS) communément connues sous l'acronyme anglo-saxon de BSS ou BSE (respectivement blind sources separation ou extraction) .

L'ACI comme les SAS et EAS proviennent du besoin de retrouver les sources originales qui, après avoir été mélangées par le milieu les portant aboutissent sur des capteurs ou antennes donnant des signaux observés qui résultent de la combinaison des sources altérées par les propriétés naturelles du milieu qui agissent souvent comme le fait un

filtre mais variant dans le temps et tributaire de plusieurs facteurs connus comme inconnus.

C'est à partir des signaux observés que ces méthodes vont chercher les différentes sources originales sans connaissances à priori des particularités des sources ni des propriétés propres de mélange ou d'altération du milieu, de là, le terme aveugle employé par ces méthodes. Quand il s'agit de séparation aveugle des sources comme c'est généralement le cas de l'ACI, on cherche à retrouver les différentes sources existantes alors que lorsque l'extraction aveugle de sources est considérée, la recherche se limite à un nombre limité de sources parmi celles existantes dans le milieu observé.

La recherche des sources originales se précise en posant le problème dans son contexte mathématique et matriciel de la façon suivante :

$$X = AS \tag{3.89}$$

Le vecteur X représente les différents capteurs fournissant les signaux observés résultant de la combinaison des sources par les propriétés de mixtion du milieu définies par la matrice de mélange A. Enfin le vecteur S contient les différentes sources recherchées.

Par exemple pour un système composé de trois sources j du vecteur Sj mélangées par la matrice de mélange $[a_{ij}]$ et observées par trois capteurs i donnant les signaux Xi la relation est présentée ainsi :

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{3.90}$$

$$\begin{aligned} x_1 &= a_{11}s_1 + a_{12}s_2 + a_{13}s_3 \\ x_2 &= a_{21}s_1 + a_{22}s_2 + a_{23}s_3 \\ x_3 &= a_{31}s_1 + a_{32}s_2 + a_{33}s_3 \end{aligned}$$

Les méthodes comme l'ACI (ou SAS et EAS) cherchent par différents algorithmes à élaborer une matrice de séparation W qui à partir du vecteur observé va retrouver une estimation des sources recherchées. La démarche est la suivante :

$$\begin{aligned} Y &= WX \\ Y &= WAS \end{aligned} \tag{3.91}$$

L'objectif des différents algorithmes existants est de trouver par calculs statistiques, la matrice de séparation qui tend vers l'inverse de la matrice de mélange inconnue, de telle sorte que :

$$W = A^{-1} \quad pour\ que \quad \hat{Y} = S \tag{3.92}$$

L'enjeu consiste donc à définir cette matrice de séparation W par des méthodes empruntant différents domaines touchant à la théorie de l'information et du traitement du signal où les théories d'estimation et d'optimisation sont mises à contribution [4].

Pour alléger les calculs complexes très exigeants, on a recourt généralement à un prétraitement des données observées afin de travailler avec des signaux dont la moyenne est réduite et la variance unitaire.

De plus les hypothèses de bases concernant les signaux observés exigent pour garantir la performance des méthodes de l'ACI de s'assurer que les sources ou composantes indépendantes sont statistiquement indépendantes et que leur distribution statistique soit non-Gaussienne.

Ces caractéristiques sont d'ailleurs souvent rencontrées dans les signaux vocaux qui nous intéressent. Pour aller dans le sens de ces dernières considérations les données observées seront généralement blanchies ou encore décorrélées.

3.21 La séparation des sources dans notre cas d'étude

Maintenant que la présentation de l'ACI a été brièvement expliquée pour les besoins de compréhension, il sera possible de décrire les similitudes mais surtout les différences dans l'approche préconisée par les auteurs [10].

En ce qui a trait aux similitudes, c'est surtout l'aspect du prétraitement qui est directement appuyé sur l'approche de la méthode de l'ACI.

Pour les différences, il faut remarquer au départ que le nombre de capteur n'égale pas celui des sources ou composantes indépendantes comme vu précédemment. Cette égalité est souvent souhaitée, puisque la matrice de séparation estimée W doit tendre vers la matrice inverse de mélange soit A^{-1}. Hors généralement une matrice doit être carrée pour espérer s'inverser. Lorsque l'inégalité découle de la supériorité du nombre des sources sur celui des signaux observés, l'approche l'ACI est dite à bases surcomplètes et son traitement est ardu, on fait souvent appel aux calculs matriciels employant une matrice pseudo-inverse mais le coût en complexité de calcul est prohibitif pour l'appliquer dans un environnement d'un DSP comme le nôtre.

Mais la différence majeure par rapport à l'approche ACI est qu'on se dissocie du contexte aveugle par le fait que l'information sur les sources ou plus particulièrement sur la source vocale qui nous intéresse est disponible même si la matrice de mélange reste méconnue. On désire extraire seulement une source vocale dont l'énergie est la plus importante. Bien que cette source possède des aspects similaires aux signaux aléatoires elle conserve des portions dont les propriétés sont quasi stationnaires et où l'on peut retrouver une structure caractérisée par une fréquence fondamentale affectée d'un déphasage connu.

C'est précisément ces dernières informations qui permettent d'éviter de recourir à l'approche ACI à bases surcomplètes, elles évitent aussi les écueils souvent rencontrés

dans les résultats donnés par les méthodes générales d'ACI qui consiste dans l'ambiguïté sur l'ordre réel des sources retrouvées et la détermination de leur énergie respective.

3.22 La méthode préconisée pour l'extraction de la source désirée

Dans l'approche des auteurs [10], la méthode du traitement du calcul pour la minimisation de la moyenne des carrés de l'erreur s'effectue principalement en lot ou hors ligne. Sachant que :

$$y_k = W^T x \quad et \quad \varepsilon_k = y_k - e^{-jk\phi_0} \tag{3.93}$$

Et la fonction coût ou l'erreur est :

$$\xi(W) = E\left[\varepsilon_k^2\right] \tag{3.94}$$

atteint son minimum lorsque le gradient s'annule alors on obtient pour W l'expression :

$$W = E\left[xx^T\right]^{-1} E\left[xe^{-jk\phi_0}\right] = E\left[xe^{-jk\phi_0}\right] \quad quand \nabla\xi(w) = 0 \quad et \quad E\left[xx^T\right]^{-1} = I \tag{3.95}$$

Φ_0 étant associée à la fondamentale ou une de ses harmoniques avec son déphasage, le vecteur entrée x ait été préalablement blanchi comme l'indique l'équation 3.95.

Nous suggèrons plutôt un traitement en ligne des auteurs D.P. Mandic et A.Cichoki [25]. Il s'appuie essentiellement sur la nature des signaux vocaux ayant une structure dynamique bien connue, aussi l'algorithme procède sur la propriété prédictive des sources et vise également la minimisation de l'erreur de prédiction instantanée de la source à extraire présentant l'énergie principale comme hypothèse des auteurs [10].

L'algorithme doit rechercher et extraire parmi un ensemble de sources celle qui a l'énergie la plus importante. Pour se faire, il doit se référer à un ou des critères de séparation qui relèvent des particularités ou de la structure du signal recherchée. Dans le cas présent c'est évidemment l'information sur la fréquence fondamentale f0 et son déphasage qui va être utile.

Le modèle de l'algorithme peut être interprété par la figure suivante où est illustré uniquement un canal pour fin de clarté, autrement dit, on ne considère qu'une rangée de la matrice de séparation ici w11 à w1M, où M est le nombre de capteur ou microphone. Les coefficients du filtre prédictif du même canal étant de b11 à b1N .

Pour le canal 1

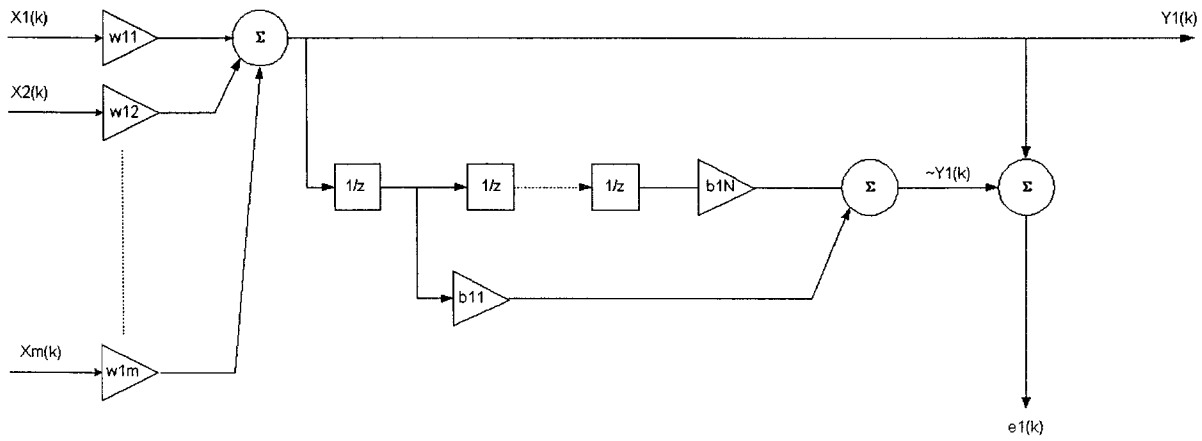

Figure 13 Matrice de séparation et prédicteur rehausseur selon [25]

L'erreur e1(k) sur la figure 13 va influencer la mise à jour des coefficients des filtres prédictifs et de la matrice de séparation W. Le nombre N de coefficient du filtre prédicteur b_{11} à b_{1N} est variable, la figure 13 n'illustre que le premier et dernier coefficient du filtre prédicteur.

La mise à jour des coefficients s'effectue de la façon suivante :

$$b_{1j}(k+1)=b_{1j}(k)-\mu_b \nabla_{b1j} J(w_1(k), b_1(k)) \tag{3.96}$$

$$W_{1i}(k+1)=W_{1i}(k)-\mu_w \nabla_{w1i} J(w_1(k), b_1(k)) \tag{3.97}$$

Sachant que l'expression de l'erreur se définit comme étant :

$$J(w_1(k), b_1(k))=\frac{1}{2}e_1^2(k) \quad et\ que \quad e_1(k)=y_1(k)-\hat{y}_1(k) \tag{3.98}$$

Où

$$y(k)=W^T X \quad et \quad \hat{y}(k)=b^T y(k-1) \tag{3.99}$$

Après avoir effectué le calcul détaillé du gradient la mise à jour de la règle d'apprentissage devient :

$$b_{1j}(k+1)=b_{1j}(k)+\mu_b\, e_1(k)\, y(k-j) \tag{3.100}$$

Et

$$W_{1i}(k+1)=W_{1i}(k)-\mu_w\, e_1(k)\, x_i(k) \tag{3.101}$$

Les taux d'apprentissage ou pas d'adaptation μ_b et μ_w sont également adaptatifs et traités de la même façon que ceux vus plus haut dans les algorithmes du LMS, c'est-à-dire qu'ils sont normalisés par la puissance instantanée de leur entrée respective, mais le présent algorithme introduit un paramètre de combinaison convexe λ défini dans l'intervalle [0,1] pour produire une pondération entre les deux pas d'adaptation. On a ainsi :

$$\mu_b=\frac{\lambda}{\|y(k)\|^2} \quad et \quad \mu_w=\frac{1-\lambda}{\|x(k)\|^2} \tag{3.102}$$

Cette approche sur les pas d'adaptation selon [25] est plus propice pour les signaux à dynamique élevée et non stationnaires et où les propriétés de mixtion sont variables dans le temps.

C'est le nombre de coefficients N du filtre prédictif qui va discriminer la source recherchée. Généralement les sources aux variations lentes exigent des prédicteurs avec des séries de coefficients plus longues que celles aux variations rapides. Connaissant la fréquence fondamentale il suffit de fixer en conséquence la valeur de N valant ainsi $1/f0 = nTe$, c'est-à-dire un multiple de la période d'échantillonnage.

CHAPITRE 4

MISE EN ŒUVRE DE LA SOLUTION CHOISIE

La mise en œuvre de la solution choisie a été faite premièrement en simulation sous Matlab puis s'est réalisée en programme exécutable sous langage C. La simulation sous Matlab part de la figure 6 du schéma synoptique de notre application au chapitre 3. Les différentes opérations sont décrites ci-après.

4.1 Préparation du signal

À partir d'un signal échantillonné, nous lui avons ajouté dans un premier temps un bruit de telle sorte qu'on obtienne un rapport signal sur bruit de 5 dB ou 3 dB selon les signaux essayés. Par la suite nous l'avons fait passer à travers deux filtres RIF qui modélisent l'effet convolutif réverbérant, la réponse fréquentielle de ces filtres est donnée sur la figure 15. On obtient ainsi deux signaux semblant venir de deux microphones illustrés sur la figure 14, où l'abscisse est exprimée en terme du nombre d'échantillon. Les programmes ayant servit à cette préparation sont mis en annexe et s'intitulent; sigbruit.m et mixconvolFIR4.m .

4.2 Détermination de la fréquence fondamentale f0

Chaque canal est traité par la TOD sur trois échelles pour relever les maxima et calculer les périodes associées de la fréquence fondamentale, cela se fait avec le programme freqf0.m, le résultat global est donnée sur les figure 16 et 17 pour chaque canal.

La simulation sur Matlab de la détermination de la fréquence fondamentale f0 selon l'approche vue dans la solution retenue [10] est mise en œuvre dans le programme freqf0.m placé en annexe I, où la fonction CWT (continuous wavelet function) est écartée mais remplacée par son calcul équivalent, ceci afin de mieux exporter sa fonctionnalité vers le langage C qui sera compilé sur DSP éventuellement.

À partir d'un échantillon de voix, on opère la TOD puis pour chaque segment analysé on calcul la fréquence fondamentale f0 qui est la donnée cruciale pour le reste des algorithmes de l'application.

La TOD sur les trois échelles, se fait sur segment de 256 échantillons, le résultat global est illustré sur les figures 16 et 17. La période et la fréquence fondamentale sont calculées sur les crêtes maximales supérieures à un seuil fixé, comme vu au chapitre 3.

4.3 Élaboration d'un banc de filtres RIF sous Matlab

Pour illustrer les principes avancés dans cette partie du document, le programme rédigé sous Matlab dont le nom de fichier est bancFIR.m mis en annexe I simule la construction d'un banc de filtre RIF à partir d'une fréquence fondamentale mesurée sur un échantillon vocal. Le listing est disponible en annexe I. La figure 18 montre le banc de filtre RIF élaboré avec une fenêtre de Kaiser et on montre sa réponse fréquentielle pour une f0 de 128 Hz sur le canal 1, l'autre banc sur le canal 2 étant sensiblement le même. Les figure 19 à 23 illustrent les résultats obtenus pour chaque passe-bande des canaux respectifs donnant les signaux filtrés.

4.4 Extraction de l'enveloppe par transformée de Hilbert

Nous avons également sur les figure 19 à 23 l'allure des enveloppes des différents signaux filtrés obtenues par la transformée de Hilbert.

4.5 La partie couvrant l'ACI

Cette partie débute l'approche de l'ACI vu selon les auteurs [10] qu'on doit considérer dans un sens très large, elle a été également vue brièvement au chapitre 3 où nous avons discuté des similitudes et des différences en rapport avec la définition de l'ACI adoptée par l'ensemble de la communauté des chercheurs scientifiques.

Chaque tranche fréquentielle filtrée sur les deux canaux est prise en entrée avec leur enveloppe respective et traitée séparément en fonction de la f0 ou de ses harmoniques. On élabore le signal synthétisé de chaque tranche fréquentielle avec un signal analytique (une exponentielle) modulée en amplitude par son enveloppe associée mais qui est pour l'instant dépourvue du déphasage original. Elle est donc mise à l'entrée d'un filtre de Wiener de type LMS dont la séquence d'apprentissage est le signal venant de la tranche fréquentielle filtrée. La sortie de ce traitement donne les signaux synthétisés $x_{i,k}(t)$ sur la fondamentale et ses harmoniques avec leurs déphasages corrigés.

C'est le programme ica.m mis à l'annexe I qui assume cette partie. Les figures 24 à 27 donnent les résultats de cette partie de l'ACI pour chaque première tranche fréquentielle synthétisée sur chaque canal et sur la tranche fréquentielle intermédiaire, car dans notre exemple le banc de filtre sur chaque canal est composé de cinq passe-bandes comme l'illustre la figure 18.

Puis vient la seconde partie du traitement de l'ACI où est mise en œuvre la matrice de séparation conjuguée avec le prédicteur linéaire de rehaussement vus à la fin du chapitre 3 et schématisés sur la figure 13. Le programme exécutant ces fonctions se nomme ica1.m et peut être consulté dans l'annexe I. Les figures 28 à 32 donnent les résultats du processus sur chaque composante fréquentielle des signaux synthétisés et combinés des deux canaux. La figure 33 montre le signal original et son estimé avec le délai de la réponse du traitement.

Les algorithmes construits pour la simulation sous Matlab nous ont permis de réaliser les blocs fonctionnels sous programmation en langage C. Chaque partie a été vérifiée par rapport à la simulation faite sur Matlab.

La différence majeure réside dans le traitement segmenté mais inclus cependant l'ensemble des algorithmes dans le programme en langage C alors que la simulation

sous Matlab, a été réalisée sous différents blocs fonctionnels. Le listing du programme intégral en langage C est mis dans l'annexe II.

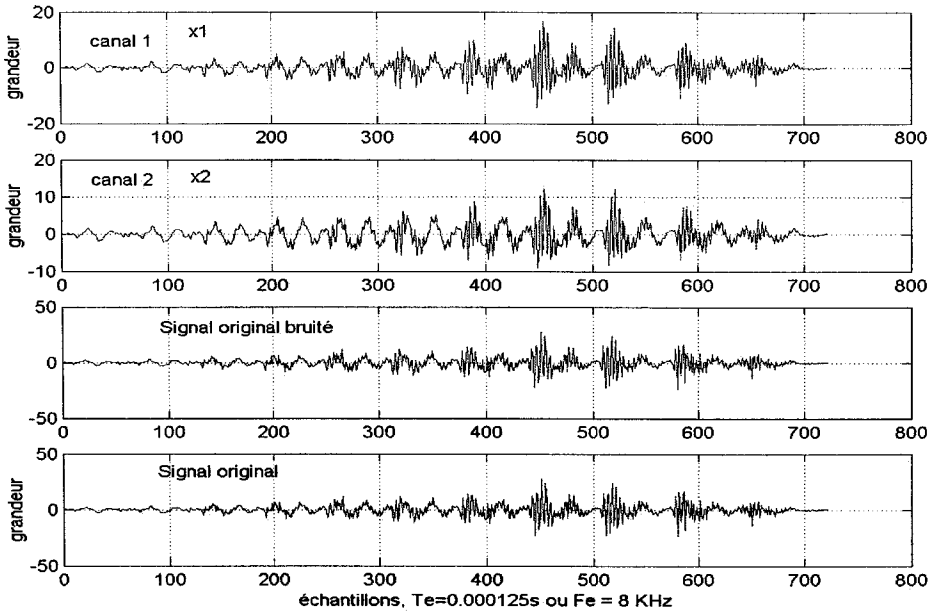

Figure 14 Signaux captés par microphones 1 et 2

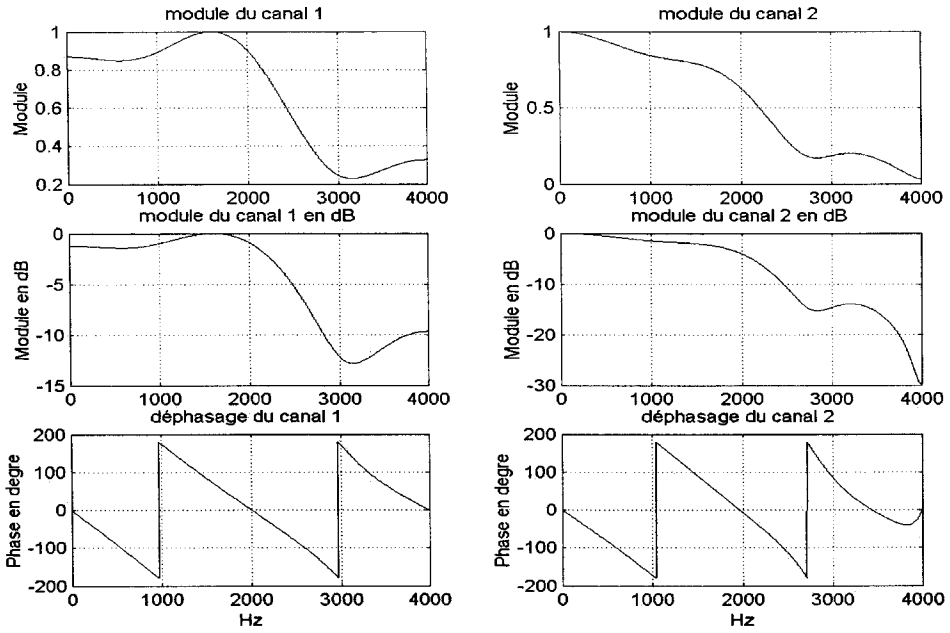

Figure 15 Réponse fréquentielle des filtres RIF des deux canaux

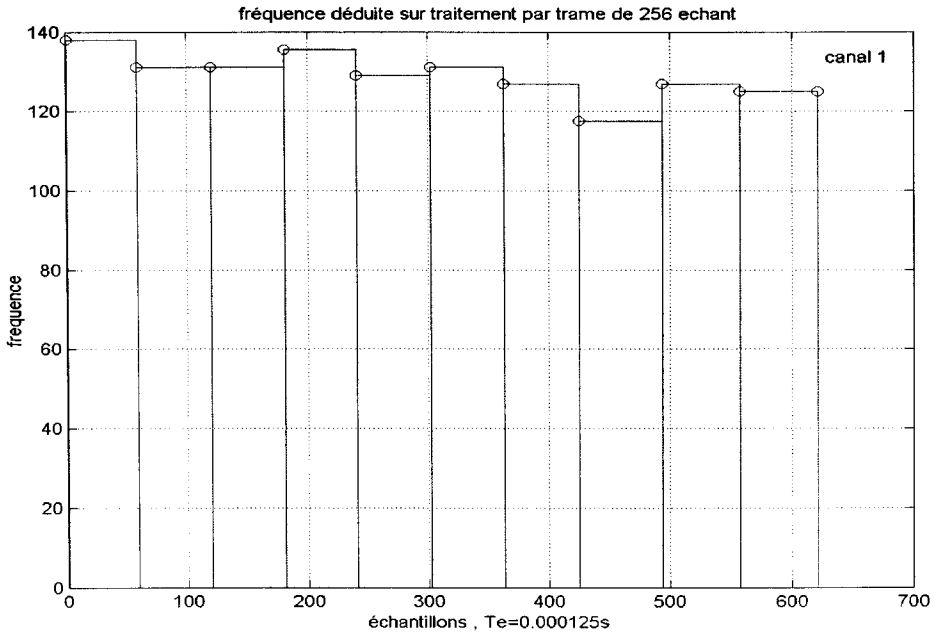

Figure 16 Détection de la fréquence fondamentale f0 sur canal 1

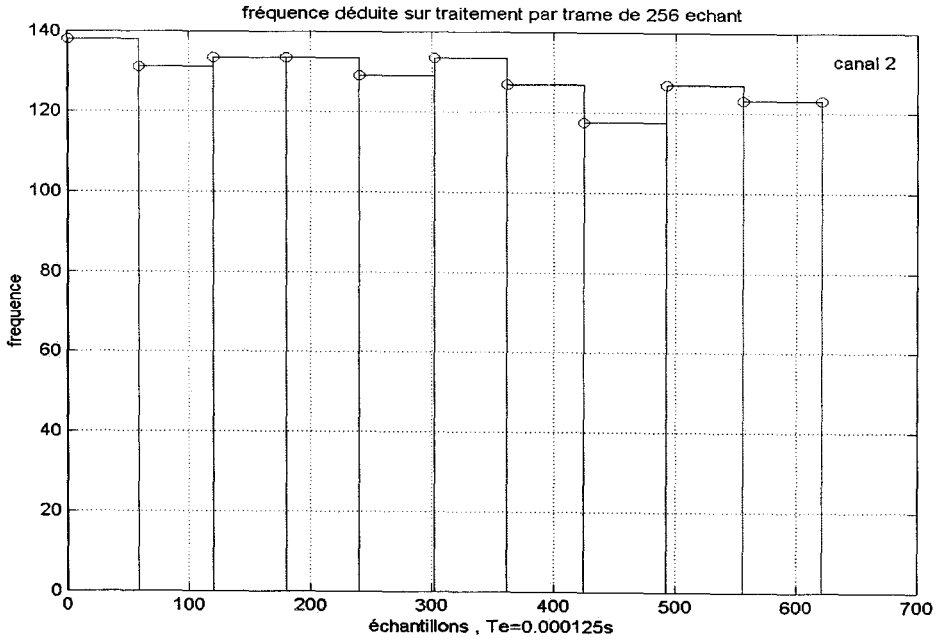

Figure 17 Détection de la fréquence fondamentale f0 sur canal 2

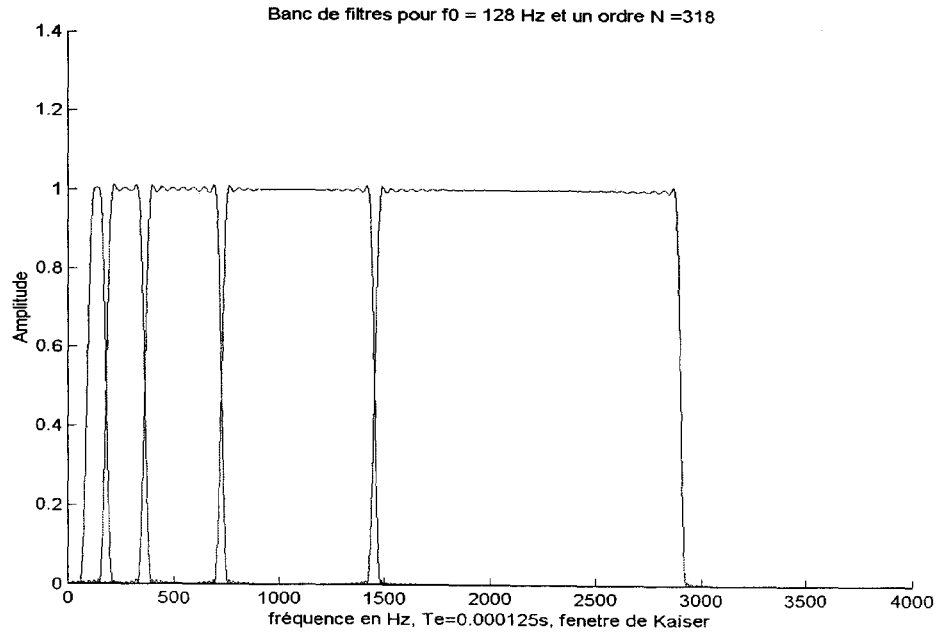

Figure 18 Réponse fréquentielle du banc de filtre canal 1

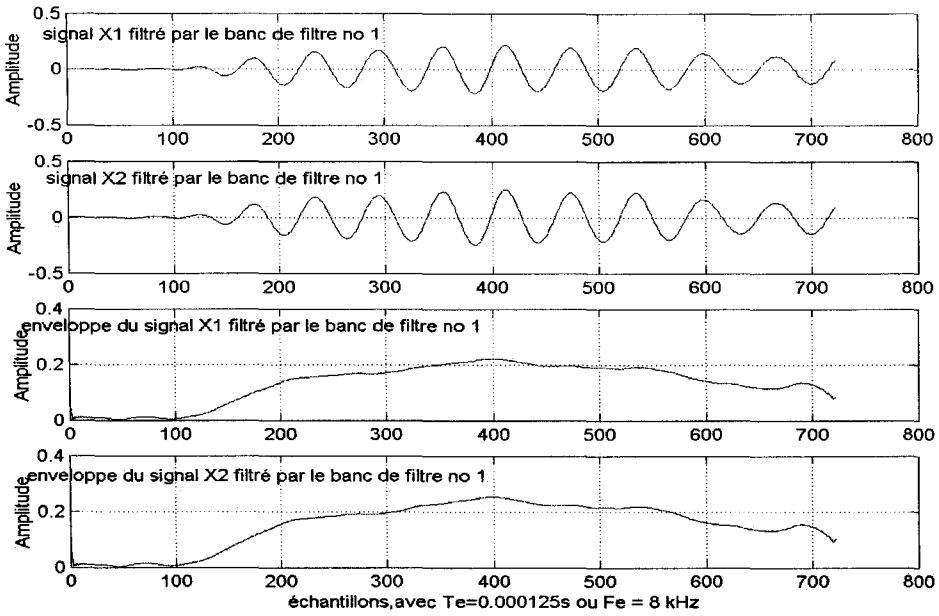

Figure 19 Premier passe-bande filtré autour de f0

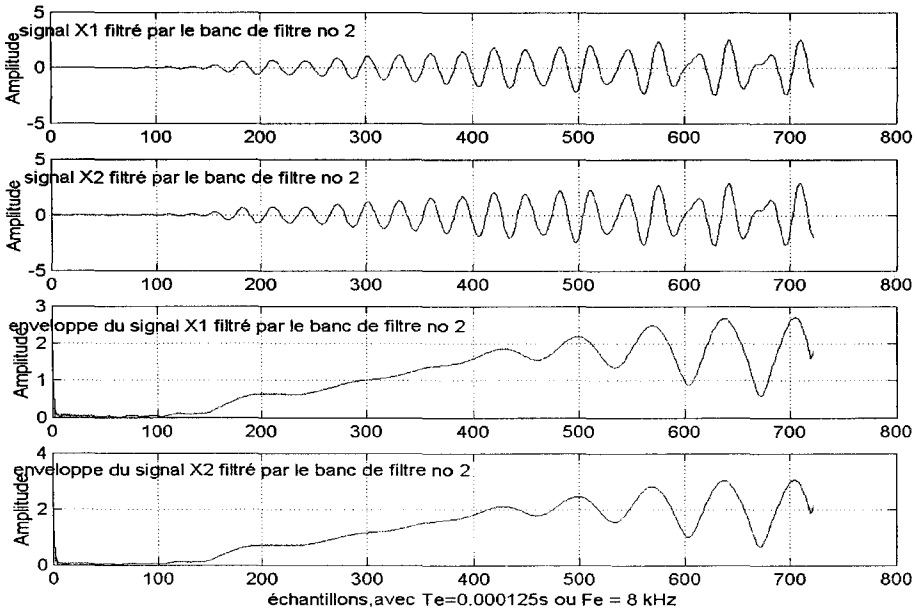

Figure 20 Second passe-bande filtré autour de 2 f0

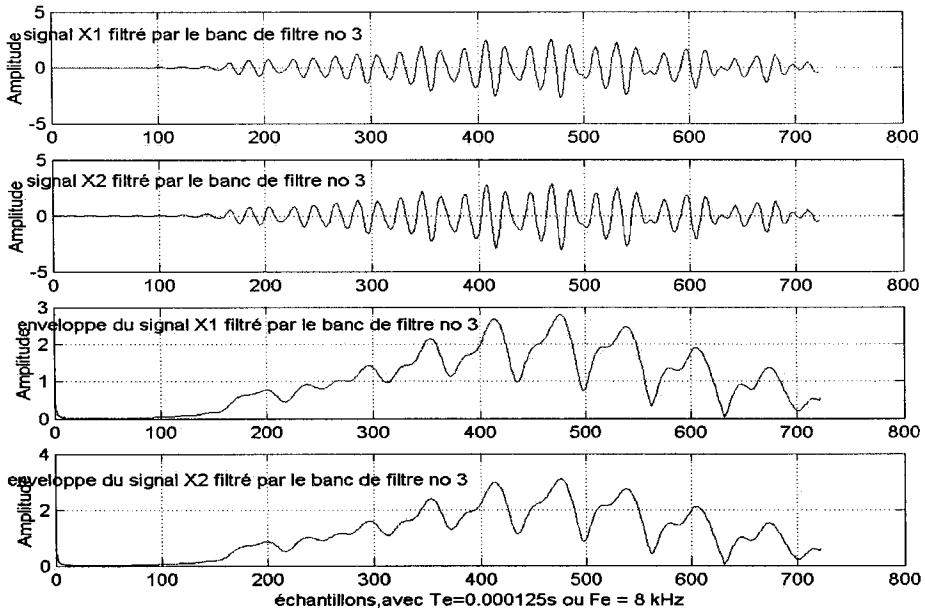

Figure 21 Troisième passe-bande filtré autour de 4 f0

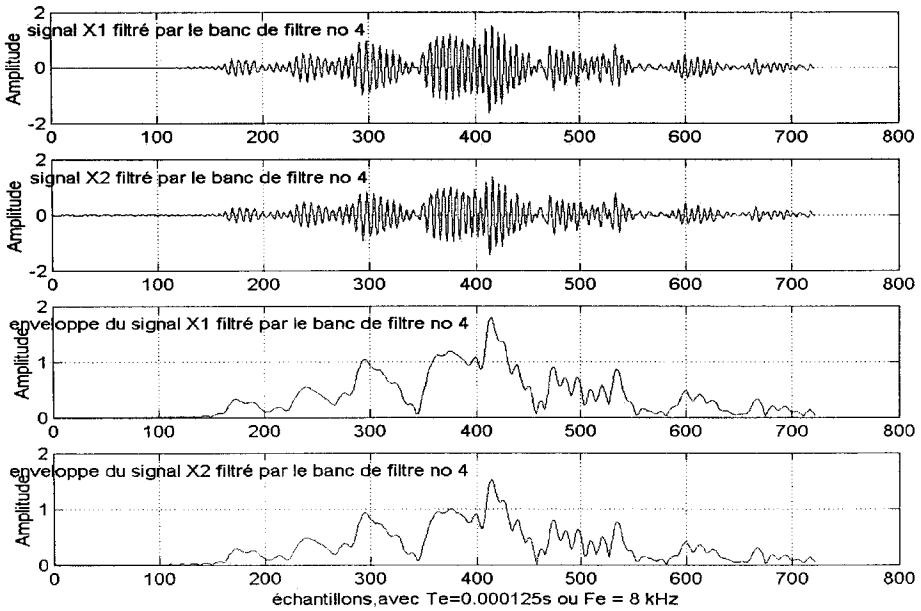

Figure 22 Quatrième passe-bande filtré autour de 8 f0

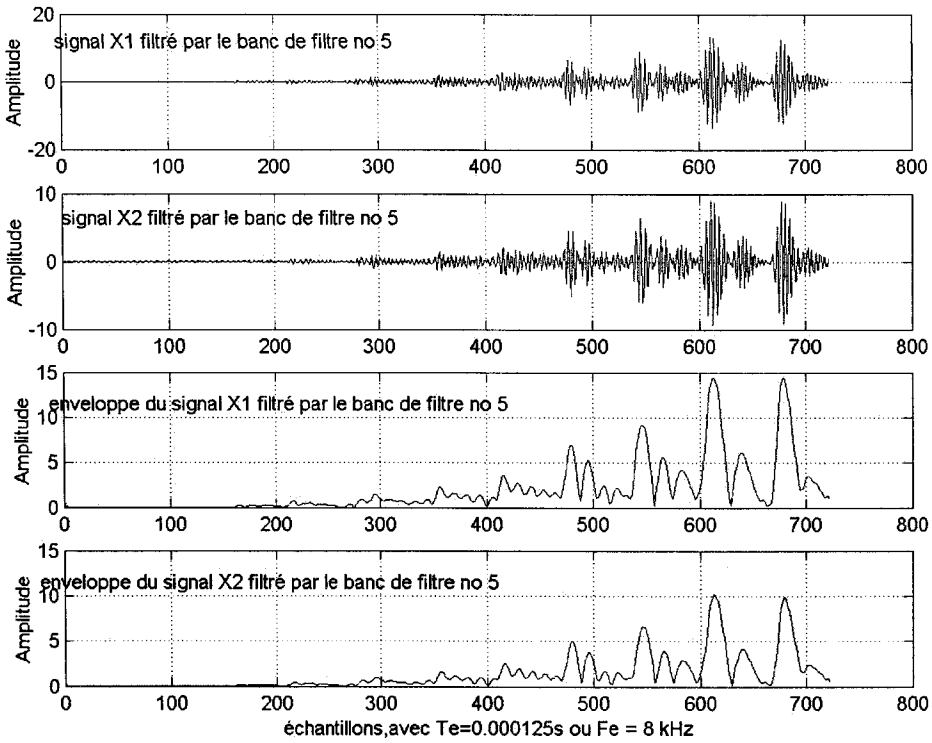

Figure 23 Cinquième passe-bande filtré autour de 16 f0

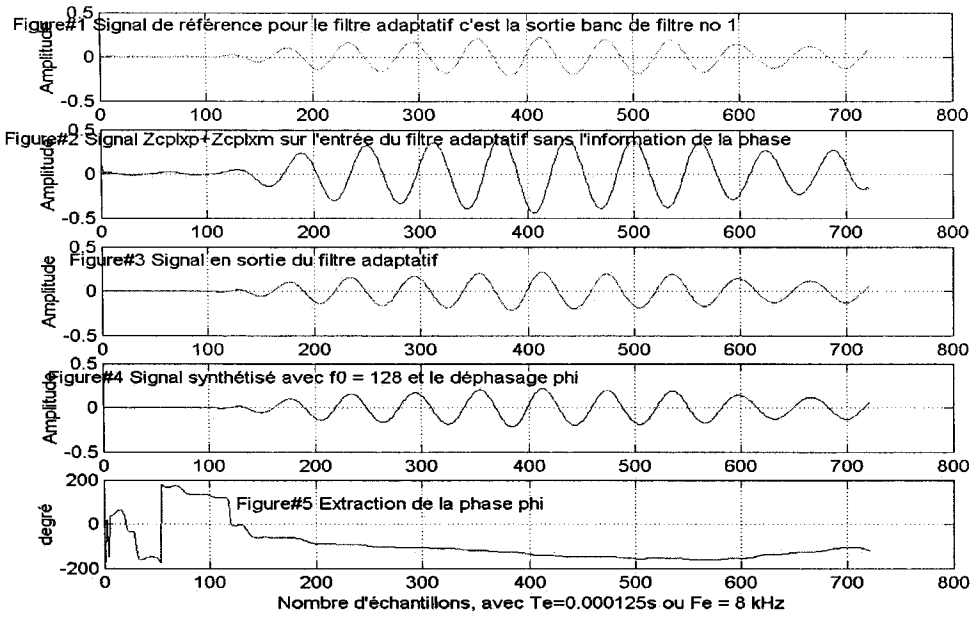

Figure 24 Signal synthétisé avec la f0 sur canal 1

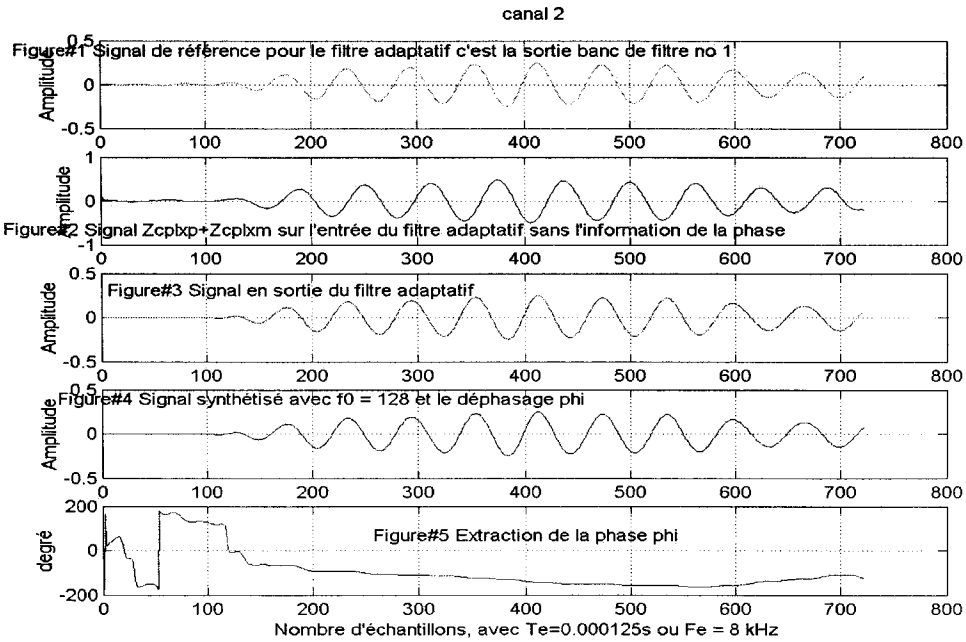

Figure 25 Signal synthétisé avec la f0 sur canal 2

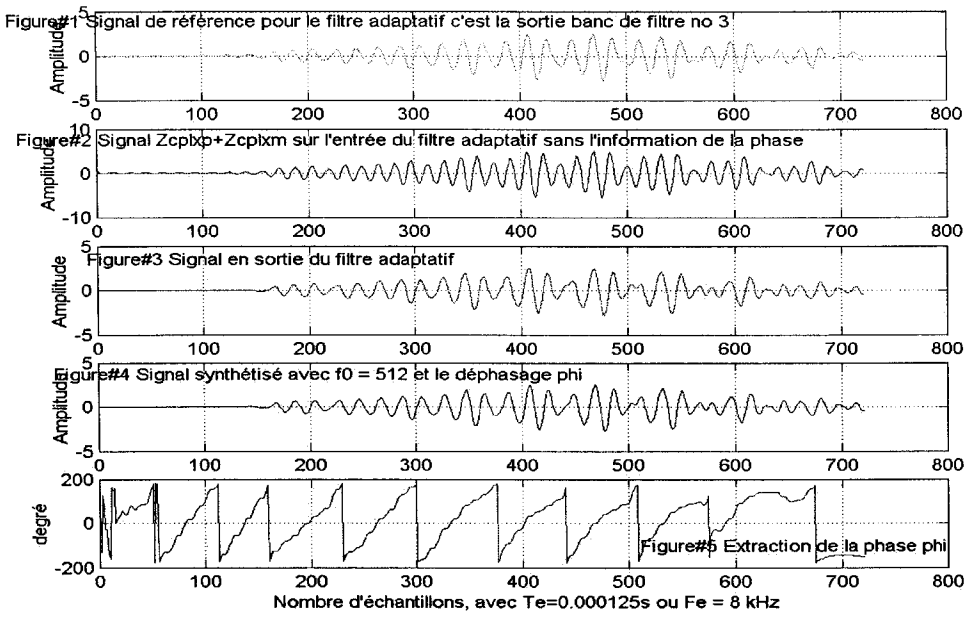

Figure 26 Signal synthétisé avec la 4 f0 sur canal 1

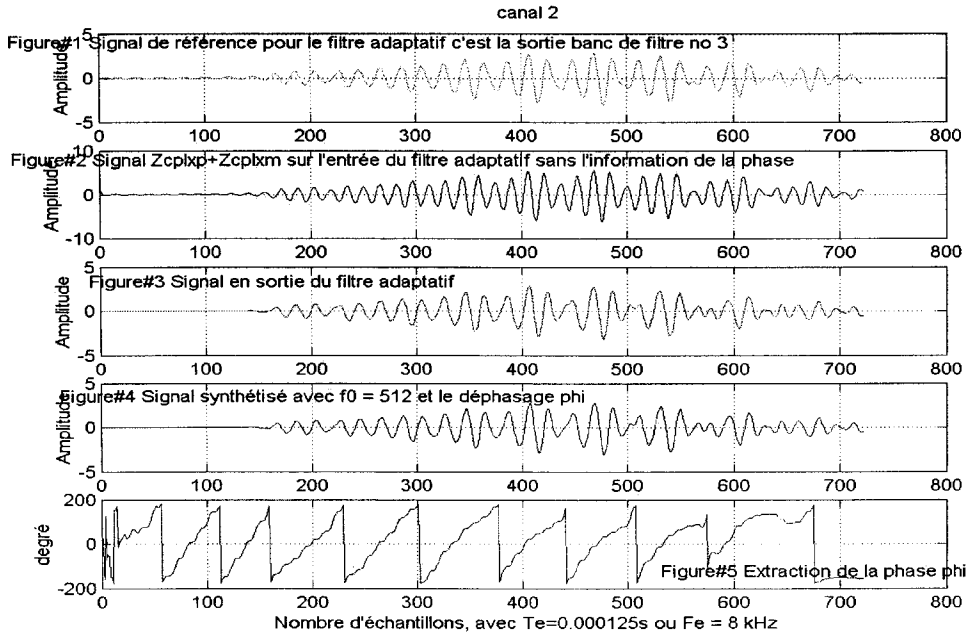

Figure 27 Signal synthétisé avec la 4 f0 sur canal 2

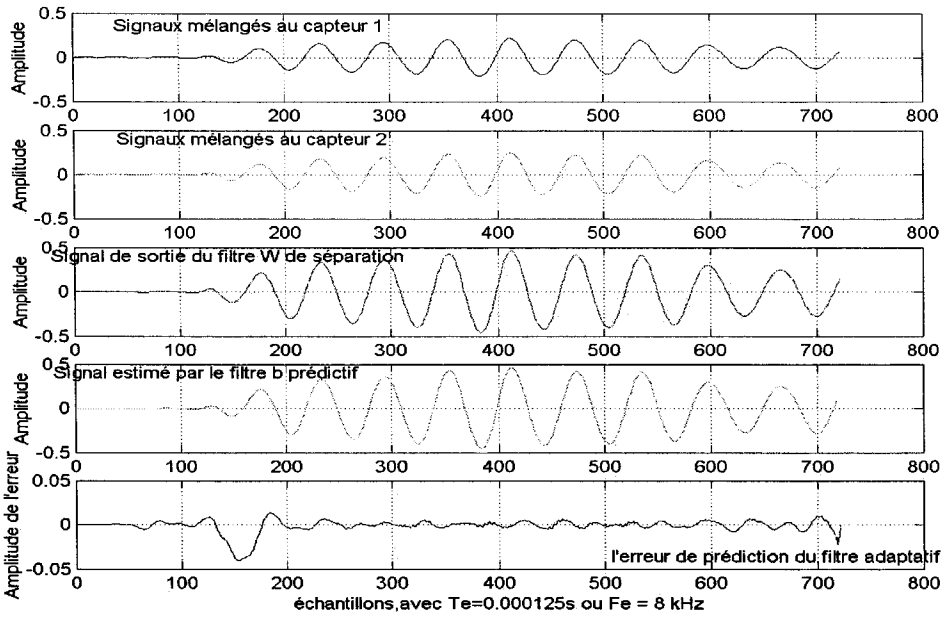

Figure 28 Extraction des composantes de la source sur f0

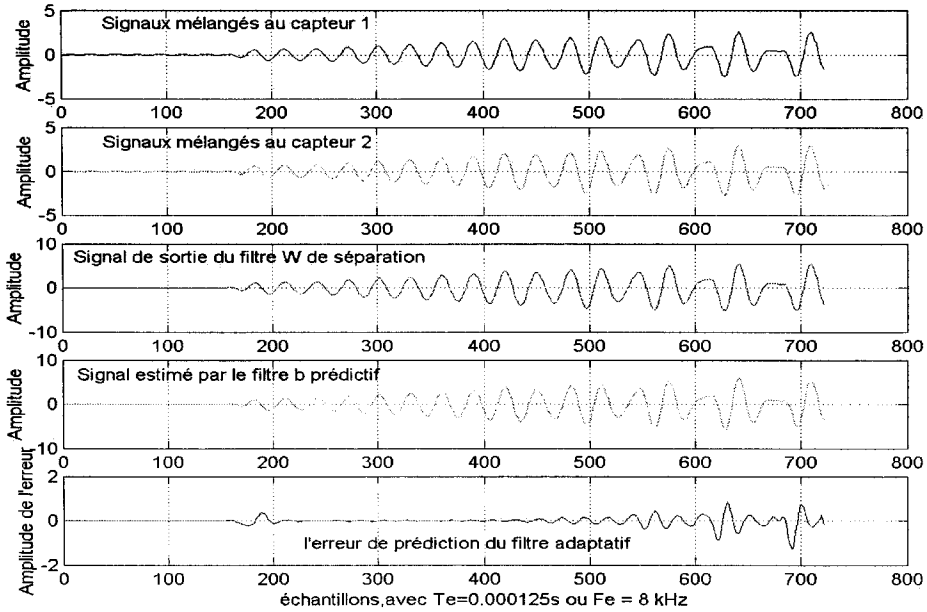

Figure 29 Extraction des composantes de la source sur 2 f0

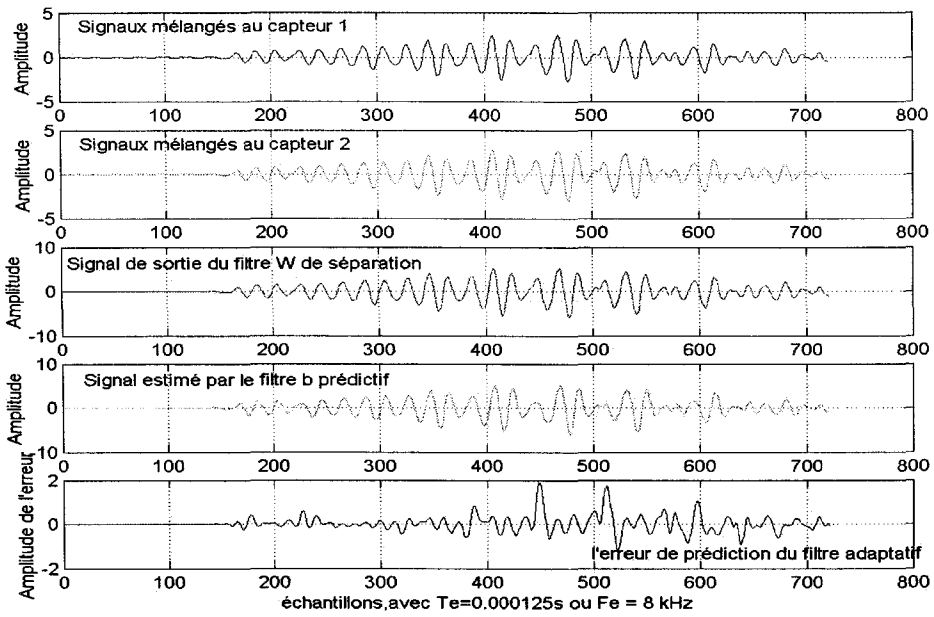

Figure 30 Extraction des composantes de la source sur 4 f0

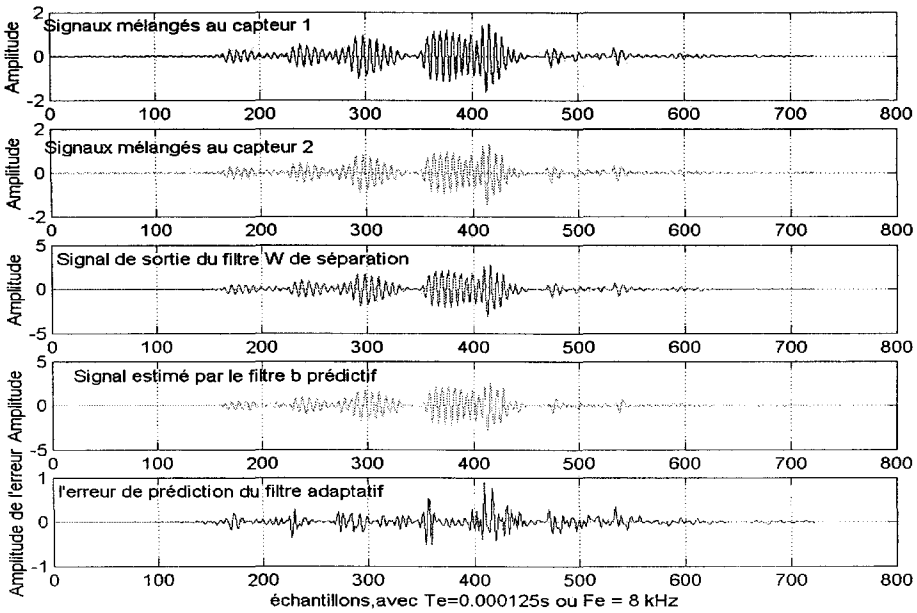

Figure 31 Extraction des composantes de la source sur 8 f0

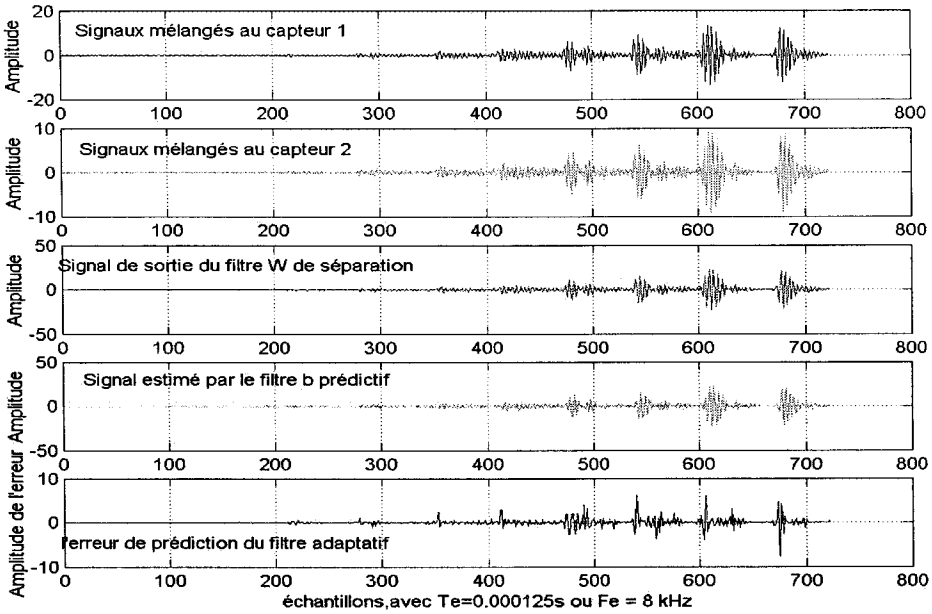

Figure 32 Extraction des composantes de la source sur 16 f0

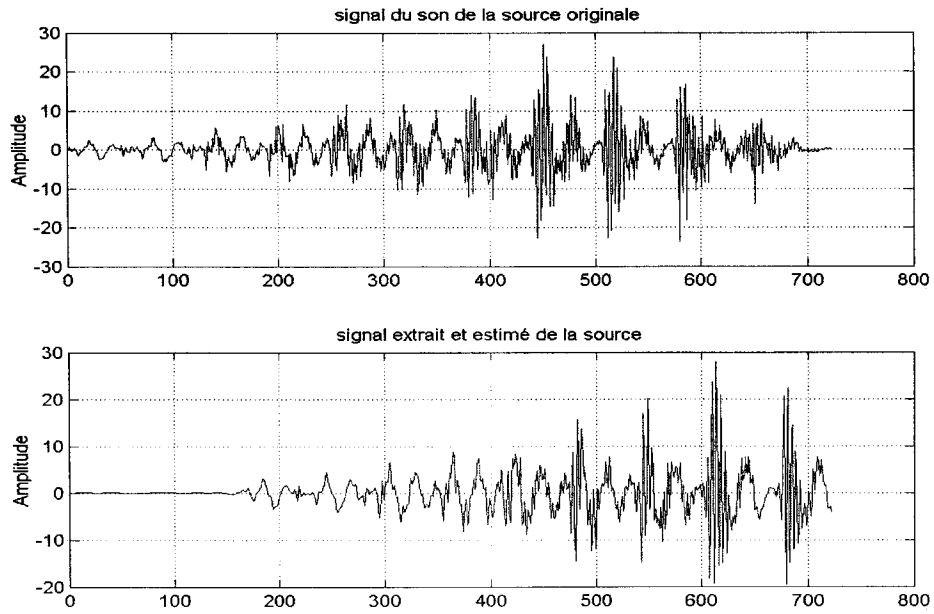

Figure 33 Sommation des composantes donnant la source estimée

CHAPITRE 5

LE PROCESSEUR DÉDIÉ AU TRAITEMENT NUMÉRIQUE

Notre application du rehaussement de la parole est destinée à être mise en œuvre sur un DSP pour toute application embarquée ou dédiée en temps réel, et notre choix s'est fixé sur le produit de Texas Instruments qui est un des principaux chefs de file dans ce domaine, mais aussi parce que ses produits de développement sont relativement conviviaux et performants et les ressources d'assistance sur internet sont utiles et accessibles. De plus, notre référence [23] nous permettait une introduction rapide sur le produit choisi, ce qui contribue à son utilisation aisée.

Le produit choisi de Texas Instruments est le DSK du TMS320C6711 qui vient avec comme matériel une platine de circuit imprimé avec ses composants et interfaces nécessaires pour la programmation, le développement et la simulation et le déverminage du DSP TMS320C6711. Son logiciel associé est le CCS qui possède une configuration IDE pour ordinateur de table ou portatif, il incorpore les principaux utilitaires de développement usuels, c'est-à-dire, éditeur pour les codes sources en langage C et en langage assembleur, le compilateur avec éditeur de liens, et les outils de déverminage inclus. Il s'insère très bien dans l'environnement Windows et autre système d'opération d'ordinateur.

Le processeur de traitement numérique TMS320C6711 est très performant pour la présente application. Travaillant avec une cadence d'horloge à 150 MHz, il peut, lorsqu'il est mené vers une optimisation maximale, accomplir selon [23] jusqu'à 900 MFLOPS ce qui se traduit par 1200 instructions par seconde sur des registres de 32 bits. L'adressage de 32 bits également permet la gestion d'une mémoire de 4 Go disposée en différents bancs.

L'entrée et la sortie des signaux se font avec un taux d'échantillonnage de 8 KHz sur des convertisseurs A/D et D/A des 16 bits. De plus il existe des interfaces en option pour se

relier à la carte mère qui ajoutent des possibilités accrues comme entrées et sorties stéréo avec possibilité d'un taux d'échantillonnage variable.

La seule limitation importante pour notre application est que nous n'avons pas deux canaux d'entrée à notre disposition, il faut donc prévoir une alternative pour simuler cette condition exigée. Dans un premier temps nous pensons faire usage de fichiers représentant les deux canaux ou faire appel à des enregistrements séparés mais alors en temps différé.

CHAPITRE 6

MISE EN ŒUVRE DES ALGORITHMES EN LANGAGE C

Les simulations faites sur Matlab nous ont servi de départ pour mettre au point le programme source en langage C de standard ANSI. La référence [23] nous a été très aidante. Nous y avons puisé des éléments de code pour l'adapter à nos besoins en particulier l'algorithme et la mise en œuvre performante de la FFT qui est utilisée intensivement pour toutes les convolutions nécessaires tant dans le filtrage numérique que dans la transformée en ondelette continue où le nombre de coefficients est important, mais également pour la transformée de Hilbert qui a été nécessaire pour définir l'enveloppe des signaux analysés.

Le programme peut être retrouvé à l'annexe II, c'est une version de déverminage développée sous environnement Borland. Cette phase de développement intermédiaire a été nécessaire compte tenu que nous ne pouvions pas avoir les interfaces dédiées aux deux canaux de traitement sur l'outil de développement DSP de Texas Instruments. Il a donc fallu simuler la saisie des deux canaux par la lecture de deux fichiers contenant l'information des signaux à traiter.

La méthode est aussi plus propice pour évaluer les performances de l'application dans sa phase de développement puisqu'elle assure un moyen répétitif de reproduire un contexte précis en observation étudiée, ce qui est exclu ou plus difficile dans le cas de l'emploi des microphones en temps réel.

Les grandes lignes des fonctionnalités de l'application peuvent se résumer ainsi : pour chaque segment prélevé du signal sur chacun des deux canaux nous avions à accomplir :

- Transformée en ondelettes dyadique réalisée par FFT sur les trois échelles
- Détermination de la fréquence fondamentale f0
- Construction du banc de filtres de type passe-bande selon la f0 retenue

- Filtration du signal par FFT, la saisie et l'obtention des sous-bandes filtrées $r_{i,k}(t)$ sur chaque canal
- Relevé de l'enveloppe $\hat{A}_{i,k}(t)$ sur les signaux des bandes filtrés par transformée de Hilbert par l'emploi de la FFT
- Génération des signaux synthétisés $Z_{i,k}(t)$ selon f0 ou kf0 modulée par l'enveloppe
- Recouvrement de l'information de la phase contenue dans chaque sous-bande filtrée par filtre de Wiener avec l'approche LMS pour livrer les signaux synthétisés avec correction de phase $x_{i,k}(t)$
- Combinaison de chaque sous-bande synthétisée des deux canaux puis traitement sur filtre adaptatif de Wiener par la matrice de séparation et la prédiction linéaire du rehausseur
- Chaque résultat précédent est sommé pour retourner le signal rehaussé en sortie

Chaque bloc fonctionnel de l'ensemble de l'application, énuméré plus haut, a été développé et testé et comparé par rapport aux résultats de la simulation sous Matlab et leur intégration dans le programme principal est terminée. Un imprimé des différentes étapes d'exécution du programme exécutable est placé à la suite de l'imprimé du code source dans l'annexe II.

La figure 34 montre un résultat intermédiaire où l'on peut observer le signal synthétisé de la première sous-bande du canal 2 à partir du signal filtré de la même sous-bande. La figure 35 montre le signal estimé à la suite des perturbations de bruit convolutif et additif subi par le signal original de la source visée.

La figure 36 montre le résultat pour le son « Matlab » bruité avec un RSB de 3 dB et convolué. Il est intéressant de remarquer que le signal estimé ne contient plus de composantes continues puisque nos bancs de filtres sont du type passe-bande.

Comme le signalaient les auteurs [10] la périodicité du signal estimé permet la perception cependant de la parole lorsqu'on écoute ce signal sous l'environnement de Matlab avec la fonction « sound.m ». Mais pour améliorer le signal estimé nous avons ajouté un filtre passe-bas contigu à la limite inférieure du premier passe-bande contenant la f0 comme fréquence centrale. Cela a donné le résultat amélioré de la figure 37.

Compte tenu de la portabilité du langage C, le programme source élaboré dans cette étude peut facilement s'adapter moyennant quelques ajustements pour tenir compte des particularités de la configuration du DSP choisi.

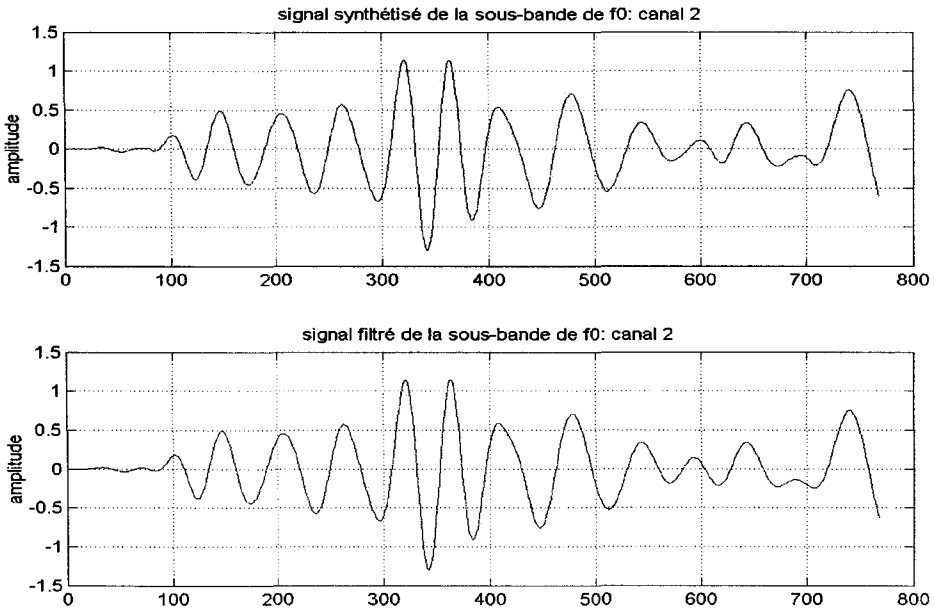

Figure 34 Signal synthétisé à partir du signal filtré sur la première sous-bande

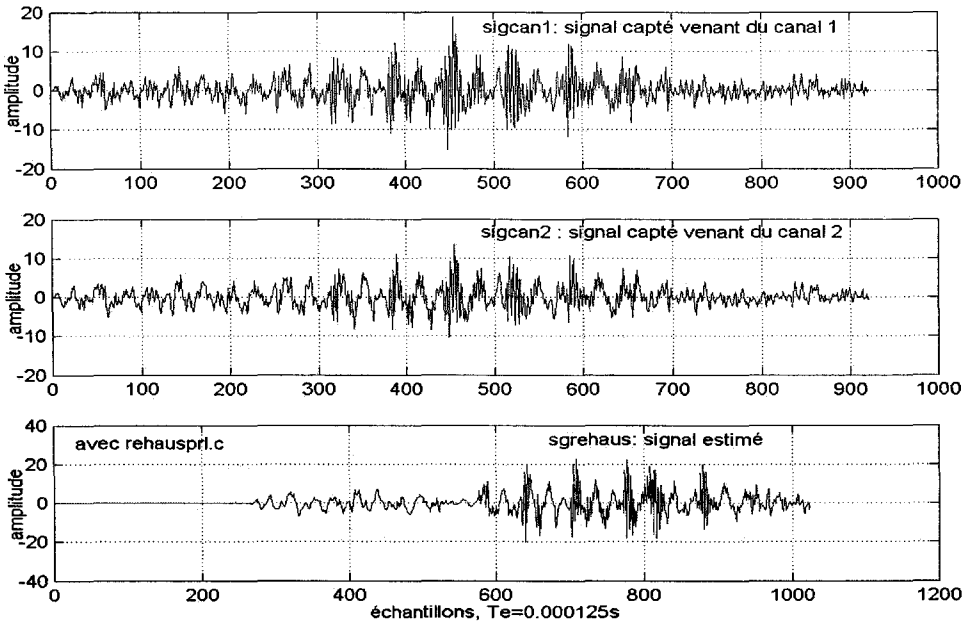

Figure 35 Signal estimé à partir des signaux captés

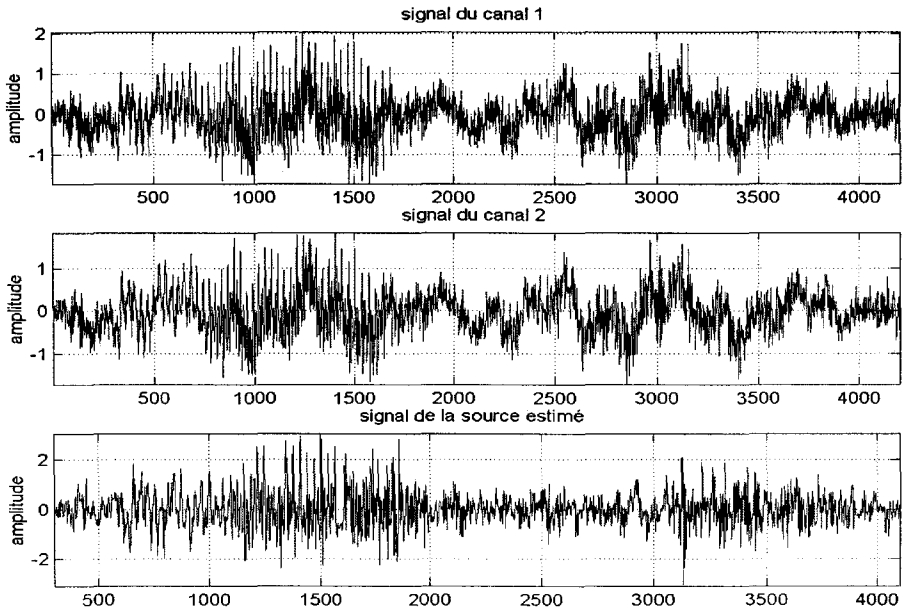

Figure 36 Estimation du son "Matlab" sans composantes continues

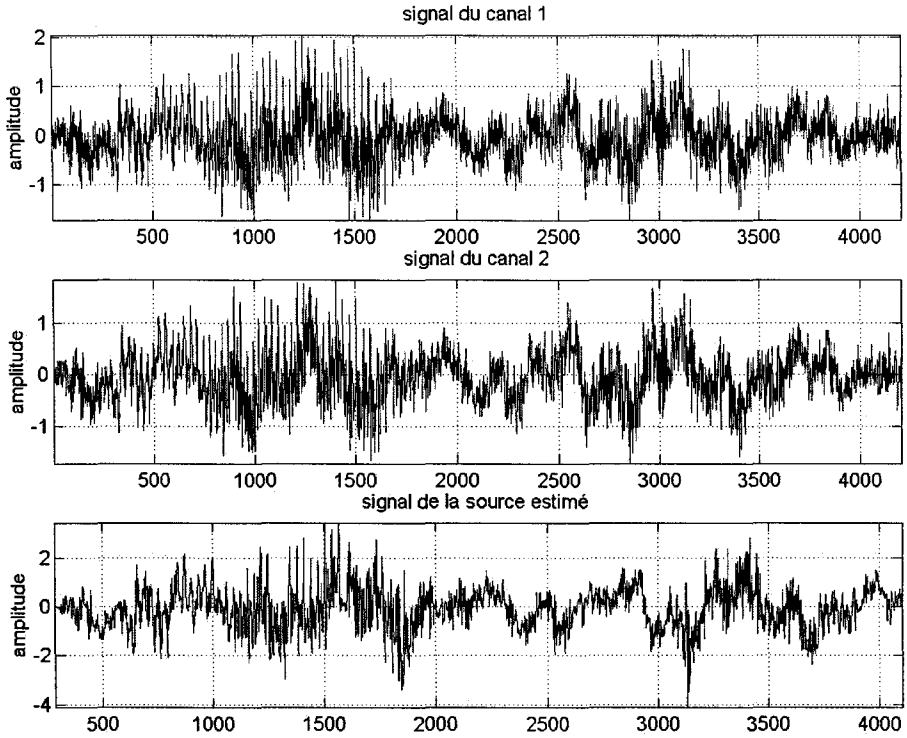

Figure 37 Estimation du son "Matlab" avec composantes continues

DISCUSSION ET INTERPRÉTATION DES RÉSULTATS

Les figures 28 à 32 donnent les composantes fréquentielles extraites et estimées de la source principale venant des deux canaux et la figure 33 donne la sommation de toutes les contributions pour aboutir au signal estimé qui est mis en comparaison par rapport au signal original. La figure 35 donne le résultat obtenu sous l'application développée en C à partir des signaux captés venant des deux canaux.

On observe autant dans la simulation que dans l'application en langage C que les composantes fréquentielles élevées du signal original sont atténuées ou absentes sur le signal estimé, cependant la signature du signal sur la portion recouvrée paraît préservée en dépit des bruits additifs et convolutifs subis.

Le délai observé sur la réponse du signal estimé est inhérent au traitement de la filtration de l'analyse fréquentielle et adaptative.

Le rehaussement du signal, ici la parole est évidemment un processus d'optimisation qui mène à l'impossibilité de la restitution fidèle idéale de la source originale. Il faut alors mettre en œuvre certaines méthodes objectives ou subjectives pour évaluer le degré de rehaussement apporté.

Généralement en traitement de signal, on a recours à la mesure du rapport signal sur bruit, cependant selon plusieurs avis d'experts du signal de la parole ce critère n'est pas toujours révélateur dans ce domaine; il arrive qu'un bon RSB d'un signal de la parole aboutisse à une intelligibilité médiocre de cette dernière et d'autres fois, un signal avec un faible RSB donne une bonne intelligibilité.

Une approche pseudo-objective est souvent préconisée. Elle se réfère à l'évaluation de la qualité de l'information de la parole et recommandée par la norme G.107 de l'ITU. C'est

la méthode qui met en place une échelle pondérée de notes subjectives comprises entre 1 et 5 donnant une appréciation de la qualité perçue par l'auditeur. Elle est connue sous l'acronyme MOS (pour Mean Opinion Score). Le tableau II présente ce pointage.

Tableau II

Échelle du MOS

Échelle du MOS	Valeur qualitative
5	Excellent
4	Bon
3	Moyen
2	Dégradé
1	Mauvais

La récupération des résultats pour les différents signaux estimés a été faite à l'aide de la fonction « sound » de Matlab. En excluant l'effet de troncature du signal à la fin de l'écoute, on peut dire que les signaux nous semblent bien reconnaissables aux échelles 3 à 4 du MOS et ceci malgré le fait que le bruit convolutif ait été ajouté sur les sources originales.

On devrait donc avoir un résultat amélioré avec le traitement en temps réel dans la mise en œuvre du DSP, car selon les auteurs [10] l'analyse spectrale à court terme des bancs de filtres devrait réduire voir annuler l'effet convolutif, ce qui n'a pas été le cas dans notre simulation ou dans nos essais avec l'application en C. Dans ces deux cas l'effet convolutif a été gardé.

Une autre approche plus quantitative cette fois-ci mais tout aussi utilisée se base sur le critère de la grandeur de l'erreur estimée présente dans les processus impliqués amenant au rehaussement de la parole.

Sur les figures 28 à 32 l'erreur se trouve sur le dernier graphique de chaque tranche fréquentielle liée à la fondamentale f0 ou à une de ses harmoniques. On constate que l'erreur absolue croît au fur et à mesure que la bande fréquentielle augmente dans le domaine spectral où la vitesse de convergence ne peut suivre les changements rapides des signaux. Ce qui se démontre bien dans le résultat global obtenu sur la figure 33 et plus prononcé sur la figure 35.

Une tentative d'explication peut être ici soumise. Deux éléments des algorithmes mis en œuvre par notre application sont, selon nous, imputables à ce résultat.

Lors de la conception des filtres passe-bande de chaque banc de filtre, pour respecter d'une part, la limite imposée par la fréquence de Nyquist et d'autre part, pour conserver constant le facteur de qualité de chaque passe-bande nous avons plus ou moins délibérément omis la partie haute du spectre selon la f0 trouvée voir à titre d'exemple la figure 18 sur ce point.

Nous avons constaté que l'ajout d'un filtre passe-bas avant le premier passe-bande sur la f0 améliorait le rehaussement en introduisant la composante continue du signal, comme le montre les résultats sur la figure 37 par rapport à la figure 36.

L'autre cause vient du prédicteur linéaire dans la partie de l'ACI où il est vu comme un rehausseur adaptatif de ligne dont le nombre de coefficients est proportionnel à la période de la f0 ou d'une de ses harmoniques ainsi donc la traçabilité des variations rapides sur les composantes de fréquence élevée ne peut être garantie ou apprise.

De plus, comme nous l'avons vu précédemment, toute l'approche adaptative basée sur les algorithmes du LMS repose sur l'hypothèse de la stationnarité des signaux traités.

Nous savons cependant que dans notre cas, le signal de la parole ne possède pas toujours cette particularité.

CONCLUSION

La simulation sous Matlab comme l'application exécutée en langage C de la solution choisie pour le rehaussement de la parole montrent selon les premiers résultats, un potentiel de performance qui s'insère dans une démarche au sens large d'une analyse en composantes indépendantes sans impliquer toutefois la complexité relative en terme de traitement de calcul qui lui est inhérente. Elle démontre ainsi que des algorithmes relativement simples et usuels peuvent traiter une démarche d'ACI à condition de respecter selon les auteurs [10], les hypothèses que le signal de la parole à rehausser possède la plus grande énergie et que certaines de ses caractéristiques ne sont pas complètement inconnues. En général la démarche d'ACI n'est pas tenue de respecter ces conditions, mais c'est souvent au prix d'une plus grande complexité que ce que nous avons présenté comme solution.

Bien que la mise en œuvre de l'application sur le DSP n'ait pas pu être entreprise, globalement les essais sur chaque bloc fonctionnel ont été vérifiés et chaque bloc se comportait conformément à la simulation. Au niveau du programme en C exécutable sur environnement PC, l'ensemble des algorithmes fonctionne comme la simulation faite sur Matlab. Cela permet d'envisager la mise en œuvre de l'application sur DSP éventuellement.

RECOMMANDATIONS

La solution préconisée par les auteurs et les adaptations apportées par le présent document n'ont pu être intensivement testées dans la simulation comme dans sa réalisation sur le programme exécutable en C pour l'exporter sur DSP en partie, compte tenu de l'envergure d'un projet de ce genre.

Bien que les premières simulations révèlent un rehaussement possible, l'application n'a pu être évaluée pour connaître ses limites fonctionnelles tant sur les types de signaux testés que sur les différents niveaux de bruits perturbateurs qu'ils soient additifs ou convolutifs.

Même si les premiers résultats de simulation peuvent susciter l'intérêt pour certaines applications dans des conditions et hypothèses vues dans la conclusion, des essais sur d'autres types de filtres convolutifs simulant la réverbération et différents niveaux de bruits seraient souhaités en temps réel sur un processeur dédié au traitement numérique pour mieux apprécier et connaître les limites de l'application choisie pour le rehaussement de la parole.

ANNEXE 1

Programmes de Matlab

Nous insérons ici, la liste des programmes sources de Matlab qui nous ont servi pour la simulation.

```
%-----------------------------------------------------------------%
% Auteur: Antoine Munoz
% nom du fichier: sigbruit.m
% remis le 4 novembre 2003
% aux professeurs : M. Marcel Gabrea et M. Christian Gargour
% Pour le mémoire 2003-4.
%-----------------------------------------------------------------%
clc,clear all,close all;
%----------  récupération du signal plus bruit  ----------------%
load dmwe m; signalwe = m(2,:);
bruit = randn(1,length(signalwe)); % création du bruit normalisé
%-----------------------------------------------------------------%
facteur = 1 ; delta = 0.001 ;
% SNRD = 0;   % valeur  du SNR recherchée: 0 5 10 15 20 25 30 dB
SNRD = 5;
% SNRD = 10;
% SNRD = 15;
% SNRD = 20;
% SNRD = 25;
% SNRD = 30;
%---------- Evaluation du RSB en dB selon le SNR demandé ----------%
rsb = 10 * log10( var(signalwe) / var(bruit) );

if rsb > SNRD
  while rsb >= SNRD
   facteur = facteur + delta;
   bruit = facteur * bruit;
   rsb = 10 * log10( var(signalwe) / var(bruit) );
   end
end

if rsb < SNRD
  while rsb <= SNRD
   facteur = facteur - delta;
   bruit = facteur * bruit;
   rsb = 10 * log10( var(signalwe) / var(bruit) );
   end
end
%---------------------- Signal bruité ----------------------------%
```

```
wepbruit = signalwe + bruit;
% save weSNR0 sigsom;
% save weSNR5 sigsom;
% save weSNR10 sigsom;
% save weSNR15 sigsom;
% save weSNR20 sigsom;
% save weSNR25 sigsom;
save weSNR5 wepbruit;
%-----------------------------------------------------------------%
subplot(311)
plot(bruit)
title(['bruit pour obtenir un SNR de ',num2str(SNRD),' dB'])
ylabel('grandeur')

subplot(312)
plot(signalwe)
title('Signal original')
ylabel('grandeur')

subplot(313) % gtext(['le SNR est: ',num2str(rsb),' dB'])
plot(wepbruit)
title(['Signal bruité avec un SNR de ' ,num2str(rsb),' dB'])
xlabel('échantillons, Te=0.000125s ou Fe = 8 KHz')
ylabel('grandeur')
%--------------------Fin du Programme sigbruit.m-------------------%


%-----------------------------------------------------------------%
% programme pour évaluer le bruit convolutif générer par des filtre RIF
% remis aux professeurs : M. Marcel Gabrea et M. Christian Gargour
% Réalisé par M.Antoine Munoz
%-----------------------------------------------------------------%
% nom du programme: mixconvolFIR4.m
% date 15 17 juin 2005
% date 10 juin 2002
%-----------------------------------------------------------------%
clc,clear all,close all;
Fs = 8000; %fréquence d'échantillonnage du dsp codec ad535
%-----------------------------------------------------------------%
%----------   récupération du signal plus bruit  ----------------%
% bruitwe = randn(1,length(signalwe)); % création du bruit normalisé
% bruitML = randn(1,length(signalML)); % création du bruit normalisé
%load bruitML; BML=bruitML; load mtlb; signalML=mtlb';
%load bruitwe; BWE=bruitwe; load bruitwe; bruit=bruitwe;
```

```
load dmwe m; signalwe = m(2,:); sigin=signalwe; % sigin=signalML;

load weSNR5 wepbruit; %signal we bruité aavec RSB=5dB
bruit = wepbruit;



%---------------------------- 1 ----------------------------------%
coeffB1 =[0.05903, -0.05268 -0.06256 0.12734 0.4901 0.142796 -0.051458 -0.0159489
0.010203 0.00012 ];

coeffB2 =[0.046520 -0.018077 0.03601 0.180955 0.36475 0.220127 -0.077205 -
0.024521 0.012375 -0.000011];
%---------------------------- 2 ----------------------------------%
coeffB3 =[0.02  -0.0089714083   -0.037772542   0.0080977106  -0.18764424
0.082851192 0.41651176  0.082851192  -0.08764424        0.0080977106  -
0.037772542  -0.0089714083   -0.0015340882 ];
coeffB4 =[0.05          0.021752101 -0.082052484  0.16251428      0.46308498
0.16251428     -0.032052484  0.02175210 -0.0037564   0.0    0.0   0.0  0.0 ];
%---------------------------- 3 ----------------------------------%
coeffB5 =[0.0105e+00  -1.0907862e-02  -9.5862249e-03  -5.4309726e-02  1.4528681e-
01   4.9461315e-01  1.4528681e-01 -5.4309726e-02  -9.5862249e-03  -1.0907862e-02
0.0105e+00  ];
coeffB6 =[0.035e+00  -1.1959607e-02  -1.7325158e-02  2.4047196e-01   4.0813720e-01
2.4047196e-01  -1.7325158e-02  -1.1959607e-02  -8.7180096e-03   0.0000000e+00
0.0000000e+00 ];
%---------------------------- 4 ----------------------------------%
coeffB7 =[-1.6543198e-03   6.4353332e-03   5.6152413e-02  -2.0557229e-01  -
0.3988439e-01   4.338056e-01  -0.3988439e-01  -2.0557229e-01   5.6152413e-02
6.4353332e-03  -1.6543198e-03 ];
coeffB8 =[-3.4046542e-03  1.0728125e-03  -1.2572478e-02  -1.0543943e-01
1.1782215e-01   4.5914059e-01   1.1782215e-01  -1.0543943e-01  -1.2572478e-02
1.0728125e-03  -3.4046542e-03];
%---------------------------- 5 ----------------------------------%
coeffB9 =[0.3549 0.4035 -0.1726 -0.05879];
coeffB10 =[0.3778 0.3423 -0.1064  0.06176];
%---------------------------- 6 ----------------------------------%
coeffB11 =[0.021e+00  -1.0707862e-02  -9.0862249e-03  -5.309726e-02  -1.4528681e-
01   5.61315e-01  -1.4528681e-01 -5.309726e-02  -9.0862249e-03  -1.0707862e-02
0.021e+00];
coeffB12 =[0.037e+00  -2.1759607e-02  -1.7325158e-03  -1.4947196e-01   4.50e-01  -
1.4947196e-01  -1.7325158e-03  -2.1759607e-02  -3.7180096e-03   0.0000000e+00
0.0000000e+00];
```

```
%-------------------------------------------------------------------%
%-------------------------------------------------------------------%
% graphiques de la fonction de transfert

for i=1:6

   switch i
   case 1
     vect1=coeffB1;vect2=coeffB2;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
%       save fx1FIR1 x1; save fx2FIR1 x2;
     %save fx1FIR1t x1 -ascii -double -tabs;
     %save fx2FIR1t x2 -ascii -double -tabs;
   case 2
     vect1=coeffB3;vect2=coeffB4;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
     %save fx1FIR2 x1; save fx2FIR2 x2;
   case 3
     vect1=coeffB5;vect2=coeffB6;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
     %save fx1FIR3 x1; save fx2FIR3 x2;
   case 4
     vect1=coeffB7;vect2=coeffB8;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
     %save fx1FIR4 x1; save fx2FIR4 x2;
   case 5
     vect1=coeffB9;vect2=coeffB10;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
     %save fx1FIR5 x1; save fx2FIR5 x2;
   case 6
     vect1=coeffB11;vect2=coeffB12;
     x1 = filter(vect1,1,sigin);
     x2 = filter(vect2,1,sigin);
     %save fx1FIR6 x1; save fx2FIR6 x2;
   end

F = 0: (Fs-0.001)/2;
H = freqz(vect1, 1, F,Fs);
H1 = freqz(vect2, 1, F,Fs);
```

```
%-------------------------------------------------------------------%
%figure(i);
figure;
subplot(321);plot(F,abs(H)./ max(abs(H)),'k');
title('module du canal 1');
grid on; ylabel('Module'); %xlabel('Hz');
subplot(323);plot(F,20*log10(abs(H)./ max(abs(H))),'k');title('module du canal 1 en
dB');
grid on; ylabel('Module en dB');%xlabel('Hz');
subplot(325);plot(F,180*angle(H)/pi,'k');title('déphasage du canal 1 ');
grid on;xlabel('Hz'); ylabel('Phase en degre');

subplot(322);plot(F,abs(H1)./ max(abs(H1)),'k');title('module du canal 2');
grid on;ylabel('Module');%xlabel('Hz');
subplot(324);plot(F,20*log10(abs(H1)./ max(abs(H1))),'k');title('module du canal 2 en
dB');
grid on; ylabel('Module en dB');%xlabel('Hz');
subplot(326);plot(F,180*angle(H1)/pi,'k');title('déphasage du canal 2 ');
grid on;xlabel('Hz'); ylabel('Phase en degre');

%-------------------------------------------------------------------%
%figure(i+6);
figure;
subplot(411);plot(x1);title('x1');ylabel('grandeur');
grid on;
subplot(412);plot(x2);title('x2');ylabel('grandeur');grid on;
subplot(413); plot(sigin);title('Signal original bruité');
% gtext(['Signal avec bruit pour obtenir un SNR de ' ,num2str(SNRD),' dB ', 'sur filtre
RIF no ',num2str(i)])
grid on;
subplot(414); plot(signalwe);title('Signal original');grid on;
xlabel('échantillons, Te=0.000125s ou Fe = 8 KHz');
ylabel('grandeur');

end % du for i=1:6
%-------------------------------------------------------------------%
%réponse indicielle du filtre
%-------------------------------------------------------------------%
%in(1:20)=0;
%n = [-20:40];
%subplot(2,1,1);stem(n,in);
%out = filter(coeffB,1,in);
%subplot(2,1,2);stem(n,out);
```

```
%le temps de réponse stabilisé correspond à N*Ts
%-------------------------Fin du Programme mixconvolFIR4.m----------------------%


% nom du fichier: freqf0.m ********* Auteur: Antoine Munoz
% travail pratique, date de création et mise a jour:
% 6, 8, 10 13 14 20 23 29 novembre, % 1,4,5,6,11,13 décembre 2004
% emploie la fonction pkpick.m
% M. Christian Gargour et M. Marcel Gabrea % mémoire 2004.
%***mise a zero des variables,figures,des lignes de commande de matlab**%
clear all;close all;clc;
%*****************************initialisation*****************************
Fe=8000;trame=256;debutrame=1;fintrame=trame;derniervalmax=0;
MMX=[];suitedeslocs=[];periode=[];MM=[];TesT2=[];
%*************fichiers des ondelettes de trois échelles*****************
load lintgauss23 ; load lintgauss24 ; load lintgauss25 ;
h23 = intgauss23(2,:); h24 = intgauss24(2,:); h25 = intgauss25(2,:);
%*******************lecture d'un fichier mat*************************
% load mtlb; allx=mtlb';  %  x=x(1:1024); x=x(1:256) x=x(1:512)
% x=allx(1:3936);
% load we ; x = we; allx=we;% données transferées dans le vecteur x


% canal 1
% load fx1FIR1 x1; x = x1; allx = x1;

% canal 2
load fx2FIR1 x2; x = x2; allx = x2;


delta = ceil(trame-length(h25)/2);
% nbretram = ceil(length(x)/(trame+2*delta)); % fixe le nombre d'itération
nbretram = ceil(length(x)/(trame)); % fixe le nombre d'itération
%*****************************************************************
for z=1:nbretram   %#####################################################
    if z==1
       tranche=x(debutrame:fintrame + delta);
       trancheinf = tranche(fintrame-delta:fintrame + delta );
    else
        if (fintrame + delta)<=length(x)
           tranche=x(debutrame + delta : fintrame + delta);
        else
           tranche=x(debutrame + delta : length(x));
        end;
```

```
        tranche=[trancheinf tranche];% concaténation
        trancheinf = tranche(end-2*delta : end);
    end;
%traitement de la transformée en ondelette continue aux trois échelles
coeffh23 = -sqrt(2^3)*diff(conv(tranche,h23));
coeffh24 = -sqrt(2^4)*diff(conv(coeffh23,h24));
test1 = -sqrt(2^5)*diff(conv(coeffh24,h25));
test2 = wkeep(test1,length(tranche)); %élimine les bords non pertinents
Bornesup=length(tranche);
%**********repérage des maxima dans le temps de la trame**************
[valmax, locmax]=max(test2);
if valmax > derniervalmax derniervalmax=valmax; end; % pour valmax global
seuil = 0.20 * derniervalmax; %seuil=0.2*valmax,si on veut un valmax local
[peaks, locs] = pkpick( test2, seuil );
fini=length(locs)-1;
for nn=1:fini
  start=locs(nn);stop=locs(nn+1);valmin = min(test2(start:stop));
  if valmin >0
     if peaks(nn) > peaks(nn+1) peaks(nn+1)=0; locs(nn+1)=0;
     else   peaks(nn)=0; locs(nn)=0; end
  end
end
locs(find(locs==0))=[];peaks(find(peaks==0))=[];%retire les fluctuations
%*****************************************************************
if z==1
   bsup=max(find(locs <=trame));
   locs=locs(1:bsup); test2=test2(1:trame);
   locsglobal = locs + ((z-1)*trame);
else
   binf=min(find(locs>=delta+1));
   bsup=max(find(locs <=Bornesup-delta));
   locs=locs(binf:bsup); test2=test2(delta+1:end-delta);
   locsglobal = locs + ((z-1)*trame)-delta;
end
suitedeslocs=[suitedeslocs locsglobal'];
figure  % affichage des résultats**********************************
%=================================================
plot(test2);
title(['transformée en ondelettes de la trame no:',num2str(z),', de ',num2str(((z-
1)*trame)+1),' a ',num2str(((z-1)*trame)+256)]);
xlabel('échantillons, Te=0.000125s')
ylabel('amplitude'); grid on; TesT2=[TesT2 test2];
%************mise a jour de la trame et de ses nouvelles bornes********
debutrame = 1 + fintrame;  % pause
```

```
if (fintrame + trame) > length(x)   fintrame=length(x);
else fintrame = fintrame + trame;   end;
%*************************************************************
end % fin de for z=1:nbretrame%##########################################
for kk=2:length(suitedeslocs)
   periode = [periode suitedeslocs(kk)- suitedeslocs(kk-1)]; end;
periode=[periode periode(end)];%obligé pour colonnes égales a chq ligne
frequence = Fe.*(1./periode); MMX=[frequence;suitedeslocs];
% valeur du traitement global pour fin de comparaison
% load donneeMM MM; valeurperiode=MM(2,1:length(suitedeslocs));
% valeurfrequence=MM(1,1:length(suitedeslocs));
% MMTotal=[valeurfrequence;valeurperiode];
figure  % affichage des résultats**********************************
subplot(1,1,1)%=================================================
===
plot(allx); title('signal original');xlabel('échantillons, Te=0.000125s')
ylabel('amplitude'); grid on
%
subplot(2,1,2)%=================================================
===
% plot(x); title('signal traité');xlabel('échantillons, Te=0.000125s')
% ylabel('amplitude'); grid on; figure %*******************************
%
subplot(2,1,1)%=================================================
===
% stairs(MMTotal(2,:),MMTotal(1,:));hold on; stem(MMTotal(2,:),MMTotal(1,:));
% title('fréquence déduite sur traitement global');
% ylabel('frequence'); xlabel('échantillons , Te=0.000125s');grid on;
figure
subplot(1,1,1)%=================================================
===
stairs(MMX(2,:),MMX(1,:));hold on; stem(MMX(2,:),MMX(1,:));
title(['fréquence déduite sur traitement par trame de ',num2str(trame),' echant']);
ylabel('frequence'); xlabel('échantillons , Te=0.000125s');grid on;
figure % *************************************************************
plot(TesT2); title('signaux de test2 concaténés');xlabel('échantillons, Te=0.000125s')
ylabel('amplitude'); grid on
%******************** ancien Fichier freqfond11.m **********************
% v=TesT2(2500:3000);
% [valmax, locmax]=max(v);
% seuil=.2*valmax;
% [peaks, locs] = pkpick( v, seuil );
%******************** Fin du Fichier freqf0.m **********************
```

```
%-------------------------------------------------------------------------------%
% nom du fichier: bancfiltre.m              ancien: bcFIRTH.m
% Pour le mémoire 2004.                    Auteur: Antoine Munoz
% 24 juin 2004
% remis le 15 mars 2004
% aux professeurs : M. Marcel Gabrea et M. Christian Gargour
% Etapes pour construire le banc de filtres passe-bande.
% fonctions appelées: I0.m, ideal_lp.m
% variables globales: f0
%-------------------------------------------------------------------------------%
clc,clear all,close all;
%---------fréquence d'échantillonnage du dsp codec ad535------------%
Fs = 8000; FNyq = Fs/2;    %fréquence de Nyquist
%-----------------------------------------------------------------%
% valeur possible de f0 et multiples: 140 280 560 1120 2240 4480
f0=128          % exemple de fréquence fondamentale en Hz
alpha=f0/sqrt(2)  % limite inferieure de la premiere bande
beta=2*alpha      % limite superieure de la premiere bande
%--------------------- Bande de transition -------------------------%
deltaomega = pi*1.5*(f0-alpha)/FNyq % largeur de la bande de transition
%-------------------L'ondulation minimale permise-------------------%
coeffondul = 0.01 %0.05 coefficient d'ondulation
OnduldB=-20*log10(coeffondul) % ondulation en dB
%-----------------------L'ordre N du filtre-------------------------%
% si l'ordre N est un nombre pair, alors le nombre de termes M de la
% fenetre sera impair et nous aurons une réponse impulsionnelle
% symétrique garantissant un déphasage linéaire (N+1=M)
N=(OnduldB-8)/(2.285*deltaomega);
N=ceil(N);
if mod(N,2)~=0
   N=N+1;
end;N
%-----------la valeur de Beta pour la fenetre Kaiser ---------------%
if OnduldB < 21
       Beta =0
      elseif OnduldB >50
       Beta = 0.1102*(OnduldB-8.7)
      else
       Beta = 0.5842*(OnduldB-21)^0.4 + 0.07886*(OnduldB-21)
      end
%-----------------Calcul de la fenetre de Kaiser------------------%
%la fonction Io(Beta),voir I0.m
windKaiser=[]; Iden=I0(Beta); n=[0:1:N];%length(n)= N+1 pour symétrie
for j=1:length(n)
```

```
   valeur=Beta*sqrt(1-((n(j)-N/2)/(N/2))^2);
   windKaiser=[windKaiser I0(valeur)/Iden];
end
%----Calcul du nombre de passe bande dans le banc de filtres------%
Nbredefilt=1;     % nbre de filtre passe bande dans le banc
flim =[alpha beta]; % valeurs limites des bandes de filtre
while 2*beta < FNyq
   beta = 2*beta;
   flim =[flim beta];
   Nbredefilt = Nbredefilt+1;
end
Nbredefilt
flim
wci = pi/FNyq.*flim
kfcentre=[];
for k=0: Nbredefilt-1
   kfcentre=[kfcentre 2^k*f0];
end; kfcentre
%-Calcul de la réponse impulsionnelle du banc de filtres passe-bande-%
F = 0:(Fs-0.001)/2;
HF=zeros(Nbredefilt,N+1);

for pb=1:Nbredefilt
   hc= ideal_lp(wci(pb+1),N+1) - ideal_lp(wci(pb),N+1);
   hf=windKaiser.*hc;
   HF(pb,:) = hf;
   figure(1);%-------------figure----------------
   hold on;
   H = freqz(hf, 1, F, Fs);
   plot(F,abs(H));
end
title(['Banc de filtres pour f0 = ',num2str(f0),' Hz et un ordre N =',num2str(N)])
xlabel('fréquence en Hz, Te=0.000125s, fenetre de Kaiser')
ylabel('Amplitude')
%----------  partie du filtrage du banc de filtres  ----------------%
%-----  récupération des signaux observés Xi pour filtrage  --------%
load fx1FIR1 x1; load fx2FIR1 x2; X1=x1; X2=x2;

signalfiltre1 = zeros(Nbredefilt,length(X1));
signalfiltre2 = zeros(Nbredefilt,length(X2));
%-----------------------------------------------------------------%
for pb=1:Nbredefilt
signalfiltre1(pb,:) = filter(HF(pb,:),1,X1);
signalfiltre2(pb,:) = filter(HF(pb,:),1,X2);
```

```
end
%-------------------------------------------------------------------%
%transformée de hilbert : on va chercher l'enveloppe des signaux filtrés
%-------------------------------------------------------------------%
for pb=1:Nbredefilt
   THr1k(pb,:)= hilbert(signalfiltre1(pb,:));
   THr2k(pb,:)= hilbert(signalfiltre2(pb,:));
   modTHr1k(pb,:) = abs(THr1k(pb,:));
   modTHr2k(pb,:) = abs(THr2k(pb,:));
end
%-------------------------------------------------------------------%
for pb=1:Nbredefilt
   figure
   subplot(411)
   plot(signalfiltre1(pb,:))
   title(['signal X1 filtré par le banc de filtre no ',num2str(pb)])
   ylabel('Amplitude');grid on
   subplot(412)
   plot(signalfiltre2(pb,:))
   title(['signal X2 filtré par le banc de filtre no ',num2str(pb)])
   ylabel('Amplitude');grid on
%-------------------------------------------------------------------%
   subplot(413)
   plot(modTHr1k(pb,:))
   title(['enveloppe du signal X1 filtré par le banc de filtre no ',num2str(pb)])
   ylabel('Amplitude');grid on
   subplot(414)
   plot(modTHr2k(pb,:))
   title(['enveloppe du signal X2 filtré par le banc de filtre no ',num2str(pb)])
   ylabel('Amplitude')
   xlabel('échantillons,avec Te=0.000125s ou Fe = 8 kHz');grid on
end
%-------------------------------------------------------------------%
save fichsf1 signalfiltre1; save fichmodTHr1k modTHr1k;
save fichsf2 signalfiltre2; save fichmodTHr2k modTHr2k;
%----------------------Fin du Programme bancfiltre.m-------------------%


%-------------------------------------------------------------------------%
% Nom du fichier: ica.m                    ancien:adaptlms4.m
% pour le mémoire 2004.                  Auteur : Antoine Munoz
% 26 juin 2004
% 2 mai 2004,
% professeurs : M. Marcel Gabrea et M. Christian Gargour
```

```
%-------------------------------------------------------------------------%
clc,clear all,close all;
%---------fréquence d'échantillonnage du dsp codec ad535------------%
Fs = 8000; FNyq = Fs/2;    % fréquence de Nyquist
f0=128                  % 140 exemple de fréquence fondamentale en Hz
global hlms;global mulms;global xlms;global puiss;global xplms;global xmlms;
%-------------------------------------------------------------------------%

% load fichsf1 signalfiltre1; load fichmodTHr1k modTHr1k;
% [pb nbredata] = size(signalfiltre1);

load fichsf2 signalfiltre2; load fichmodTHr2k modTHr2k;
[pb nbredata] = size(signalfiltre2);

% synth = zeros(pb,nbredata);
synth2 = zeros(pb,nbredata);

f0in=f0;

for kk=1:pb
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
f0 = 2^(kk-1)*f0in;

% d=signalfiltre1(kk,:); % signal désiré de référence
% enveloppe=modTHr1k(kk,:);

d=signalfiltre2(kk,:); % signal désiré de référence
enveloppe=modTHr2k(kk,:);

for t=1:length(enveloppe);
   Zcplxp(t) = enveloppe(t).*(exp(2*pi*f0*t*j/Fs));
   Zcplxm(t) = enveloppe(t).*(exp(-2*pi*f0*t*j/Fs));
end
%-----------------filtre adaptatif de Wiener pour retrouver la phase----------------%
% N nombre de coefficients du filtre ou ordre du filtre
N=1; % valeur minimale N=1
pas = 0.8; % valeur de départ; algodulms a un pas adaptatif
vecthlms=[];
lmsinicmplx(N,pas);
for i=1:length(d)
   xpd=Zcplxp(i); xmd=Zcplxm(i); dd= d(i);
   [y,h,ede,p] = algcmplxdulms(xpd,xmd,dd);
   spm(i)=y;
```

```
  hlms = h;
  epm(i)=ede;
  vecthlms=[vecthlms h];
end; erreurmax=max(diag(epm'*epm)),p % epm nombres complexe,p=pas
%-------------------------------------------------------------------------------------%
coefC=vecthlms(1,:); phi=angle(coefC);
%---------------signal v synthétisé avec la phase= f0 et déphassage phi---------------%
t=1:length(coefC);
v(t)=2*enveloppe(t).*sqrt((real(coefC(t))).^2 +(imag(coefC(t))).^2).*cos((2*pi*f0*t/Fs)-
phi(t));
u(t)=2*sqrt((real(coefC(t))).^2 +(imag(coefC(t))).^2).*cos((2*pi*f0*t/Fs)-phi(t));
%------------------------------1111111111111111111111------------------------------%
figure;
subplot(511); plot(d,'g'); % plot(real(d),'g');
title(['Figure#1 Signal de référence pour le filtre adaptatif c''est la sortie banc de filtre no
',num2str(kk)])
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------------------%
subplot(512); plot(real(Zcplxp+Zcplxm),'b');
title('Figure#2 Signal Zcplxp+Zcplxm sur l''entrée du filtre adaptatif sans l''information
de la phase')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------------------%
subplot(513); plot(spm,'r');
title('Figure#3 Signal en sortie du filtre adaptatif')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------------------%
subplot(514); plot(v)  %eX11,eX21
title(['Figure#4 Signal synthétisé avec f0 = ',num2str(f0),' et le déphasage phi'])
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------------------%
subplot(515); plot(phi*180/pi)
title('Figure#5 Extraction de la phase phi')
ylabel('degré')
xlabel(' Nombre d''échantillons, avec Te=0.000125s ou Fe = 8 kHz');grid on
%-------------------------------------------------------------------------------------%
% synth(kk,:)= v; synthu(kk,:)= u;
synth2(kk,:)= v; synthu2(kk,:)= u;
end  %de
forXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX

% save fichsynth1 synth; save fichsynthu1 synthu;
% save fichsynth2 synth2; save fichsynthu2 synthu2;
```

```
%------------------------------ Fin du Programme ica.m ------------------------------%


%-----------------------------------------------------------------------%
% nom du fichier: ica1.m                    ancien:filtunitrait2.m
% version 29 juin 2004
% version 13 juin 2004                      Auteur: Antoine Munoz
% Professeurs :          M. Marcel Gabrea et M. Christian Gargour
% Pour le mémoire 2004. la prédiction linéaire%
%-----------------------------------------------------------------------%
clc,clear all,close all;
%---------fréquence d'échantillonnage du dsp codec ad535------------%
Fs = 8000; FNyq = Fs/2;    % fréquence de Nyquist
f0=128                % 140 exemple de fréquence fondamentale en Hz
global vectx;global w;global vecdelai;global vecinb;global b;global mub;
global muw;global puissx;global puissy;
%-----------------------------------------------------------------------%
% signaux synthétisés avec kf0 et phi des capteurs 1 et 2
load fichsynth1 synth;
synth1=synth; %canal 1
[pbd nbredata] = size(synth1);
load fichsynth2 synth2; % canal 2
% synth2=synth;
[pbd nbredata] = size(synth2);
%-----------------------------------------------------------------------%
for kk=1:pbd
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
   kf0 = 2^(kk-1)*f0; %la fondamentale ou harmonique concernée
%Nb=round(Fs/(2*kf0)); % Nb: nombre de coefficients du filtre predictif, ordre = N-1
selon kf0
      Nb=round(Fs/kf0); % Nb=round(Fs/(2*kf0));
% pas: valeur des 2 taux d'adaptation ou d'apprentissage
% delai: nbre d'échantillons retardés pour l'entrée du filtre, Nb=10;
Nw=2; pas = 0.001; delai = 1;
unitraitini1(Nb,Nw,pas,delai) % unitraitini1(N,pas,delai);
%-----------------------------------------------------------------------%
% xin = signal d'entrée (scalaire)
% En sortie:
% y sortie de w*x
% ye sortie de y(n-Delai)*b
% b = MAJ des coefficients du vecteur filtre predictif linéaire
% w = MAJ des coefficients du vecteur de la matrice de séparation w
% ee = erreur entre y et ye
```

```
% pb = mub pas adaptatif pour b
% pw = muw pas adaptatif pour w
%-------------------------------------------------------------------------%
x1=synth1(kk,:); x2=synth2(kk,:); %x1=eX11; x2=eX21;
for i=1:nbredata %length(x1)
   xin1=x1(i); xin2=x2(i);
   [y,ye,b,w,ee,pb,pw] = icatrait1(xin1,xin2);
   s(i)=y; yp(i)=ye; e(i)=ee;
end
resultk(kk,:)= yp; % resultk(kk,:)= s;
%-------------------------------------------------------------------------%
erreurmax=max(sqrt(e.^2));
pb, pw,                    % pas adaptatif final du traitement
b,w          % coefficients des filtres prédictif et de séparation
figure; %-----------------Figure ---------------------------------%
subplot(511); plot(x1,'b');
title('Signaux mélangés au capteur 1')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------%
subplot(512); plot(x2,'g');
title('Signaux mélangés au capteur 2')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------%
subplot(513); plot(s,'r');

title('Signal de sortie du filtre W de séparation')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------%
subplot(514); plot(yp,'g');
title('Signal estimé par le filtre b prédictif')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------------%
subplot(515); plot(e); % plot(20*log10(e.^2)); plot(e.^2);
title('l''erreur de prédiction du filtre adaptatif')
xlabel('échantillons,avec Te=0.000125s ou Fe = 8 kHz')
ylabel('Amplitude de l''erreur');grid on
end  %de
forXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
extrait = zeros(1,nbredata);
for kk=1:pbd;    extrait = extrait + resultk(kk,:);  end
load we; s1 = we; % signal du son oui source originale
figure; %-----------------Figure ---------------------------------%
subplot(211); plot(s1);
```

```
title('signal du son de la source originale')
ylabel('Amplitude');grid on
%-------------------------------------------------------------------%
subplot(212); plot(extrait);
title('signal extrait et estimé de la source')
ylabel('Amplitude');grid on
%----------------------Fin du Programme ica1.m----------------------%
```

ANNEXE 2

Programme source en langage C

La mise en œuvre en langage C, dédié pour le DSP, est ici présentée dans sa version actuelle de déverminage, de plus, l'imprimé des résultats des étapes de l'exécutable est à la suite du code source ci-dessous.

Version de déverminage développée sous Borland pour le DSP choisi.

```
/*Programme:          rehausprl.c
* =========          ========
*Conception:  Antoine Munoz tél.:(514)280-5456 ou (450)433-9517
*Date:        27 / XI / 2005
*Version:     V1.0 (développée sous win):
* TargetExpert: target type: application (.exe)
*                platform win 32
*           target model : console
*                libraries: static
*Fichier:     C:\WINDOWS\Bureau\projet2005\programmeC\realisation
*Étude:       mémoire 2005
*Projet:      rehaussement de la parole pour DSP
*Directeurs:  ETS: M. Christian Gargour et M. Marcel Gabrea
*Révision:    0
* ____________________________________________________________
*       Description:   rehaussement de la parole bruitée et convoluée
* ____________________________________________________________*/
/*======= fichiers d'entête de la librairie standards inclus =======*/
#include<stdio.h>
#include<conio.h>
#include <math.h>
#include <stdlib.h>

/*======================== fichiers inclus ========================*/
#include "intgau23.h" /*float intgau23[81] initialisé*/
#include "intgau24.h" /*float intgau24[161]initialisé*/
#include "intgau25.h" /*float intgau25[321]initialisé*/
#include "complex.h"  /*definition de la structure complex */
#include "twiddle1.h" /*header file sur constantes twiddle pour 1024*/
#include "factoriel.h"  /*float factoriel[20] initialisé*/

/*=================== définitions de constantes ==================*/
#define pi 3.141592654
#define  N 1024
#define  longdataini 356  /*256+100 trame pour le calcul initial*/
```

```
#define  longdata      256  /*trame pour le resultat traité*/
#define  longbuffer    456  /*100+256+100 trame pour le calcul*/

/*nbre d'échantillon retiré aux extrémités du traitement final*/
#define  echret        280  /* (80+160+320)/2-->560/2*/
#define  extsup        458 /*456+2, avant 354 */
#define  indexHmax   80  /*index max des coeff de l'ondelette échelle 2^3*/
#define  indexH2max  160 /*index max des coeff de l'ondelette échelle 2^4*/
#define  indexH3max  320 /*index max des coeff de l'ondelette échelle 2^5*/
#define  indexXmax   455  /*indexXmax correspondant au tabl buffrin1-2[]*/
#define  indexYmax   535  /*index max 455+80 de la 1ere convolution*/
#define  indexY2max  695  /*index max 535+160 de la 2ieme convolution*/
#define  indexY3max  1015 /*index max 695+320 de la 3ieme convolution*/

/*========== Déclaration des fonctions du programme ==================*/

/*======== fonction: void FFT(COMPLEX *Y, int dir) ================*/
/*=== vecteur d'entrée, flag de la FFT dir ou inv ================*/
 void FFT(COMPLEX *Y, int dir);
 /*======================== fonction  seuilmin()
========================*/
 /*======== trouve le seuil minimal pour les valeurs maximales  ========*/
 float seuilmin(int indexY,float y[]);
/*================== fonction deff0(mindesmax,y2,y3)
==================*/
/*cherche les maximas de la transformee en ondelettes*/
/*lde y3:1016, 1016-456=560,560/2=280 , indexXmax=455*/
 int deff0(float mindesmax,float y2[],float y3[]);
/*====================== fonction algcmplxLMS
=========================*/
 void algcmplxLMS(COMPLEX *pxlms, COMPLEX *phlms, float d, COMPLEX
*rslt);
/*======================== fonction fseprd
============================*/
void fseprd(float *vinx, float *VW, float *vinb, float *b, float *vdelai, float
*rsltSEPRD, int Nb);
/*============================================================
========*/
/*##########################################################################
#*/
/*======================== Programme principal
=========================*/
void main(void)
        {
```

```
  /*============== Lecture et transformée en ondelette ================*/
  /*=========== détermination du pitch Fo de chaque trame ============*/
  int r, nbredata=0, segmnt=longdataini;
  int dernposmax1=0, dernposmax2=0;
  int nbredemax1, nbredemax2;
        int quit,n,i,f,rmax,dir;
  char lignetextlue[30];

  float signalin1[longdataini]={0}, signalin2[longdataini]={0};
  float buffrin1[longbuffer]={0}, buffrin2[longbuffer]={0};
  float cumulval1, moyval1, cumulval2, moyval2, varval1, varval2;
  float cumulvalcar1, cumulvalcar2;
  float valeur, mindesmax, arg1=2*pi, fs=8000;
  float tbvectrv[N]={0}; /*tbvectrv: tableau vecteur de travail*/
  float tbf01[40]={0};  /*tbf01: tableau vecteur de f0 du canal1*/
  float tbf02[40]={0};  /*tbf02: tableau vecteur de f0 du canal2*/
  double xr;
  COMPLEX X[N],H[N];  /*COMPLEX X[512],H[512];*/


/*================================================================
=*/
  /*Section sur le signal analytique et le filtre adaptatif LMS de Wiener*/
  float valr1, vali1, enveloppe[longdata]= {0};
  COMPLEX synthp[longdata],synthm[longdata],vecthlms[longdata],yLMS[longdata];
  COMPLEX xlms[2],hlms[2],resuLMS[5];
  /* résultat du LMS :  resuLMS[5]={y,h,ede,p,mu};
  sortie y ,coefficient h, erreur, puissance, mu (le pas)*/
  /*=============== vecteurs pour déverminage
=========================*/
  /*
  COMPLEX lepas[longdata], pwr[longdata], lerreur[longdata];
  */

/*================================================================
=*/
  /*Section sur la partie du prédicteur rehausseur */
  /*Nb: nombre de coefficients du filtre predictif ordre = N-1 selon kf0*/
  int Nb, sbdes, nbsbandes[2];
  float tbf0[2];
  float b[longdata],vinb[longdata], vdelai[5], VW[2],vinx[2];
  float rsltSEPRD[10];

  /*
```

```
résultat DE LA SÉPARATION ET DU PRÉDICTEUR pour la fonction:
fseprd( vinx, VW, inb, b, vdelai, rsltSEPRD)
rsltSEPRD[10]={y,ye,b,w,ee,mub,muVW, pwrx, pwry,lambda};
0) y sortie de la matrice de séparation: MW*xin
      1) ye sortie du prédicteur: y(n-Delai)*b
      2) ee = erreur entre y et ye
      3) lambda
      4) pwrx puissance instantanée de x
      5) pwry puissance instantanée de y
      6) mub pas adaptatif pour b
7) muVW pas adaptatif pour w
8) b = MAJ des coefficients du vecteur filtre predictif linéaire
9) w = MAJ des coefficients du vecteur de la matrice de séparation MW
*/

/*==========================en
lecture============================*/
FILE *ptrfichr1;   /* fichier d'entrée canal 1 en lecture*/
FILE *ptrfichr2;   /* fichier d'entrée canal 2 en lecture*/
/*=========================en écriture============================*/
/* FILE *ptrfichw1;   fichier en sortie du canal 1 en écriture*/
/* FILE *ptrfichw2;    fichier en sortie du canal 2 en écriture*/

FILE *ptrfichw3;   /* fichier du suivi résultats généraux en écriture*/
FILE *ptrfichw4;   /* fichier résultats auxiliaires en écriture*/
/*=============== Partie couvrant le banc de filtres ===============*/
int ordN, nbrefilt;

      float fnyq=4000;
/* variables de calcul*/
      float deltaomega, onduldB, Beta, I0Beta, vt,I0valeur, Vt, x;
      float alpha, beta, omegac0, omegac1, omegac2, f0, kf0;

/* Kaiserwin: vecteur des coefficients de la fenêtre de Kaiser
* bisdspb: vecteur des bornes inférieure et supérieure des filtres
* passe-bande inférieur à la fréquence de Nyquist dans le banc.
*/
float Kaiserwin[680]={0};
float bisdspb[10]={0};

/*coefficients des passe-bandes du banc de filtre pour canal 1 et 2*/
/*grandeur fixée pour une f0 min de 60 Hz*/

/* coeffiltx: vecteur des coefficients de chaque filtre passe-bande
```

```
 * du banc de filtre plus le passe-bas en dernier*/
float coeffilt0[680]={0};
float coeffilt1[680]={0};
float coeffilt2[680]={0};
float coeffilt3[680]={0};
float coeffilt4[680]={0};
float coeffilt5[680]={0};
float coeffilt6[680]={0};
/* sigfilt1x: vecteur des signaux filtrés sur chaque filtre passe-bande
 * du banc de filtre et du passe-bas du canal 1*/
float sigfilt10[N]={0};
float sigfilt11[N]={0};
float sigfilt12[N]={0};
float sigfilt13[N]={0};
float sigfilt14[N]={0};
float sigfilt15[N]={0};
float sigfilt16[N]={0};
/* sigfilt1x: vecteur des signaux filtrés sur chaque filtre passe-bande
 * du banc de filtre et du passe-bas du canal 2*/
float sigfilt20[N]={0};
float sigfilt21[N]={0};
float sigfilt22[N]={0};
float sigfilt23[N]={0};
float sigfilt24[N]={0};
float sigfilt25[N]={0};
float sigfilt26[N]={0};
/* offset1x: vecteur des offset des signaux filtrés sur chaque filtre
passe-bande et du passe-bas du canal 1 sur la convolution */
float offset10[N]={0};
float offset11[N]={0};
float offset12[N]={0};
float offset13[N]={0};
float offset14[N]={0};
float offset15[N]={0};
float offset16[N]={0};
/* offset2x: vecteur des offset des signaux filtrés sur chaque filtre
passe-bande et du passe-bas du canal 2 pour la convolution */
float offset20[N]={0};
float offset21[N]={0};
float offset22[N]={0};
float offset23[N]={0};
float offset24[N]={0};
float offset25[N]={0};
float offset26[N]={0};
```

```
/* ssynt1x: vecteur des  signaux synthétisés sur chaque filtre
 passe-bande du banc de filtre du canal 2 */
float ssynt10[N]={0};
float ssynt11[N]={0};
float ssynt12[N]={0};
float ssynt13[N]={0};
float ssynt14[N]={0};
float ssynt15[N]={0};
/* ssynt2x: vecteur des  signaux synthétisés sur chaque filtre
 passe-bande du banc de filtre du canal 2 */
float ssynt20[N]={0};
float ssynt21[N]={0};
float ssynt22[N]={0};
float ssynt23[N]={0};
float ssynt24[N]={0};
float ssynt25[N]={0};
/* tampon1x: vecteur des  signaux tampon mémorisés sur chaque filtre
 passe-bande du banc de filtre du canal 1 pour le calcul
 de la transformée de Hilbert*/
float tampon10[50]={0};
float tampon11[50]={0};
float tampon12[50]={0};
float tampon13[50]={0};
float tampon14[50]={0};
float tampon15[50]={0};
/* tampon2x: vecteur des  signaux tampon mémorisés sur chaque filtre
 passe-bande du banc de filtre du canal 2 pour le calcul
 de la transformée de Hilbert*/
float tampon20[50]={0};
float tampon21[50]={0};
float tampon22[50]={0};
float tampon23[50]={0};
float tampon24[50]={0};
float tampon25[50]={0};
/* resulms1x: vecteur des  signaux mémorisés sur chaque filtre
 passe-bande du banc de filtre du canal 1 pour le calcul
 du LMS : context du résultat du LMS :  resuLMS[5]={y,h,ede,p,mu};
sortie y ,coefficient h, erreur, puissance, mu (le pas)*/
COMPLEX resulms10[5];
COMPLEX resulms11[5];
COMPLEX resulms12[5];
COMPLEX resulms13[5];
COMPLEX resulms14[5];
COMPLEX resulms15[5];
```

```
/* resulms2x: vecteur des  paramètres émorisés sur chaque filtre
 passe-bande du banc de filtre du canal 2 pour le calcul
 du LMS : context du résultat du LMS :  resuLMS[5]={y,h,ede,p,mu};
sortie y ,coefficient h, erreur, puissance, mu (le pas)*/
COMPLEX resulms20[5];
COMPLEX resulms21[5];
COMPLEX resulms22[5];
COMPLEX resulms23[5];
COMPLEX resulms24[5];
COMPLEX resulms25[5];
/* hlms1x: vecteur des  signaux complex mémorisés sur chaque filtre
 passe-bande du banc de filtre du canal 1 pour le calcul
 des coefficient du vecteur h LMS */
COMPLEX hlms10[2];
COMPLEX hlms11[2];
COMPLEX hlms12[2];
COMPLEX hlms13[2];
COMPLEX hlms14[2];
COMPLEX hlms15[2];
/* hlms2x: vecteur des  signaux complex mémorisés sur chaque filtre
 passe-bande du banc de filtre du canal 2 pour le calcul
 des coefficient du vecteur h LMS */
COMPLEX hlms20[2];
COMPLEX hlms21[2];
COMPLEX hlms22[2];
COMPLEX hlms23[2];
COMPLEX hlms24[2];
COMPLEX hlms25[2];
      /*affectation des pointeurs filtre dans le tableau des pointeurs*/
/*adresse du début du tableau de coeffilt0[] dans tblptr[0]
 ptrcof prend les valeurs successives tblptr[0] a tblptr[5]*/

float *ptrcof, *psigfilt, *poffset1, *poffset2, *pssynt1, *pssynt2;
float *ptampon1, *ptampon2 ;
float *tblptr[7],*tblptr11[7], *tblptr12[7], *tblptr21[7], *tblptr22[7];
float *tblptr31[6], *tblptr32[6], *tblptr41[6], *tblptr42[6];
COMPLEX *ptresulms1, *tblptr51[6];
COMPLEX *ptresulms2, *tblptr52[6];
COMPLEX *ptrhlms1, *tblptr61[6];
COMPLEX *ptrhlms2, *tblptr62[6];
/*valeur du pointeur ptrcof dans tableau tblptr*/
tblptr[0] = coeffilt0;
tblptr[1] = coeffilt1;
tblptr[2] = coeffilt2;
```

```
tblptr[3] = coeffilt3;
tblptr[4] = coeffilt4;
tblptr[5] = coeffilt5;
tblptr[6] = coeffilt6;
/*valeur du pointeur psigfilt dans tableau tblptr11 pour signaux filtrés
     sur canal 1*/
tblptr11[0] = sigfilt10;
tblptr11[1] = sigfilt11;
tblptr11[2] = sigfilt12;
tblptr11[3] = sigfilt13;
tblptr11[4] = sigfilt14;
tblptr11[5] = sigfilt15;
tblptr11[6] = sigfilt16;
/*valeur du pointeur psigfilt dans tableau tblptr12 pour signaux filtrés
     sur canal 2*/
tblptr12[0] = sigfilt20;
tblptr12[1] = sigfilt21;
tblptr12[2] = sigfilt22;
tblptr12[3] = sigfilt23;
tblptr12[4] = sigfilt24;
tblptr12[5] = sigfilt25;
tblptr12[6] = sigfilt26;
/*valeur du pointeur poffset1 dans tableau tblptr21 pour la convolution
     overlap and add sur canal 1*/
tblptr21[0] = offset10;
tblptr21[1] = offset11;
tblptr21[2] = offset12;
tblptr21[3] = offset13;
tblptr21[4] = offset14;
tblptr21[5] = offset15;
tblptr21[6] = offset16;
/*valeur du pointeur poffset2 dans tableau tblptr22 pour la convolution
     overlap and add sur canal 2*/
tblptr22[0] = offset20;
tblptr22[1] = offset21;
tblptr22[2] = offset22;
tblptr22[3] = offset23;
tblptr22[4] = offset24;
tblptr22[5] = offset25;
tblptr22[6] = offset26;
/*valeur du pointeur pssynt1 dans tableau tblptr31 pour le signal
 synthétisé sur canal 1*/
tblptr31[0] = ssynt10;
tblptr31[1] = ssynt11;
```

```
tblptr31[2] = ssynt12;
tblptr31[3] = ssynt13;
tblptr31[4] = ssynt14;
tblptr31[5] = ssynt15;
/*valeur du pointeur pssynt2 dans tableau tblptr32 pour le signal
 synthétisé sur canal 2*/
tblptr32[0] = ssynt20;
tblptr32[1] = ssynt21;
tblptr32[2] = ssynt22;
tblptr32[3] = ssynt23;
tblptr32[4] = ssynt24;
tblptr32[5] = ssynt25;
/*valeur du pointeur ptampon1 dans tableau tblptr41 pour le signal
 mis en tampon sur canal 1*/
tblptr41[0] = tampon10;
tblptr41[1] = tampon11;
tblptr41[2] = tampon12;
tblptr41[3] = tampon13;
tblptr41[4] = tampon14;
tblptr41[5] = tampon15;
/*valeur du pointeur ptampon2 dans tableau tblptr42 pour le signal
 mis en tampon sur canal 2*/
tblptr42[0] = tampon20;
tblptr42[1] = tampon21;
tblptr42[2] = tampon22;
tblptr42[3] = tampon23;
tblptr42[4] = tampon24;
tblptr42[5] = tampon25;
/*valeur du pointeur ptresulms1 dans tableau tblptr51 pour les signaux
 mis dans resuLMS[5]={y,h,ede,p,mu}sur canal 1*/
tblptr51[0] = resulms10;
tblptr51[1] = resulms11;
tblptr51[2] = resulms12;
tblptr51[3] = resulms13;
tblptr51[4] = resulms14;
tblptr51[5] = resulms15;
/*valeur du pointeur ptrhlms1 dans tableau tblptr61 pour les signaux
 mis dans hlms[2] sur canal 1*/
tblptr61[0] = hlms10;
tblptr61[1] = hlms11;
tblptr61[2] = hlms12;
tblptr61[3] = hlms13;
tblptr61[4] = hlms14;
tblptr61[5] = hlms15;
```

```
   /*valeur du pointeur ptresulms2 dans tableau tblptr52 pour les signaux
    mis dans resuLMS[5]={y,h,ede,p,mu}sur canal 2*/
   tblptr52[0] = resulms20;
   tblptr52[1] = resulms21;
   tblptr52[2] = resulms22;
   tblptr52[3] = resulms23;
   tblptr52[4] = resulms24;
   tblptr52[5] = resulms25;
   /*valeur du pointeur ptrhlms2 dans tableau tblptr62 pour les signaux
    mis dans hlms[2] sur canal 2*/
   tblptr62[0] = hlms20;
   tblptr62[1] = hlms21;
   tblptr62[2] = hlms22;
   tblptr62[3] = hlms23;
   tblptr62[4] = hlms24;
   tblptr62[5] = hlms25;

/*==========================================================
=*/
   clrscr();

/*==========================================================
=*/
   /* Ouverture des fichiers sigcan1.dat et sigcan2.dat contenant
      les signaux captés par les microphones 1 & 2 */
   /*lecture du fichier sigcan1.dat et sigcan2.dat-------*/
   printf(" commence lecture des fichiers sigcan1.dat et sigcan2.dat:\n");

   if ((ptrfichr1 = fopen("sigcan1.dat", "r"))==NULL)
   { fprintf(stderr,"le fichier 1 ne peut être ouvert\n");
     exit(1);}

   if ((ptrfichr2 = fopen("sigcan2.dat", "r"))==NULL)
   { fprintf(stderr,"le fichier 2 ne peut être ouvert\n");
     exit(1);}

   /*==================== fichiers en écriture ================*/
   /*ptrfichw1 = fopen("fichTO1.dat", "a+");*/
   /*ptrfichw2 = fopen("fichTO2.dat", "a+");*/
   /* ptrfichw1 = fopen("envelop10.dat", "a+");*/

   ptrfichw3 = fopen("fchrsult.dat", "a+");
   ptrfichw4 = fopen("sgrehaus.dat", "a+");
```

```
/*==========================================================
=*/

   n=0;
   while (!feof(ptrfichr1)) {fgets(lignetextlue,30,ptrfichr1); n++; }
    printf("le nbre de donnees est: %d\n", n);
    rewind(ptrfichr1);
    rmax = (int)ceil((double)n/(double)longdata);
    printf("rmax vaut: %d\n", rmax);



/*==========================================================
=*/
   /* pour déverminage et suivi de développement */
   /*ptrfichw4 = fopen("sigf10.dat", "a+");*/
   /*ptrfichw4 = fopen("sigf20.dat", "a+");*/

   fprintf(ptrfichw3, "Commence lecture fichiers sigcan1.dat et sigcan2.dat:\n");
   fprintf(ptrfichw3, "avec le programme rehausprl.c \n\n\n");
   fprintf(ptrfichw3, "le nbre de donnees est %d\n", n);
   fprintf(ptrfichw3, "le nbre des itérations est de %d\n", rmax);

/*==========================================================
=*/

   /*%%%%%%%%%% itération r pour chaque trame en lecture %%%%%%%%%%*/
   for (r=0;r<rmax;r++) /* 16 pour essai r=1: une trame, r=2: deux trames*/
   { /*%%%%%%%%%%%%%  début de for (r=0;r<8;r++)
%%%%%%%%%%%%%%%%%%%%*/

   /*====================== lecture du canal 1 ====================*/
   n=0; cumulval1=0.0; moyval1=0.0; varval1=0.0; cumulvalcar1= 0.0;
   while ((!feof(ptrfichr1)) && (n<segmnt))
   {
   fgets(lignetextlue,30,ptrfichr1); valeur = atof(lignetextlue);
   signalin1[n]=valeur; n++;
   cumulval1 += valeur; cumulvalcar1 += (valeur*valeur); moyval1= cumulval1/n;
   }

   if (n>1)  varval1= ( cumulvalcar1 - ((cumulval1*cumulval1)/n) )/(n-1) ;
    printf("le nbre n sur canal 1 est: %d\n", n);
```

```
   fprintf(ptrfichw3,
"================================================\n");
   fprintf(ptrfichw3, "le nbre n sur canal 1 est: %d\n", n);
   fprintf(ptrfichw3, "la moyenne sur canal 1 est: %5.3f\n", moyval1);
   fprintf(ptrfichw3, "la variance sur canal 1 est: %5.5f\n", varval1);
  if (n<segmnt) { for (i=n;i<segmnt;i++) signalin1[i]=0.0; }

  /*====================== lecture du canal 2 ====================*/
    n=0; cumulval2=0.0; moyval2=0.0; varval2=0.0; cumulvalcar2= 0.0;
  while ((!feof(ptrfichr2)) && (n<segmnt))
  {
  fgets(lignetextlue,30,ptrfichr2); valeur = atof(lignetextlue);
  signalin2[n]=valeur; n++;
  cumulval2 += valeur; cumulvalcar2 += (valeur*valeur); moyval2= cumulval2/n;
  }

  if (n>1)  varval2= ( cumulvalcar2 - ((cumulval2*cumulval2)/n) )/(n-1) ;
  printf("le nbre n sur canal 2 est: %d\n", n);
  fprintf(ptrfichw3, "le nbre n sur canal 2 est: %d\n", n);
  fprintf(ptrfichw3, "la moyenne sur canal 2 est: %5.3f\n", moyval2);
  fprintf(ptrfichw3, "la variance sur canal 2 est: %5.5f\n", varval2);
 if (n<segmnt) { for (i=n;i<segmnt;i++) signalin2[i]=0.0; }


/*============================================================*/
  /*Les données sont dans les vecteurs signalin1[] & signalin2[] */

/*============================================================
=*/
   nbredata+=n;
   printf("le nbre de data est: %d\n", nbredata);
   fprintf(ptrfichw3, "le nbre de data est: %d\n", nbredata);
   printf("le nbre de segment lu est: %d\n", r+1);
   fprintf(ptrfichw3, "le nbre de segment lu est: %d\n\n\n", r+1);
   fprintf(ptrfichw3,
"================================================\n");

/*============================================================
=*/

    /*longdataini 356   longdata       256   longbuffer       456  */
    /* transfert des signaux signalin1/2 dans buffrin1/2 */
   if (segmnt==longdataini)
```

```
  {

       for(i=0;i<segmnt;i++)
       {buffrin1[i+100]=signalin1[i]; buffrin2[i+100]=signalin2[i];}
  }
  else
  {
       for(i=0;i<segmnt;i++)
   {buffrin1[i+200]=signalin1[i]; buffrin2[i+200]=signalin2[i];}
  }

/*==========================================================
=*/

/*==========================================================
=*/
  /*============ 1) traitement des signaux sur buffrin1 ==========*/

  /* RAZ des vecteurs de traitement X,H pour la FFT */
  for (n=0;n<N;n++)
       {X[n].real =0; X[n].imag =0;  H[n].real =0; H[n].imag =0;}

/*==========================================================
=*/
  /*transfert des données réelles du signal vers vecteurs complexes
  * pour la FFT indexHmax = 80,longbuffer = 456,indexYmax = 535;*/
  for (n=0;n<longbuffer;n++)        {X[n].real = buffrin1[n];}
  for (n=0;n<=indexHmax;n++)        {H[n].real = intgau23[n];}

       /*------------------- la FFT de X et H  --------------------*/
  printf(" commence la transformee en ondelettes sur canal 1:\n");
  fprintf(ptrfichw3,"commence la transformee en ondelettes sur canal 1:\n");

  /* commence les FFT sur les vecteurs X et H*/
  dir=1; FFT(X,dir); FFT(H,dir);
  for (n=0;n<N;n++)
       {
    xr= X[n].real; /* mémorise pour la 2 ieme equation*/
    X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
    X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
   }
  /*fait la FFT inverse de X(f)-->X(n):*/
  dir=-1; FFT(X,dir);
```

```
   for (n=0;n<N;n++) {H[n].real =0; H[n].imag =0;} /*FFT(H,dir,N);*/
   for(i = 0; i <= indexYmax-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la première convolution nx=536*/

/*=============================================================
=*/
   /* indexH2max = 160, indexX2max = 535, indexY2max = 695; */
   for (n=0;n<indexH2max+1;n++)  { H[n].real = intgau24[n];}
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
        {
     xr= X[n].real; /* mémorise pour la 2 ieme equation*/
     X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
     X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
     }
   /*fait la FFT inverse de X(f)-->X(n):*/
   dir=-1; FFT(X,dir);
   for (n=0;n<N;n++) {H[n].real =0; H[n].imag =0;} /*FFT(H,dir,N);*/
   for(i = 0; i <= indexY2max-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la seconde convolution nx=696*/

/*=============================================================
=*/
   /* indexH3max = 320, indexX3max = 695, indexY3max = 1015; */
   for (n=0;n<indexH3max+1;n++)  { H[n].real = intgau25[n];}
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
        {
     xr= X[n].real; /* mémorise pour la 2 ieme equation*/
     X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
     X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
     }
   /*fait la FFT inverse de X(f)-->X(n):*/
   dir=-1; FFT(X,dir);
   for(i = 0; i <= indexY3max-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la troisième convolution nx=1016*/

/*=============================================================
=*/
   /*------ envoi le résultat de la convolution par FFT  ---------*/
   /*ouverture du fichier du résultat et mise à jour*/
   /*
   printf(" commence écriture du fichier fichTO1.dat\n");
   fprintf(ptrfichw3," commence écriture du fichier fichTO1.dat\n");
```

```
  */
  /*
  ptrfichw1 = fopen("fichTO1.dat", "a+");
        for(f = 0+380; f <= indexY3max-380; f++)
          {fprintf(ptrfichw1, "%f\n", X[f].real);}
  */

/*================================================================
=*/
  /*écriture dans tbvectrv: résultat de la trf-en-ondelette*/
  for(f = 0; f <= indexY3max; f++){tbvectrv[f]= X[f].real ;}

  printf("la transformee en ondelette du canal 1 est faite.\n");
  fprintf(ptrfichw3,"la transformee en ondelette du canal 1 est faite.\n");

/*================================================================
=*/

  /*---- trouve le seuil minimal pour les valeurs maximales ------*/
  mindesmax=seuilmin(indexY3max,tbvectrv);

  printf(" le seuil de la trfond du canal 1 est: %5.4f\n",mindesmax);
  fprintf(ptrfichw3,"le seuil de la trfond du canal 1 est: %5.4f\n",mindesmax);


/*================================================================
=*/

  /* cherche les maxima de la transformee en ondelettes */
  nbredemax1=deff0(mindesmax,tbf01,tbvectrv);
  printf(" le nombre de max dans le signal 1 est: %d\n",nbredemax1);
  printf(" les frequences fondamentales dans la fenetre du signal 1 sont:\n");

  fprintf(ptrfichw3,"le nombre de max dans le signal 1 est: %d\n",nbredemax1);
  fprintf(ptrfichw3,"les frequences fondamentales dans la fenetre du signal 1 sont:\n");

  for(i = 0 ; i < nbredemax1-1 ; i++)
  {
  printf(" f0[%d]= %4.1f Hz a echantillon
%d\n",i,tbf01[i+nbredemax1],(int)tbf01[i+1]);
  fprintf(ptrfichw3," f0[%d]= %4.1f Hz a echantillon
%d\n",i,tbf01[i+nbredemax1],(int)tbf01[i+1]);
```

```
   if  (tbf01[i+1]<longdata) dernposmax1=(int)tbf01[i+1];
   }
   printf(" dernposmax1= %d\n",dernposmax1);
   fprintf(ptrfichw3," dernposmax1= %d\n\n\n",dernposmax1);


/*=============================================================
========*/

/*=============================================================
========*/



/*=============================================================
=*/
   /*============ 2) traitement des signaux sur buffrin2 ==========*/

   /* RAZ des vecteurs de traitement X,H pour la FFT */
   for (n=0;n<N;n++)
        { X[n].real =0; X[n].imag =0;  H[n].real =0; H[n].imag =0; }

/*=============================================================
=*/
   /*transfert des données réelles du signal vers vecteurs complexes
   * pour la FFT indexHmax = 80,longbuffer = 456,indexYmax = 535;*/
   for (n=0;n<longbuffer;n++)        {X[n].real = buffrin2[n];}
   for (n=0;n<=indexHmax;n++)        {H[n].real = intgau23[n];}

         /*------------------- la FFT de X et H  ---------------------*/
   printf(" commence la transformee en ondelettes sur canal 2:\n");
   fprintf(ptrfichw3,"commence la transformee en ondelettes sur canal 2:\n");

   /* commence les FFT sur les vecteurs X et H*/
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
        {
     xr= X[n].real; /* mémorise pour la 2 ieme equation*/
     X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
     X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
     }
   /*fait la FFT inverse de X(f)-->X(n):*/
   dir=-1; FFT(X,dir);
   for (n=0;n<N;n++) {H[n].real =0; H[n].imag =0;} /*FFT(H,dir,N);*/
```

```
   for(i = 0; i <= indexYmax-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la première convolution nx=536*/

/*============================================================
=*/
   /* indexH2max = 160, indexX2max = 535, indexY2max = 695; */
   for (n=0;n<indexH2max+1;n++)  { H[n].real = intgau24[n];}
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
         {
     xr= X[n].real; /* mémorise pour la 2 ieme equation*/
     X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
     X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
     }
   /*fait la FFT inverse de X(f)-->X(n):*/
   dir=-1; FFT(X,dir);
   for (n=0;n<N;n++) {H[n].real =0; H[n].imag =0;} /*FFT(H,dir,N);*/
   for(i = 0; i <= indexY2max-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la seconde convolution nx=696*/

/*============================================================
=*/
   /* indexH3max = 320, indexX3max = 695, indexY3max = 1015; */
   for (n=0;n<indexH3max+1;n++)  { H[n].real = intgau25[n];}
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
         {
     xr= X[n].real; /* mémorise pour la 2 ieme equation*/
     X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
     X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);
     }
   /*fait la FFT inverse de X(f)-->X(n):*/
   dir=-1; FFT(X,dir);
   for(i = 0; i <= indexY3max-1 ; i++) {X[i].real=(X[i+1].real)- (X[i].real);}
   /*X contient le résultat de la troisième convolution nx=1016*/

/*============================================================
=*/
   /*------ envoi le résultat de la convolution par FFT  ---------*/
   /*ouverture du fichier du résultat et mise à jour*/
   /*
   printf(" commence écriture du fichier fichTO2.dat\n");
   fprintf(ptrfichw3," commence écriture du fichier fichTO2.dat\n");
   */
```

```
  /*
  ptrfichw2 = fopen("fichTO2.dat", "a+");
        for(f = 0+380; f <= indexY3max-380; f++)
          {fprintf(ptrfichw2, "%f\n", X[f].real);}
  */

/*============================================================
=*/
  /*écriture dans tbvectrv: résultat de la trf-en-ondelette*/
  for(f = 0; f <= indexY3max; f++){tbvectrv[f]= X[f].real ;}

  printf(" la transformee en ondelette est faite.\n");
  fprintf(ptrfichw3,"la transformee en ondelette sur le canal 2 est faite.\n");

/*============================================================
=*/

  /*---- trouve le seuil minimal pour les valeurs maximales ------*/
  mindesmax=seuilmin(indexY3max,tbvectrv);

  printf(" le seuil de la trfond du canal 2 est: %5.4f\n",mindesmax);
  fprintf(ptrfichw3,"le seuil de la trfond du canal 2 est: %5.4f\n",mindesmax);

/*============================================================
=*/

  /* cherche les maxima de la transformee en ondelettes */
  nbredemax2=deff0(mindesmax,tbf02,tbvectrv);
  printf(" le nombre de max dans le signal 2 est: %d\n",nbredemax2);
  printf(" les frequences fondamentales dans la fenetre du signal sont:\n");

  fprintf(ptrfichw3,"le nombre de max dans le signal 2 est: %d\n",nbredemax2);
  fprintf(ptrfichw3,"les frequences fondamentales dans la fenetre du signal 2 sont:\n");

  for(i = 0 ; i < nbredemax2-1 ; i++)
  {
  printf(" f0[%d]= %4.1f Hz a echantillon
%d\n",i,tbf02[i+nbredemax2],(int)tbf02[i+1]);
  fprintf(ptrfichw3," f0[%d]= %4.1f Hz a echantillon
%d\n",i,tbf02[i+nbredemax2],(int)tbf02[i+1]);
  if  (tbf02[i+1]<longdata) dernposmax2=(int)tbf02[i+1];
  }

  printf(" dernposmax2= %d\n",dernposmax2);
```

```
  fprintf(ptrfichw3," dernposmax2= %d\n\n\n",dernposmax2);

/*============================================================
==========*/

/*============================================================
==========*/

  /*====== Le traitement global doit s'insérer dans la suite ici =====*/

/*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX*/

/*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX*/

  /* Sélection de la fréquence fondamentale f0 la plus représentative
   * dans la trame courante sur le canal 1*/
  /*
  premiere approche choisie: on garde la plus grande f0
  seconde approche choisie: on garde la plus petite f0
  */

  /*
  f0=0;
  for(i=nbredemax1; i<(2*nbredemax1-1);i++){if (tbf01[i]>f0)  f0 = tbf01[i];}
  */

  /*seconde approche choisie: on garde la plus petite f0*/
  f0=1000;
  for(i=nbredemax1; i<(2*nbredemax1-1);i++){if (tbf01[i]<f0)  f0 = tbf01[i];}

  printf("la f0 choisie dans la trame du canal 1 vaut %4.1f Hz\n",f0);
  fprintf(ptrfichw3,"la f0 choisie dans la trame du canal 1 vaut %4.1f Hz\n",f0);


/*============================================================
=*/
  tbf0[0]= f0; /*f0 choisie dans la trame courante du canal 1*/
  alpha = f0/sqrt(2);          /* borne inférieure en Hertz*/
  beta = 2*alpha;              /* borne supérieure en Hertz*/

  /*============== largeur de la bande de transition ============*/
  /*1.5;on augmente de 50% la bande de transition*/
```

```
deltaomega = pi*1.5*(f0-alpha)/fnyq ;
/*ondulation en dB pour un coefficient d'ondulation de 0.01*/
onduldB = -20*log10(0.01); /*en valeur absolue*/

/*============= détermination de l'ordre du filtre =============*/
/* si l'ordre N est un nombre pair, alors le nombre de termes M
de la fenetre sera impair (M=N+1)et nous aurons une réponse
 impulsionnelle      symétrique garantissant un déphasage linéaire*/
ordN = ceil((onduldB-8)/(2.285*deltaomega));
if ((ordN % 2)!= 0) ordN++;
/*====== détermination du Beta pour la fenetre de Kaiser =======*/
if (onduldB < 21) Beta=0;
if ((onduldB >= 21)&& (onduldB <= 50))
      Beta = 0.5842*pow((onduldB-21),0.4)+ 0.07886*(onduldB-21);
if (onduldB > 50) Beta = 0.1102*(onduldB - 8.7);

/*============= Calcul de la fenetre de Kaiser ================*/
/*Calcul de la fonction modifiée de Bessel d'ordre zéro du premier
  genre; Io(Beta)*/
/*calcul itératif sur 21 termes*/
I0Beta=1;
for(i=1;i<20;i++) { vt=pow((Beta/2),i)/factoriel[i];I0Beta += (vt*vt);}

/*============== L'équation de la fenetre de Kaiser ==============*/
/*Kaiserwin[n]=Io[ Beta*sqrt(1-((n-ordN/2)/ordN/2)^2 ] / I0Beta*/
for(n=0;n <= ordN;n++)
{ x=n; /*valeur d'un int vers un float pour le calcul*/
  Vt = (x-(ordN/2))/(ordN/2);
  valeur = Beta*sqrt(1-(Vt*Vt));

   I0valeur=1;
      for(i=1;i<20;i++)  /*calcul itératif sur 21 termes*/
                        {
                vt = (pow((valeur/2),i))/(factoriel[i]) ;
                I0valeur += (vt*vt);
                        }
  Kaiserwin[n]=  I0valeur/I0Beta ; }


/*============================================================
=*/
   /*calcule le nombre de passe bandes contenu dans le banc de filtre*/

   nbrefilt=1; /*par défaut*/
```

```
  bisdspb[0]= beta/2; bisdspb[1]= beta;
  while (2*beta < fnyq ) { beta*=2; nbrefilt++; bisdspb[nbrefilt]= beta;}
  nbsbandes[0]= nbrefilt;

  printf(" il y a %d passe-bandes dans le banc de filtre\n",nbrefilt);
  printf(" dont l'ordre N est: %d \n",ordN);
  fprintf(ptrfichw3," il y a %d passe-bandes dans le banc de filtre\n",nbrefilt);
  fprintf(ptrfichw3,"  dont l'ordre N est: %d \n",ordN);

  for(f=0; f <= nbrefilt; f++)
  {printf(" bisdspb[%d] = %5.4f\n",f,bisdspb[f]);
   fprintf(ptrfichw3,"  bisdspb[%d] = %5.4f\n",f,bisdspb[f]);
  }


/*==========================================================
=*/
  /*nbrefilt=4; /*pour essai du programme!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
  for(f=0; f < nbrefilt; f++)
  { /*f111111111111or(f=0; f < nbrefilt; f++)11111111111111111111*/
  /*pointeur courant ptrcof égale au début d'adresse du tableau correspondant
   au filtre passe-bande dans le banc de filtre*/
  ptrcof = tblptr[f];
  omegac2=(pi/fnyq)* bisdspb[f+1]; /*omegac2=(pi/fnyq)* bisdspb[1]*/
  omegac1=(pi/fnyq)* bisdspb[f]; /*omegac1=(pi/fnyq)* bisdspb[0]*/

/*==========================================================
=*/
  for(n=0;n <= ordN;n++)
  { if (n == ordN/2)
    *(ptrcof+n)=(omegac2-omegac1)/pi;
    else
    *(ptrcof+n)=(sin(omegac2*(n-(ordN/2)))-sin(omegac1*(n-(ordN/2))))/(pi*(n-
(ordN/2)));
  }

/*==========================================================
=*/
  /*le produit de la fenetre de Kaiser sur le vecteur des coefficients
    le résultat est mis dans le vecteur des coefficients */
  for(n=0; n <= ordN;n++) {*(ptrcof+n)=  *(ptrcof+n) * Kaiserwin[n];}
  } /*f111111111111or(f=0; f < nbrefilt; f++)11111111111111111111*/
```

```
/*============================================================
=*/
   /*========== Pour le filtre passe-bas sur canal 1 ==============*/
   omegac0=(pi/fnyq)* bisdspb[0]; ptrcof = tblptr[nbrefilt];
   for(n=0;n <= ordN;n++)
   { if (n == ordN/2)
     *(ptrcof+n) = omegac0/pi;
     else
     *(ptrcof+n)=sin(omegac0*(n-(ordN/2))) / ( pi*(n-(ordN/2)));

     *(ptrcof+n)=  *(ptrcof+n) * Kaiserwin[n];
   }


/*============================================================
=*/
   printf("ecriture des coefficients dans coeffiltX[] pour canal 1\n");
   fprintf(ptrfichw3,"ecriture des coefficients dans coeffiltX[] pour canal 1\n");
   /*
   ptrfichw4 = fopen("coeff0.dat", "a+");
   for(f = 0; f <= ordN; f++){ fprintf(ptrfichw4,"%f\n",coeffilt0[f]);}
   */
   printf(" le banc de filtre pour le canal 1 est fait.\n");
   fprintf(ptrfichw3," le banc de filtre pour le canal 1 est fait.\n");

/*============================================================
=*/

   /* partie du filtrage du signal du canal 1 par son banc de filtres */
   /*filtration de signal au centre de buffrin1 par chaque passe-bande
    du banc de filtre, donne k sous-bandes de signaux filtrés: sigfiltk
    emploi des tableaux:
   * buffrin1[256], COMPLEX X[1024],H[1024],tbvectrv[1024],coeffiltx[680]
   * et des pointeurs: *ptrcof, *tblptr[6] suivant le nbrefilt.
   *  *psigfilt, *tblptr11[6], *tblptr12[6];*/

/*============================================================
=*/
   /*si f=nbrefilt on traite le filtre passe-bas du canal 1 sinon
    c'est les passe-bandes: */
   for(f=0; f <= nbrefilt; f++)
   { /*22222222222222222 for(f=0; f < nbrefilt; f++)2222222222222222*/
   /* RAZ des vecteurs de traitement X,H pour la convolution par FFT */
```

```
for (n=0;n<N;n++)  /*rappel ici N=1024 */
     {X[n].real =0; X[n].imag =0;  H[n].real =0; H[n].imag =0;}
/*-------------------------------------------------------------*/
/*transfert des données réelles de buffrin1 vers vecteurs complexes
* pour la convolution par FFT*/
/*longdataini 356   longdata        256   longbuffer        456 */
/*for (n=0+60;n<longdata+140;n++) {X[n-60].real = buffrin1[n];}*/
/* buffrin1(60@395)--->X[0@335].real*/

/* buffrin1(100@355)--->X[0@255].real*/
for (n=0+100;n<longdata+100;n++) {X[n-100].real = buffrin1[n];}
/*-------------------------------------------------------------*/
/*for(f=0; f < nbrefilt; f++)--->ptrcof = tblptr[f];22222222222222*/
/*for(f=0; f < nbrefilt; f++)--->psigfilt = tblptr11[f];222222222222*/
/*si f=0-->psigfilt = tblptr11[0];ptrcof = tblptr[0];*/
ptrcof = tblptr[f];  psigfilt = tblptr11[f];  poffset1 = tblptr21[f];

for (n=0; n<ordN; n++){H[n].real = *(ptrcof+n);}
/*-------------------------------------------------------------*/
      /*--------/* commence les FFT sur les vecteurs X et H* /----------*/
dir=1; FFT(X,dir); FFT(H,dir);
for (n=0;n<N;n++)
  {xr= X[n].real; /* mémorise pour la 2 ieme equation*/
   X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
   X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);}

dir=-1; FFT(X,dir); /*fait la FFT inverse de X(f)-->X(n):*/
/*-------------------------------------------------------------*/
/*X[n].real contient le résultat de la convolution faite entre
* buffrin1(256) et coeffiltx(ordN), le nombre d'éléments à transférer
* vers vecteur sigfiltk[] est 256+ordN-1 ou encore longdata+ordN-1
* le tableau overlap1 permet de garder le principe de superposition
* de la convolution sur les trames successives du canal 1*/

/* convolution précédente prise en charge par la convolution courante*/

for(i=0;i < N;i++){ X[i].real = (X[i].real) + *(poffset1+i);}
for(i=0;i < N;i++){*(psigfilt+i)=X[i].real;}
for(i=0;i < N; i++){*(poffset1+i)=0;}  /*mise à zéro du vecteur offset*/
for(i=(0+longdata);i <(longdata+ordN-1);i++){*(poffset1+(i-(longdata)))=X[i].real;}
/*-------------------------------------------------------------*/
} /*2222222222222222 for(f=0; f < nbrefilt; f++)22222222222222222*/
```

```
/*==========================================================
=*/

  /*sigfilt10 à sigfilt1x: signaux filtrés par les passe-bandes
   et  sigfilt1x+1 signaux filtrés par le passe-bas du canal 1*/
  printf(" ecriture des signaux filtrés dans sigfilt1X[]\n");
  fprintf(ptrfichw3," ecriture des signaux filtrés dans sigfilt1X[] canal 1\n\n\n");
  /*sigfilt10[i],sigfilt11[i],sigfilt12[i],sigfilt13[i],sigfilt14[i],*/
  /*
  for(i=0;i <longdata;i++){fprintf( ptrfichw4,"%f\n",sigfilt10[i]);}
  */

/*==========================================================
=*/


/*QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
QQQQQQQQQQQQQQQQQQQ*/
  /* Partie sur la transformée de Hilbert pour extraire l'enveloppe
  des signaux filtrés du canal 1*/
  /*si f=0-->psigfilt = tblptr11[0]= &sigfilt10[0]; */
  /*si f=0-->pssynt1 = tblptr31[0] = &ssynt10[0]; */
  /*si f=0-->ptampon1 = tblptr41[0] = &tampon10[0] ou tampon10; */

  /* initialisation des parametres nécessaires au calculs du LMS pour
   chaque sous-bande pour chaque trame de données traitées*/
  /*           resuLMS[5]={y,h,ede,p,mu}              */
  /* y:sortie, h:coefficient, ede:erreur, p:puissance, mu:le pas*/

  for(i=0; i < 5; i++)
  {(resuLMS[i]).imag=0.0;
   if (i==4)  (resuLMS[i]).real = 0.8;
   else  (resuLMS[i]).real=0.0;   }

  /*mise à zéro des coefficient du filtre h: COMPLEX hlms[2]*/
  (hlms[0]).real=0.0; (hlms[0]).imag=0.0;
  (hlms[1]).real=0.0; (hlms[1]).imag=0.0;

  for(f=0; f < 6; f++) /*==== toutes les sous-bandes================ */
  {
   ptresulms1 = tblptr51[f] ; ptrhlms1 = tblptr61[f] ;

        /*initialisation de chaque élément complex du tableau resulms1x[]*/
```

```
    for(i=0; i < 5; i++)
      {(ptresulms1[i]).imag = (resuLMS[i]).imag;
                        if (i==4)  (ptresulms1[i]).real =(resuLMS[i]).real ;
                        else  (ptresulms1[i]).real =(resuLMS[i]).real;   }

    /*initialisation de chaque élément complex du tableau hlms1x[]*/
    for(i=0; i < 2; i++)
      {(ptrhlms1[i]).imag = (hlms[i]).imag;
       (ptrhlms1[i]).real = (hlms[i]).real;   }

   } /*======================== toutes les sous-bandes================ */

   for(f=0; f < nbrefilt; f++)
   { /*333333333333333 for(f=0; f < nbrefilt; f++)33333333333333333333333333*/
   psigfilt = tblptr11[f]; pssynt1 = tblptr31[f]; ptampon1 = tblptr41[f];
   ptresulms1 = tblptr51[f] ; ptrhlms1 = tblptr61[f] ;

/*============================================================
=*/

   /* RAZ du vecteur de traitement X pour la FFT, N=1024*/
   for (n=0;n<N;n++) {X[n].real =0; X[n].imag =0;}

/*============================================================
=*/
   /*X[0].real @X[49].real : les 50 données passées*/
   for (n=0;n<50 ;n++)          {X[n].real = *(ptampon1+n);}

   /*transfert des données réelles du signal filtré de chaque passe bande
    du banc de filtre vers vecteur complexe pour la FFT, longdata = 256*/
   /*X[50].real @X[305].real : les 256 données présentes du segment actif*/
   for (n=50;n<longdata+50 ;n++) {(X[n].real)= *(psigfilt+(n-50));}

   /*mémorise les 50 dernières données du segment filtré actif-->tampon*/
   for (n=206;n<longdata ;n++) {*(ptampon1+(n-206))= *(psigfilt+n);}

         /*------------------- la FFT de X   --------------------*/
   /*printf(" commence la FFT sur le vecteur X:\n");*/
   /* commence la FFT sur le vecteur X */
   dir=1; FFT(X,dir);
   for (n=0; n<N ;n++)
         {
         if (n < N/2)
       {X[n].real = 2*(X[n].real);
```

```
    X[n].imag = 2*(X[n].imag);}
   else
   {X[n].real = 0;
    X[n].imag = 0;}
 }
/*------------------- la FFT inverse de X   --------------------*/
/*fait la FFT inverse de X(f)-->X(n):*/
dir=-1; FFT(X,dir);

/*---- retrait de la partie imaginaire de la  FFT inverse de X ----*/
/* ------------ qui est la transformée de Hilbert -----------------*/
for (n=50;n<longdata+50 ;n++)  {tbvectrv[n-50]=X[n].imag ;}

/* ------- le module du signal analytique ou l'enveloppe -----------*/
for (n=0;n<longdata ;n++)
{
enveloppe[n] = ( (*(psigfilt+n))*(*(psigfilt+n)))+ ( tbvectrv[n]* tbvectrv[n]);
enveloppe[n] = sqrt(enveloppe[n]);
}
/*la dernière valeur calculée est mise égale à l'avant dernière
 pour réduire l'erreur de discontinuité de la fin du segment*/
enveloppe[longdata-1]=  enveloppe[longdata-2];


/*
quand ptrfichw1 = fopen("envelop10.dat", "a+") pour le premier banc
f==0 f==1 f==2 f==3 f==4 f==5
if (f==0)
{ for(i=0; i < longdata; i++){fprintf( ptrfichw1,"%f\n",enveloppe[i]);}}
*/

/*génération des signaux synthétisés COMPLEX conjugués:
fréquence fondamentale f0 ou harmonique kf0 modulée par l'enveloppe
kf0 = 2^(kk-1)*f0  kk;la sousbande du banc de filtre:0,1,2,3...
mais sans la correction de phase:
synthp[longdata] = enveloppe(t).*(exp(2*pi*kf0*t*j/fs));
synthm[longdata] = enveloppe(t).*(exp(-2*pi*kf0*t*j/fs)); canal 1*/

if (f==0) kf0=f0; else  kf0*=2;

/*fprintf(ptrfichw3,"la f0 ou kf0 est: %5.4f\n",kf0);*/

 for(n=0;n<longdata;n++)
                {
```

```
(synthp[n]).real = (enveloppe[n]) * (float)cos((n*arg1*kf0/fs));
(synthp[n]).imag = (enveloppe[n]) * (float)sin((n*arg1*kf0/fs));

(synthm[n]).real = (synthp[n]).real;
(synthm[n]).imag = -(synthp[n]).imag;
}


/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*==========================================================
=*/
  /*======== LA PARTIE TOUCHANT LE FILTRE DE WIENER LMS
=========*/
  /*          resuLMS[5]={y,h,ede,p,mu}              */

  /*==========================================================
  mise à zéro de la puissance; p=resuLMS[3]={y,h,ede,(p),mu}
  (resuLMS[3]).real=0.0; (resuLMS[3]).imag=0.0;
  initialisation du pas du LMS=resuLMS[4]={y,h,ede,p,(mu)}
  (resuLMS[4]).real = 0.8; (resuLMS[4]).imag = 0.0;
  mise à zéro des coefficient du filtre h: COMPLEX hlms[2]
  (hlms[0]).real=0.0; (hlms[0]).imag=0.0;
  (hlms[1]).real=0.0; (hlms[1]).imag=0.0;
  =========================================================*/

  /*Récupération du context du LMS pour chaque sous-bande de chaque trame*/
    for(i=0; i < 5; i++)
      { (resuLMS[i]).imag = (ptresulms1[i]).imag;
              (resuLMS[i]).real = (ptresulms1[i]).real;  }
    for(i=0; i < 2; i++)
      { (hlms[i]).imag = (ptrhlms1[i]).imag;
              (hlms[i]).real = (ptrhlms1[i]).real;  }

  /*================= début de l'algorithme du LMS==================*/
  for(n=0;n<longdata;n++)
      { /*saisie du signal d'entrée du filtre de Wiener*/
      (xlms[0]).real = (synthp[n]).real;
    (xlms[0]).imag = (synthp[n]).imag;
      (xlms[1]).real = (synthm[n]).real;
```

```
(xlms[1]).imag = (synthm[n]).imag;

/* ========== algorithme complexe du lms ==================*/
algcmplxLMS(xlms, hlms,*(psigfilt+n), resuLMS);

/*sortie du filtre de Wiener est ici optionnelle*/
(yLMS[n]).real =(resuLMS[0]).real; (yLMS[n]).imag =(resuLMS[0]).imag;

/*sorties du filtre de Wiener en suivi pour la correction de phase*/
  (vecthlms[n]).real= (hlms[0]).real; (vecthlms[n]).imag= (hlms[0]).imag;

/*Sauvegarde du context du LMS pour chaque sous-bande de chaque trame*/
for(i=0; i < 5; i++)
 { (ptresulms1[i]).imag = (resuLMS[i]).imag;
                  (ptresulms1[i]).real = (resuLMS[i]).real;  }
for(i=0; i < 2; i++)
 {  (ptrhlms1[i]).imag  = (hlms[i]).imag;
            (ptrhlms1[i]).real  = (hlms[i]).real;  }

/* =============pour développement uniquement ====================*/
/*
(lepas[n]).real=(resuLMS[4]).real; (lepas[n]).imag=(resuLMS[4]).imag;
(pwr[n]).real=(resuLMS[3]).real; (pwr[n]).imag=(resuLMS[3]).imag;
(lerreur[n]).real=(resuLMS[2]).real; (lerreur[n]).imag=(resuLMS[2]).imag;
*/
}/*================fin de l'algorithme du LMS==================*/


/*============================================================
=*/
  /*== ecriture du fichier ywiener.dat ==*/
  /*yLMS[f].real, yLMS[f].imag=0, (vecthlms[f]).real=grand,
  (vecthlms[f]).imag =grand, (lepas[f]).real=400-->1.53,
  (pwr[f]).real<<1 (pwr[f]).imag=0 ,(lerreur[f]).real=grand
  (lerreur[f]).imag=0*/

  /*
  printf(" commence ecriture du fichier ywiener.dat\n");
  ptrfichw = fopen("ywiener.dat", "a+");
       for(f = 0; f <= indexXmax; f++){fprintf(ptrfichw, "%f\n", (yLMS[f]).real);}
  fclose(ptrfichw);
  */
```

```
/*==========================================================
=*/
  /*========  LA PARTIE TOUCHANT LA SYNTHESE DU SIGNAL
=========
   ===A PARTIR DES COEFFICIENTS DE LA PHASE PRIS DANS vecthlms,
   =ET DE L'ENVELOPPE ET DE LA FREQUENCE FONDAMENTALE ou
HARMONIQUE=*/

   /*--------  coefC=vecthlms(1,:); phi=angle(coefC); ------------%
   %  signal v synthétisé avec la phase= f0 et déphassage phi
   t=1:length(coefC);
   v(t)=2*enveloppe(t).*sqrt((real(coefC(t))).^2 +
      (imag(coefC(t))).^2).*cos((2*pi*kf0*t/Fs)-phi(t));
   %-------------------------------------------------------% */

    for(n=0;n<longdata;n++)
    {
    /*calcul du déphasage retrouvé par filtre LMS de Wiener*/
    tbvectrv[n]=atan2((vecthlms[n]).imag, (vecthlms[n]).real);

    enveloppe[n]=2*enveloppe[n];  /*enveloppe(t)=2*enveloppe(t)*/

    valr1= (((vecthlms[n]).real)*((vecthlms[n]).real));/*(real(coefC(t))).^2*/

    vali1= (((vecthlms[n]).imag)*((vecthlms[n]).imag));/*(imag(coefC(t))).^2*/

    enveloppe[n]=(enveloppe[n])*(sqrt(valr1+ vali1));
    *(pssynt1+n)=(enveloppe[n])* cos((n*arg1*kf0/fs)-tbvectrv[n]);
    }


/*==========================================================
=*/


/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/
```

```
/*==========================================================
=*/
   }/*333333333333333 for(f=0; f < nbrefilt; f++) 333333333333333333333333*/


/*==========================================================
=*/
   printf(" ecriture des signaux synthétisés\n");
   fprintf(ptrfichw3," ecriture des signaux synthétisés du canal 1\n\n\n");

   /*== ecrit fichiers synthpi1, synthmi1, synthpr1, synthmr1 ==*/
   /* ssynt10[i],ssynt11[i],ssynt12[i],ssynt13[i],ssynt14[i] */
   /*
   for(i=0; i < longdata; i++){fprintf( ptrfichw1,"%f\n",ssynt10[i] );}
   */

/*==========================================================
=*/

/*QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
QQQQQQQQQQQQQQ*/



/*==========================================================
=*/
   /* Sélection de la fréquence fondamentale f0 la plus représentative
    * dans la trame courante sur le canal 2*/
   /*
   premiere approche choisie: on garde la plus grande f0
   seconde approche choisie: on garde la plus petite f0
   */

   /*
   f0=0;
   for(i=nbredemax2; i<(2*nbredemax2-1);i++){if (tbf02[i]>f0)  f0 = tbf02[i];}
   */

   /*seconde approche choisie: on garde la plus petite f0*/
   f0=1000;
   for(i=nbredemax2; i<(2*nbredemax2-1);i++){if (tbf02[i]<f0)  f0 = tbf02[i];}
```

```
printf("la f0 choisie dans la trame du canal 2 vaut %4.1f Hz\n",f0);
fprintf(ptrfichw3,"la f0 choisie dans la trame du canal 2 vaut %4.1f Hz\n",f0);


/*============================================================
=*/
tbf0[1]= f0; /*f0 choisie dans la trame courante du canal 2*/
alpha = f0/sqrt(2);          /* borne inférieure en Hertz*/
beta = 2*alpha;              /* borne supérieure en Hertz*/

/*============== largeur de la bande de transition =============*/
/*1.5;on augmente de 50% la bande de transition*/
deltaomega = pi*1.5*(f0-alpha)/fnyq ;
/*ondulation en dB pour un coefficient d'ondulation de 0.01*/
onduldB = -20*log10(0.01); /*en valeur absolue*/

/*============== détermination de l'ordre du filtre =============*/
/* si l'ordre N est un nombre pair, alors le nombre de termes M
de la fenetre sera impair (M=N+1)et nous aurons une réponse
 impulsionnelle      symétrique garantissant un déphasage linéaire*/
ordN = ceil((onduldB-8)/(2.285*deltaomega));
if ((ordN % 2)!= 0) ordN++;

/*====== détermination du Beta pour la fenetre de Kaiser =======*/
if (onduldB < 21) Beta=0;
if ((onduldB >= 21)&& (onduldB <= 50))
      Beta = 0.5842*pow((onduldB-21),0.4)+ 0.07886*(onduldB-21);
if (onduldB > 50) Beta = 0.1102*(onduldB - 8.7);

/*============== Calcul de la fenetre de Kaiser ================*/
/*Calcul de la fonction modifiée de Bessel d'ordre zéro du premier
  genre; Io(Beta)*/
/*calcul itératif sur 21 termes*/
I0Beta=1;
for(i=1;i<20;i++) { vt=pow((Beta/2),i)/factoriel[i];I0Beta += (vt*vt);}
/*============== L'équation de la fenetre de Kaiser =============*/
/*Kaiserwin[n]=Io[ Beta*sqrt(1-((n-ordN/2)/ordN/2)^2 ] / I0Beta*/
for(n=0;n <= ordN;n++)
{ x=n; /*valeur d'un int vers un float pour le calcul*/
  Vt = (x-(ordN/2))/(ordN/2);
  valeur = Beta*sqrt(1-(Vt*Vt));

   I0valeur=1;
      for(i=1;i<20;i++)  /*calcul itératif sur 21 termes*/
```

```
                    {
                vt = (pow((valeur/2),i))/(factoriel[i]) ;
                I0valeur += (vt*vt);
                    }
    Kaiserwin[n]=  I0valeur/I0Beta ; }


/*==========================================================
=*/
   /*calcule le nombre de passe bandes contenu dans le banc de filtre*/
   nbrefilt=1; /*par défaut*/
   bisdspb[0]= beta/2; bisdspb[1]= beta;
   while (2*beta < fnyq ) { beta*=2; nbrefilt++; bisdspb[nbrefilt]= beta;}
   nbsbandes[1]= nbrefilt;

   printf(" il y a %d passe-bandes dans le banc de filtre\n",nbrefilt);
   printf(" dont l'ordre N est: %d \n",ordN);
   fprintf(ptrfichw3," il y a %d passe-bandes dans le banc de filtre\n",nbrefilt);
   fprintf(ptrfichw3,"  dont l'ordre N est: %d \n",ordN);

   for(f=0; f <= nbrefilt; f++)
   {printf(" bisdspb[%d] = %5.4f\n",f,bisdspb[f]);
    fprintf(ptrfichw3,"  bisdspb[%d] = %5.4f\n",f,bisdspb[f]);
   }


/*==========================================================
=*/
   for(f=0; f < nbrefilt; f++)
   { /*f111111111111or(f=0; f < nbrefilt; f++)11111111111111111111*/
   /*pointeur courant ptrcof égale au début d'adresse du tableau correspondant
    au filtre passe-bande dans le banc de filtre*/
   ptrcof = tblptr[f];
   omegac2=(pi/fnyq)* bisdspb[f+1]; /*omegac2=(pi/fnyq)* bisdspb[1]*/
   omegac1=(pi/fnyq)* bisdspb[f]; /*omegac1=(pi/fnyq)* bisdspb[0]*/

/*==========================================================
=*/
   for(n=0;n <= ordN;n++)
   { if (n == ordN/2)
     *(ptrcof+n)=(omegac2-omegac1)/pi;
     else
     *(ptrcof+n)=(sin(omegac2*(n-(ordN/2)))-sin(omegac1*(n-(ordN/2))))/(pi*(n-
(ordN/2)));
```

```
   }

/*=============================================================
=*/
   /*le produit de la fenetre de Kaiser sur le vecteur des coefficients
     le résultat est mis dans le vecteur des coefficients */
   for(n=0; n <= ordN;n++) {*(ptrcof+n)=  *(ptrcof+n) * Kaiserwin[n];}
   } /*f111111111111or(f=0; f < nbrefilt; f++)11111111111111111111*/

/*=============================================================
=*/

   /*============ Pour le filtre passe-bas sur canal 2 ============*/
   omegac0=(pi/fnyq)* bisdspb[0]; ptrcof = tblptr[nbrefilt];
   for(n=0;n <= ordN;n++)
   { if (n == ordN/2)
     *(ptrcof+n) = omegac0/pi;
     else
     *(ptrcof+n)=sin(omegac0*(n-(ordN/2))) / ( pi*(n-(ordN/2)));

     *(ptrcof+n)=  *(ptrcof+n) * Kaiserwin[n];
   }

/*=============================================================
=*/
   printf("ecriture des coefficients dans coeffiltX[] pour canal 2\n");
   fprintf(ptrfichw3,"ecriture des coefficients dans coeffiltX[] pour canal 2\n");
   /*
   ptrfichw4 = fopen("coeff0.dat", "a+");
   for(f = 0; f <= ordN; f++){ fprintf(ptrfichw4,"%f\n",coeffilt0[f]);}
   */
   printf(" le banc de filtre pour le canal 2 est fait.\n");
   fprintf(ptrfichw3," le banc de filtre pour le canal 2 est fait.\n");

/*=============================================================
=*/

   /* partie du filtrage du signal du canal 2 par son banc de filtres */
   /*filtration de signal au centre de buffrin2 par chaque passe-bande
    du banc de filtre, donne k sous-bandes de signaux filtrés: sigfiltk
    emploi des tableaux:
   * buffrin2[256], COMPLEX X[1024],H[1024],tbvectrv[1024],coeffiltx[680]
   * et des pointeurs: *ptrcof, *tblptr[6] suivant le nbrefilt.
   *  *psigfilt, *tblptr11[6], *tblptr12[6];*/
```

```
/*==========================================================================
=*/
   /*si f=nbrefilt on traite le filtre passe-bas du canal 2 sinon
    c'est les passe-bandes: */
    for(f=0; f <= nbrefilt; f++)
   { /*22222222222222222 for(f=0; f < nbrefilt; f++)2222222222222222*/

   /* RAZ des vecteurs de traitement X,H pour la convolution par FFT */
   for (n=0;n<N;n++)  /*rappel ici N=1024 */
         {X[n].real =0; X[n].imag =0;  H[n].real =0; H[n].imag =0;}
   /*-----------------------------------------------------------------*/
   /*transfert des données réelles de buffrin2 vers vecteurs complexes
   * pour la convolution par FFT*/
   /*longdataini 356   longdata        256   longbuffer        456 */
   /* buffrin1(100@355)--->X[0@255].real*/
   for (n=0+100;n<longdata+100;n++) {X[n-100].real = buffrin2[n];}
   /*-----------------------------------------------------------------*/
   /*for(f=0; f < nbrefilt; f++)--->ptrcof = tblptr[f];22222222222222*/
   /*for(f=0; f < nbrefilt; f++)--->psigfilt = tblptr11[f];222222222222*/
   /*si f=0-->psigfilt = tblptr12[0]= sigfilt20; */
   /*ptrcof = tblptr[0]= coeffilt0; poffset2 = tblptr22[0]=offset20 */
   ptrcof = tblptr[f];   psigfilt = tblptr12[f];   poffset2 = tblptr22[f];

   for (n=0; n<ordN; n++){H[n].real = *(ptrcof+n);}
   /*-----------------------------------------------------------------*/
         /*--------/* commence les FFT sur les vecteurs X et H* /----------*/
   dir=1; FFT(X,dir); FFT(H,dir);
   for (n=0;n<N;n++)
     {xr= X[n].real; /* mémorise pour la 2 ieme equation*/
      X[n].real = ((H[n].real)*xr)-((H[n].imag)*(X[n].imag));
      X[n].imag = ((H[n].real)*(X[n].imag))+((H[n].imag)*xr);}

   dir=-1; FFT(X,dir); /*fait la FFT inverse de X(f)-->X(n):*/
   /*-----------------------------------------------------------------*/
   /*X[n].real contient le résultat de la convolution faite entre
   * buffrin2(256) et coeffiltx(ordN), le nombre d'éléments à transférer
   * vers vecteur sigfiltk[] est 256+ordN-1 ou encore longdata+ordN-1
   * le tableau offsetk[] permet de garder le principe de superposition
   * de la convolution sur les trames successives du canal 2*/
   /* convolution précédente prise en charge par la convolution courante*/

   for(i=0;i < N;i++){ X[i].real = (X[i].real) + *(poffset2+i);}
   for(i=0;i < N;i++){*(psigfilt+i)=X[i].real;}
```

```
for(i=0;i < N; i++){*(poffset2+i)=0;} /*mise à zéro du vecteur offset[]*/
/*mise à jour du vecteur offset*/
for(i=(0+longdata);i <(longdata+ordN-1);i++){*(poffset2+(i-(longdata)))=X[i].real;}
/*-------------------------------------------------------------*/
} /*2222222222222222 for(f=0; f < nbrefilt; f++)222222222222222222*/

/*===========================================================
=*/

/*sigfilt20 à sigfilt2x: signaux filtrés par les passe-bandes
 et  sigfilt2x+1 signaux filtrés par le passe-bas du canal 2*/
printf(" ecriture des signaux filtrés dans sigfilt2X[]\n");
fprintf(ptrfichw3," ecriture des signaux filtrés dans sigfilt2X[] canal 2\n\n\n");
/* sigfilt20[i],sigfilt21[i],sigfilt22[i],sigfilt23[i],sigfilt24[i]*/
/*
for(i=0;i <longdata;i++){fprintf( ptrfichw4,"%f\n",sigfilt24[i]);}
*/

/*===========================================================
=*/


/*QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
QQQQQQQQQQQQQQQ*/
/* Partie sur la transformée de Hilbert pour extraite l'enveloppe
des signaux filtrés du canal 2*/
/*si f=0-->psigfilt = tblptr12[0]= &sigfilt20[0];*/
/*si f=0-->pssynt2 = tblptr32[0] = &ssynt20[0];*/
/*si f=0-->ptampon2 = tblptr42[0] = &tampon20[0] ou tampon20;*/

/* initialisation des parametres nécessaires au calculs du LMS pour
 chaque sous-bande pour chaque trame de données traitées*/
/*          resuLMS[5]={y,h,ede,p,mu}           */
/* y:sortie, h:coefficient, ede:erreur, p:puissance, mu:le pas*/

for(i=0; i < 5; i++)
{(resuLMS[i]).imag=0.0;
 if (i==4)  (resuLMS[i]).real = 0.8;
 else  (resuLMS[i]).real=0.0;   }

/*mise à zéro des coefficient du filtre h: COMPLEX hlms[2]*/
(hlms[0]).real=0.0; (hlms[0]).imag=0.0;
(hlms[1]).real=0.0; (hlms[1]).imag=0.0;
```

```
  for(f=0; f < 6; f++) /*==== toutes les sous-bandes sur canal 2 ==== */
  {
   ptresulms2 = tblptr52[f] ; ptrhlms2 = tblptr62[f] ;

        /*initialisation de chaque élément complex du tableau resulms2x[]*/
    for(i=0; i < 5; i++)
      {(ptresulms2[i]).imag = (resuLMS[i]).imag;
                        if (i==4)  (ptresulms2[i]).real =(resuLMS[i]).real ;
                        else  (ptresulms2[i]).real =(resuLMS[i]).real;   }

   /*initialisation de chaque élément complex du tableau hlms2x[]*/
   for(i=0; i < 2; i++)
     {(ptrhlms2[i]).imag = (hlms[i]).imag;
      (ptrhlms2[i]).real = (hlms[i]).real;   }

  } /*==================== toutes les sous-bandes sur canal 2======= */

  for(f=0; f < nbrefilt; f++)
  { /*333333333333333 for(f=0; f < nbrefilt; f++)3333333333333333333*/
  psigfilt = tblptr12[f];  pssynt2 =  tblptr32[f]; ptampon2 = tblptr42[f];
  ptresulms2 = tblptr52[f] ; ptrhlms2 = tblptr62[f] ;

/*============================================================
=*/

  /* RAZ des vecteurs de traitement X,H pour la FFT */
  for (n=0;n<N;n++) {X[n].real =0; X[n].imag =0;}

/*============================================================
=*/
  /*X[0].real @X[49].real : les 50 données passées*/
  for (n=0;n<50 ;n++)         {X[n].real = *(ptampon2+n);}

  /*transfert des données réelles du signal filtré de chaque passe bande
   du banc de filtre vers vecteur complexe pour la FFT, longdata = 256*/
  /*X[50].real @X[305].real : les 256 données présentes du segment actif*/
  for (n=50;n<longdata+50 ;n++) {(X[n].real)= *(psigfilt+(n-50));}

  /*mémorise les 50 dernières données du segment actif dans le tampon*/
  for (n=206;n<longdata ;n++) {*(ptampon2+(n-206))= *(psigfilt+n);}

       /*------------------ la FFT de X    --------------------*/
 /* printf(" commence la FFT sur le vecteur X:\n");*/
  /* commence la FFT sur le vecteur X */
```

```
dir=1; FFT(X,dir);
for (n=0; n<N ;n++)
      {
      if (n < N/2)
    {X[n].real = 2*(X[n].real);
     X[n].imag = 2*(X[n].imag);}
    else
    {X[n].real = 0;
     X[n].imag = 0;}
  }
/*------------------- la FFT inverse de X   --------------------*/
/*fait la FFT inverse de X(f)-->X(n):*/
dir=-1; FFT(X,dir);

/*---- retrait de la partie imaginaire de la  FFT inverse de X ----*/
/* ------------ qui est la transformée de Hilbert -----------------*/
for (n=50;n<longdata+50 ;n++)  {tbvectrv[n-50]=X[n].imag ;}

/* ------- le module du signal analytique ou l'enveloppe ----------*/
for (n=0;n<longdata;n++)   /*L'enveloppe du signal*/
{
enveloppe[n] = ( (*(psigfilt+n))*(*(psigfilt+n)))+ ( tbvectrv[n]* tbvectrv[n]);
enveloppe[n] = sqrt(enveloppe[n]);
}
/*la dernière valeur calculée est mise égale à l'avant dernière
 pour réduire l'erreur de discontinuité de la fin du segment*/
enveloppe[longdata-1]=  enveloppe[longdata-2];

 /*
quand ptrfichw1 = fopen("envelop10.dat", "a+") pour le premier banc
f==0 f==1 f==2 f==3 f==4 f==5
if (f==0)
{ for(i=0; i < longdata; i++){fprintf( ptrfichw1,"%f\n",enveloppe[i]);}}
*/

/*génération des signaux synthétisés COMPLEX conjugués:
fréquence fondamentale f0 ou harmonique kf0 modulée par l'enveloppe
kf0 = 2^(kk-1)*f0  kk;la sousbande du banc de filtre:0,1,2,3...
mais sans la correction de phase:
synthp[longdata] = enveloppe(t).*(exp(2*pi*kf0*t*j/fs));
synthm[longdata] = enveloppe(t).*(exp(-2*pi*kf0*t*j/fs)); canal 1*/

if (f==0) kf0=f0; else  kf0*=2;
```

```
/*fprintf(ptrfichw3,"la f0 ou kf0 est: %5.4f\n",kf0);*/

 for(n=0;n<longdata;n++)
            {
   (synthp[n]).real = (enveloppe[n]) * (float)cos((n*arg1*kf0/fs));
   (synthp[n]).imag = (enveloppe[n]) * (float)sin((n*arg1*kf0/fs));

   (synthm[n]).real = (synthp[n]).real;
   (synthm[n]).imag = -(synthp[n]).imag;
  }


/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*==============================================================
=*/
 /*======== LA PARTIE TOUCHANT LE FILTRE DE WIENER LMS
=========*/
 /*           resuLMS[5]={y,h,ede,p,mu}               */

 /*==============================================================
 mise à zéro de la puissance; p=resuLMS[3]={y,h,ede,(p),mu}
 (resuLMS[3]).real=0.0; (resuLMS[3]).imag=0.0;
 initialisation du pas du LMS=resuLMS[4]={y,h,ede,p,(mu)}
 (resuLMS[4]).real = 0.8; (resuLMS[4]).imag = 0.0;
 mise à zéro des coefficient du filtre h: COMPLEX hlms[2]
 (hlms[0]).real=0.0; (hlms[0]).imag=0.0;
 (hlms[1]).real=0.0; (hlms[1]).imag=0.0;
 ============================================================*/

 /*Récupération du context du LMS pour chaque sous-bande de chaque trame*/
   for(i=0; i < 5; i++)
     { (resuLMS[i]).imag = (ptresulms2[i]).imag;
              (resuLMS[i]).real = (ptresulms2[i]).real;   }
   for(i=0; i < 2; i++)
     { (hlms[i]).imag = (ptrhlms2[i]).imag;
              (hlms[i]).real = (ptrhlms2[i]).real;   }

 /*================ début de l'algorithme du LMS==================*/
 for(n=0;n<longdata;n++)
```

```
    { /*saisie du signal d'entrée du filtre de Wiener*/
     (xlms[0]).real = (synthp[n]).real;
    (xlms[0]).imag = (synthp[n]).imag;
     (xlms[1]).real = (synthm[n]).real;
    (xlms[1]).imag = (synthm[n]).imag;

    /* ========== algorithme complexe du lms ==================*/
    algcmplxLMS(xlms, hlms,*(psigfilt+n), resuLMS);

    /*sortie du filtre de Wiener est ici optionnelle*/
    (yLMS[n]).real =(resuLMS[0]).real; (yLMS[n]).imag =(resuLMS[0]).imag;

    /*sorties du filtre de Wiener en suivi pour la correction de phase*/
     (vecthlms[n]).real= (hlms[0]).real; (vecthlms[n]).imag= (hlms[0]).imag;

   /*Sauvegarde du context du LMS pour chaque sous-bande de chaque trame*/
   for(i=0; i < 5; i++)
    { (ptresulms2[i]).imag = (resuLMS[i]).imag;
                  (ptresulms2[i]).real = (resuLMS[i]).real;  }
   for(i=0; i < 2; i++)
    { (ptrhlms2[i]).imag  = (hlms[i]).imag;
              (ptrhlms2[i]).real  = (hlms[i]).real;  }

   /* ============pour développement uniquement ====================*/
   /*
   (lepas[n]).real=(resuLMS[4]).real; (lepas[n]).imag=(resuLMS[4]).imag;
   (pwr[n]).real=(resuLMS[3]).real; (pwr[n]).imag=(resuLMS[3]).imag;
   (lerreur[n]).real=(resuLMS[2]).real; (lerreur[n]).imag=(resuLMS[2]).imag;
   */
   }/*================fin de l'algorithme du LMS==================*/


/*============================================================
=*/
  /*== ecriture du fichier ywiener.dat ==*/
  /*yLMS[f].real, yLMS[f].imag=0, (vecthlms[f]).real=grand,
  (vecthlms[f]).imag =grand, (lepas[f]).real=400-->1.53,
  (pwr[f]).real<<1 (pwr[f]).imag=0 ,(lerreur[f]).real=grand
  (lerreur[f]).imag=0*/

  /*
  printf(" commence ecriture du fichier ywiener.dat\n");
  ptrfichw = fopen("ywiener.dat", "a+");
       for(f = 0; f <= indexXmax; f++){fprintf(ptrfichw, "%f\n", (yLMS[f]).real);}
```

```
  fclose(ptrfichw);
  */


/*==========================================================
=*/
  /*======== LA PARTIE TOUCHANT LA SYNTHESE DU SIGNAL
=========
   ===A PARTIR DES COEFFICIENTS DE LA PHASE PRIS DANS vecthlms,
   =ET DE L'ENVELOPPE ET DE LA FREQUENCE FONDAMENTALE ou
HARMONIQUE=*/

   /*--------  coefC=vecthlms(1,:); phi=angle(coefC); ------------%
  %  signal v synthétisé avec la phase= f0 et déphassage phi
  t=1:length(coefC);
  v(t)=2*enveloppe(t).*sqrt((real(coefC(t))).^2 +
     (imag(coefC(t))).^2).*cos((2*pi*kf0*t/Fs)-phi(t));
  %-------------------------------------------------------% */

   for(n=0;n<longdata;n++)
   {
   /*calcul du déphasage retrouvé par filtre LMS de Wiener*/
   tbvectrv[n]=atan2((vecthlms[n]).imag, (vecthlms[n]).real);

   enveloppe[n]=2*enveloppe[n];  /*enveloppe(t)=2*enveloppe(t)*/

   valr1= (((vecthlms[n]).real)*((vecthlms[n]).real));/*(real(coefC(t))).^2*/

   vali1= (((vecthlms[n]).imag)*((vecthlms[n]).imag));/*(imag(coefC(t))).^2*/

   enveloppe[n]=(enveloppe[n])*(sqrt(valr1+ vali1));
   *(pssynt2+n)=(enveloppe[n])* cos((n*arg1*kf0/fs)-tbvectrv[n]);
   }


/*==========================================================
=*/


/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/

/*WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW*/
```

```
/*==========================================================
=*/
   }/*333333333333333 for(f=0; f < nbrefilt; f++)333333333333333333*/


/*==========================================================
=*/
   printf(" ecriture des signaux synthétisés\n");
   fprintf(ptrfichw3," ecriture des signaux synthétisés du canal 2\n\n\n");
   /*
   if (f==0)
   { for(i=0; i < longdata; i++){fprintf( ptrfichw1,"%f\n",*(pssynt2+i));}}
   */
   /*ssynt20[i],ssynt21[i],ssynt22[i],ssynt23[i],ssynt24[i]*/
   /*
   for(i=0; i < longdata; i++){fprintf( ptrfichw1,"%f\n",ssynt24[i] );}
   */

/*==========================================================
=*/

/*QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
QQQQQQQQQQQQQ*/


   /*partie prédictive du rehausement du signal à insérer ici pour
        le canal 1 et le canal 2*/
    /*Les données sont dans les signaux synthétisés du canal 1 & 2
    plus on ajoute la contribution des signaux filtrés du canal 1
    & 2par les passe-bas*/

/*==========================================================
=*/
   kf0 = tbf0[0]; if (kf0 < tbf0[1])  kf0 = tbf0[1];
   sbdes = nbsbandes[0];  if (sbdes < nbsbandes[1])  sbdes = nbsbandes[1];
   for(i=0;i<longdata;i++){tbvectrv[i]=0.0;}


/*PRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRP
RPRPRPRPR*/
   for(f=0; f < sbdes; f++)
   {
```

```
/*si f=0-->pssynt1 = tblptr31[0] = &ssynt10[0] = ssynt10 ; */
pssynt1 = tblptr31[f];
/*si f=0-->pssynt2 = tblptr32[0] = &ssynt20[0]= ssynt20 ;*/
pssynt2 =  tblptr32[f];

if (f!=0)  kf0*=2;
/*Nb: nombre de coefficients du filtre predictif proportionnel a kf0
      en terme du nombre d'échantillon de fs*/
Nb = (int)ceil(fs/kf0);
/* ---------------------- initialisations: ----------------------*/
for(i=0;i<longdata;i++){b[i]=0.0; vinb[i]=0.0;}
for(i=0;i<5;i++){vdelai[i]=0.0;}

rsltSEPRD[3]= 0.999;   /* lambda */
rsltSEPRD[4]= 0.0;     /* pwrx */
rsltSEPRD[5]= 0.0;     /* pwry */
rsltSEPRD[6]= 0.001;   /* mub */
rsltSEPRD[7]= 0.001;   /* muVW */

VW[0]=1.0; VW[1]=1.0;
/* ------------------------------------------------------------*/

for(i=0;i<longdata;i++)
{
 vinx[0]=*(pssynt1+i); vinx[1]=*(pssynt2+i);

 fseprd( vinx, VW, vinb, b, vdelai, rsltSEPRD, Nb);

 /* (ye) sortie y estimée du prédicteur: y(n-Delai)*b */
 tbvectrv[i]= tbvectrv[i]+ rsltSEPRD[1];
 /* -----Pour déverminage sur le developpement ------------
 (ee) erreur entre la sortie y et ye
 tbvectrv1[i]= rsltSEPRD[2];
 (y) sortie de la matrice de séparation W
 tbvectrv[i]= rsltSEPRD[0];
 */
} /* Fin de ====for(i=0;i<longdata;i++)====================*/

} /* Fin de ====for(f=0; f < sbdes; f++)====================*/

/*PRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPRPR
RPRPRPRPR*/
 /*contributions des signaux des passe-bas sur canal 1 & 2 */
 poffset1 = tblptr11[sbdes]; poffset2 = tblptr12[sbdes];
```

```
   for(i=0;i<longdata;i++)
   { tbvectrv[i]= tbvectrv[i]+ *(poffset1+i)+ *(poffset2+i); }

   /* ------------------------------------------------------------*/
   /*Ecriture du signal rehaussé issu des signaux synthétisés sur canal 1&2*/
   printf(" commence ecriture du fichier sgrehaus.dat\n");
   fprintf(ptrfichw3,"commence ecriture du fichier sgrehaus.dat signal rehaussé\n\n\n");

   for(i=0;i <longdata;i++){fprintf( ptrfichw4,"%f\n",tbvectrv[i]);}

/*============================================================
=*/


/*============================================================
=*/
   /*==== Le reste du traitement global doit s'insérer ci-haut ====*/

/*============================================================
=*/

   /* transfert les derniers 200 éch de buffrin au début de buffrin*/
   for(i=longbuffer-200;i<longbuffer;i++)  /*i=256,i<456,i++*/
    {buffrin1[i-longdata]= buffrin1[i]; buffrin2[i-longdata]= buffrin2[i];}

   /* changement de la longueur de fenetre de lecture*/
   /* longdataini=356 premiere lecture, longdata=256 lecture suivante*/
    if (segmnt==longdataini)  segmnt = longdata;

/*============================================================
=*/

   }/*%%%%%%%%%%%%%% fin de for (r=0;r<8;r++)
%%%%%%%%%%%%%%%%%%%%%%%%%%%*/


/*============================================================
=*/
   printf(" debut de la fermeture des fichiers \n");
   /*ferme le fichier en lecture sigcan1.dat appelé*/
   fclose(ptrfichr1);
   /*ferme le fichier en lecture sigcan2.dat appelé*/
   fclose(ptrfichr2);
   /*ferme le fichier en écriture sur canal 1 appelé*/
```

```
  /*fclose(ptrfichw1);*/
  /*ferme le fichier en écriture fichTO2.dat appelé*/
  /* fclose(ptrfichw2); */
  /*ferme le fichier en écriture fchrsult.dat appelé*/
  fclose(ptrfichw3);
  /*ferme le fichier en écriture auxiliaire pour "sgrehaus.dat" appelé*/
  fclose(ptrfichw4);

/*============================================================
=*/
  /*--pour quitter la fenêtre DOS sous environnement Windows------*/
  printf(" Entrez un nombre entier pour quitter:\n");
       scanf("%d",&quit); printf(" merci, à la prochaine.\n");

       }/*================== Fin du programme principal
================*/

/*##########################################################################
#*/

/*========== définitions des fonctions du programme ==============*/
/*FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF*/

/*======== fonction: void FFT(COMPLEX *Y, int dir) =================*/
/*=== vecteur d'entrée, flag de la FFT dir ou inv ==================*/
 void FFT(COMPLEX *Y, int dir)
 {
 COMPLEX temp1,temp2;     /*temporary storage variables        */
 double Wi;
 int i,j,k;              /*loop counter variables             */
 int upper_leg, lower_leg; /*index of upper/lower butterfly leg  */
 int leg_diff;           /*difference between upper/lower leg  */
 int num_stages=0;       /*number of FFT stages, or iterations */
 int index, step;        /*index and step between twiddle factor*/
 i=1;              /* log(base 2) of # of points = # of stages  */
 do
  {
  num_stages+=1;
  i=i*2;
  } while (i!=N);

 leg_diff=N/2; /*starting difference between upper & lower legs*/
 step=1024/N;  /*step between values in twiddle.h  step=512/N; */
```

```
for (i=0;i<num_stages;i++)  /*for N-point FFT              */
 {
 index=0;
 for (j=0;j<leg_diff;j++)
  {
  for (upper_leg=j;upper_leg<N;upper_leg+=(2*leg_diff))
   {
   lower_leg=upper_leg+leg_diff;
   temp1.real=(Y[upper_leg]).real + (Y[lower_leg]).real;
   temp1.imag=(Y[upper_leg]).imag + (Y[lower_leg]).imag;
   temp2.real=(Y[upper_leg]).real - (Y[lower_leg]).real;
   temp2.imag=(Y[upper_leg]).imag - (Y[lower_leg]).imag;
   Wi=(w[index]).imag;
   if (dir==-1) Wi=-Wi;
   (Y[lower_leg]).real=temp2.real*(w[index]).real-temp2.imag*Wi;
   (Y[lower_leg]).imag=temp2.real*Wi+temp2.imag*(w[index]).real;
   (Y[upper_leg]).real=temp1.real;
   (Y[upper_leg]).imag=temp1.imag;
   }
  index+=step;
  }
 leg_diff=leg_diff/2;
 step*=2;
 }
 if (dir==-1)
                 {
                 for (i=0;i<(N-1);i++)
                 {
                 (Y[i]).real = ((Y[i]).real)/N;
                 (Y[i]).imag = ((Y[i]).imag)/N;
        }
                 }

 j=0;
 for (i=1;i<(N-1);i++) /*bit reversal for resequencing data*/
 {
  k=N/2;
  while (k<=j)
   {
   j=j-k;
   k=k/2;
   }
  j=j+k;
  if (i<j)
```

```
  {
  temp1.real=(Y[j]).real;   temp1.imag=(Y[j]).imag;
  (Y[j]).real=(Y[i]).real;  (Y[j]).imag=(Y[i]).imag;
  (Y[i]).real=temp1.real;   (Y[i]).imag=temp1.imag;
  }
 }
 } /* fin de la fonction: void FFT(COMPLEX *Y, int dir, int N)*/
/*================================================================
========*/

 /*======================== fonction  seuilmin()
========================*/
 /*======== trouve le seuil minimal pour les valeurs maximales  ========*/
 float seuilmin(int indexY,float y[])
{
 int i;
 float valmax=0;
 for(i = 0; i <= indexY ; i++)
         {
   if (y[i]> valmax) valmax= y[i];
   }
  /*le seuil est fixé à 25% de la valeur maximale*/
   return (0.25*valmax);
} /* fin de la fonction: float seuilmin(int indexY,float y[])*/
/*================================================================
========*/

/*=================== fonction deff0(mindesmax,y2,y3)
===================*/
/*cherche les maxima de la transformee en ondelettes*/
/*lde y3:1016, 1016-456=560,560/2=280 , indexXmax=455*/
int deff0(float mindesmax,float y2[],float y3[])
{ /* echret=280,extsup=458,indexXmax=455, y2=tbf01/2, y3=tbvectrv */
int i;       /*indexe générique*/
int pmax=0;  /*indexe des positions des max trouvés dans tableau y2 */
/*float ffond; /* valeur de la fréquence fondamentale*/

/*transfert les données pertinentes après le premier élément du tableau*/
for (i=echret ;i<=(echret+indexXmax); i++ )
    { y3[i-echret+1]= y3[i];}  /*y3(1)=y3(280)--->y3(456)=y3(735)*/
     /*met le 1er élément y3[0] inférieur au 1er élément des données*/
       y3[0]= y3[1]-1;
     /*rend inférieur l'élément qui suit le dernier élément des données*/
       y3[indexXmax+2]= y3[echret+indexXmax]-1;  /*y3[457]=y3[735]-1*/
```

```
/*-----------------------------------------------------------------*/
/*première différentiation*/
for(i = 0; i <= indexXmax+2 ; i++) {y3[i+extsup]= y3[i+1]-y3[i];}
/*i: [0 @ 455+2] ---------> y3: [458 @ 915] */
/* fixe toutes les données différenciées par le signe +1 ou -1*/
for(i = 0; i <= indexXmax+1 ; i++)
      {        if (y3[i+extsup]>=0)   y3[i+extsup]=1;
      else  y3[i+extsup]=-1; } /*y3: [458 @ 914]*/
/*-----------------------------------------------------------------*/
/*deuxième différentiation, y3:[458 @ 914] */
for(i = 0+extsup; i <= indexXmax+1+extsup ; i++){y3[i]= y3[i+1]-y3[i];}
/*-----------------------------------------------------------------*/
 /*for(i = 0+extsup+100; i <= indexXmax+extsup-100 ; i++) */
 /* trouve les max:(val négative)*/
 for(i = 0+extsup; i <= indexXmax+extsup ; i++)
   { /*y3(256):[558 @ 813] cmp a y3(256):[100 @ 356]*/
      if ((y3[i]<0) && (y3[i-(extsup-1)]>mindesmax))
               /*retient la position du max de y3 dans y2*/
                 { y2[pmax]=(i-(extsup));pmax++;}
    }
/*-----------------------------------------------------------------*/
/* met à zéro les valeurs du tabl: y2=tbf0 après pmax*/
for(i = pmax; i < 40 ; i++) {y2[i]=0;}
/*-----------------------------------------------------------------*/
for(i = 0 ; i < pmax-1 ; i++) {y2[i+pmax]= 8000/(y2[i+1]-y2[i]);}

/* offset de 100  */
/*for(i = 0 ; i < pmax ; i++) {if (y2[i]>99) y2[i]=y2[i]-100;}*/
for(i = 0 ; i < pmax ; i++) {y2[i]=y2[i]-100;}

/* sélection des fréquences fondamentales entre 70 et 600 Hz*/
/*
for(i = pmax ; i < 2*pmax-1 ; i++)
{if ((y2[i]<60) || (y2[i]>1000) )  y2[i]=0;}
*/
for(i = pmax ; i < 2*pmax-1 ; i++)
{
      if (y2[i]<60)  y2[i]=70;
      if (y2[i]>600) y2[i]=600;
}

/*-----------------------------------------------------------------*/
return pmax;
```

```
}/*========== fin de int deff0(mindesmax,y2,y3)
==========================*/

/*algcmplxLMS(COMPLEX *pxlms, COMPLEX *phlms, float d, COMPLEX *rslt)*/
/*
  xlms: signal d'entrée
  hlms: coefficients dufiltre de Wiener
  d: signal désiré au temps n
  rslt: tableau des résultats du LMS aussi resuLMS[5]={y,h,ede,p,mu};
  rslt[5]={y,h,ede,p,mu}=
  sortie y ,coefficient h, erreur, puissance, mu (le pas)
*/
void algcmplxLMS(COMPLEX *pxlms, COMPLEX *phlms, float d, COMPLEX *rslt)
{
   COMPLEX dcmpx,ede;

   float vr1, vr2, vi1, vi2, pin, pmem, mu ;

   /*saisie du signal désiré (réel) dans sa forme complexe*/
   dcmpx.real=d; dcmpx.imag=0.0;

   /*ylms = hlms'*xlms; = resuLMS[0]= y =rslt[0]={(y),h,ede,p,mu}=======*/
   /*     hlms' attention cela veut transposée et conjuguée          */

   vr1= (((phlms[0]).real)*((pxlms[0]).real))+(((phlms[0]).imag)*((pxlms[0]).imag));
   vi1= (((phlms[0]).real)*((pxlms[0]).imag))-(((phlms[0]).imag)*((pxlms[0]).real));
   vr2= (((phlms[1]).real)*((pxlms[1]).real))+(((phlms[1]).imag)*((pxlms[1]).imag));
   vi2= (((phlms[1]).real)*((pxlms[1]).imag))-(((phlms[1]).imag)*((pxlms[1]).real));
   (rslt[0]).real =  vr1+vr2;  (rslt[0]).imag =  vi1+vi2;

   /*ede = d - ylms;
resuLMS[2]={y,h,(ede),p,mu}==========================*/
   /*erreur entre le signal désiré et celui estimé, calcul complexe*/
   (rslt[2]).real =  dcmpx.real - ((rslt[0]).real);
   (rslt[2]).imag =  dcmpx.imag - ((rslt[0]).imag);
   ede.real=(rslt[2]).real;  ede.imag=(rslt[2]).imag;

   /* calcul de la puissance instantanée à l'entrée====================*/
   /*            resuLMS[3]={y,h,ede,(p),mu}                  */
   /*if xlms(1)*xlms(1)' > puiss % quand xlms est complexe
   puiss = xlms(1)*xlms(1)'; end  %retient la valeur max des données*/

   /*puissance mémorisée valeur réelle*/
   pmem =((rslt[3]).real);
```

```
  /*puissance courante saisie à l'entrée*/
  pin =(((pxlms[0]).real)*((pxlms[0]).real))+(((pxlms[0]).imag)*((pxlms[0]).imag));

  /*retient que la valeur max des puissances données*/
  if (pin > pmem)      {(rslt[3]).real = pin; pmem = pin;  }

        /*mise à jour du pas adaptatif du LMS; mu
==========================*/
  /*initialisation du pas du LMS  resuLMS[4]={y,h,ede,p,(mu)}*/
  /* mu = rslt[4].real;  possible que cela ne soit pas nécessaire*/
  /* mu = 1/(length(xlms)* puiss + 0.002);*/
  mu=1/(2 * pmem + 0.002); /*mu=1/(508 * pmem + 0.002);*/
  (rslt[4]).real = mu;

  /*mise à jour de h:
================================================*/
  /* hlms = hlms + mulms * (xlms * ede');      COMPLEX xlms[2],hlms[2]*/

  /* (mulms *  xlms); produit de nombre réels et complexe*/
  (pxlms[0]).real= mu * ((pxlms[0]).real);
  (pxlms[0]).imag= mu * ((pxlms[0]).imag);

  (pxlms[1]).real= mu * ((pxlms[1]).real);
  (pxlms[1]).imag= mu * ((pxlms[1]).imag);

  /*(mu* xlms * ede'); produit de nombre complexe, ede'=conjugué de ede*/
  ede.imag = -(ede.imag);  /* ede'; conjugué de ede*/
  vr1= (((pxlms[0]).real) * (ede.real)) - (((pxlms[0]).imag) * (ede.imag));
  vi1= (((pxlms[0]).real) * (ede.imag)) + (((pxlms[0]).imag) * (ede.real));

  /* hlms = hlms + mu * xlms * ede';*/
  (phlms[0]).real = ((phlms[0]).real) +  vr1;
  (phlms[0]).imag = ((phlms[0]).imag) +  vi1;
  /*-----------------------------------------------------------------*/

  vr2= (((pxlms[1]).real) * (ede.real)) - (((pxlms[1]).imag) * (ede.imag));
  vi2= (((pxlms[1]).real) * (ede.imag)) + (((pxlms[1]).imag) * (ede.real));

  /* hlms = hlms + mu * (xlms * ede');*/
  (phlms[1]).real = ((phlms[1]).real) +  vr2;
  (phlms[1]).imag = ((phlms[1]).imag) +  vi2;
 }
```

```
/*fin de algcmplxLMS(COMPLEX *pxlms, COMPLEX *phlms, float d, COMPLEX
*rslt)*/

/*============================================================
==========*/
 void fseprd(float *vinx, float *VW, float *vinb, float *b, float *vdelai, float
*rsltSEPRD, int Nb)
        {
  /* fonction fseprd avec comme arguments d'entrée:
  vinx : vecteur d'entrée des  deux canaux
  VW   : vecteur ligne de la matrice de séparation W
  vinb : tampon d'entrée du filtre prédicteur FIR
  b :    filtre prédicteur FIR
  vdelai: vecteur de délai à la sortie de VW égale à 1 délai
  rsltSEPRD: tableau des résultats de la séparation et de la prédiction
          {y, ye, ee, lambda, pwrx, pwry, mub, muVW, b, w};
  Nb : nombre de coefficients de b
  */
   int i,ii,j;
   float accum, lambda, pwrx, pwry, pin1, pin2, mub, muVW, ee ;

   /* (y) sortie de la matrice de séparation W*/
   rsltSEPRD[0]= VW[0]* vinx[0] + VW[1]* vinx[1];

   /*delai de un échantillon*/
   for(ii = Nb; ii >0; ii--) { vinb[ii] =  vinb[ii-1];}
   vinb[0]= vdelai[0];  vdelai[0] = rsltSEPRD[0];

   /* (ye) sortie du prédicteur: y(n-Delai)*b */
   accum = 0.0;
   for(j = 0; j < Nb; j++) {accum += (b[j] * vinb[j]);}
   rsltSEPRD[1]=accum;

   /* (ee) erreur entre la sortie y et ye */
   rsltSEPRD[2] = rsltSEPRD[0]- rsltSEPRD[1]; ee = rsltSEPRD[2];


   /* mémorise la plus grande puissance instantanée parmi vinb & vinx */
   lambda = rsltSEPRD[3]; pwrx = rsltSEPRD[4]; pwry = rsltSEPRD[5];
   /*puissance courante saisie à l'entrée de W prend la plus grande
   des valeurs d'entrée*/
   pin1 = vinx[0]*vinx[0]; pin2 = vinx[1]*vinx[1];
   if (pin1 < pin2)  pin1 = pin2;
   if (pin1 > pwrx )  {pwrx =  pin1; rsltSEPRD[4] = pin1;}
```

```
  /*puissance courante saisie à l'entrée de b*/
  pin1 = vinb[0] * vinb[0];
  if (pin1 > pwry ) {pwry =  pin1; rsltSEPRD[5] = pin1;}

  /*mise à jour des pas adaptatifs mub et muVW*/
  mub = lambda /( Nb*pwry + 0.002);
  muVW = (1-lambda)/(2*pwrx + 0.002);

  /*mise à jour des coefficents de b et VW */
  for(i = 0; i < Nb; i++) {b[i] += mub * vinb[i]* ee ;}
  VW[0] -= muVW * vinx[0]* ee ; VW[1] -= muVW * vinx[1]* ee ;
       }
 /* ============= fin de la fonction: fseprd ============================*/

/*FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF*/
/*================== Fin du fichier rehausprl.c
=========================*/
```

Imprimé des étapes de l'exécutable

Commence lecture fichiers sigcan1.dat et sigcan2.dat:
avec le programme rehausprl.c

le nbre de donnees est 4201
le nbre des itérations est de 17
==========
le nbre n sur canal 1 est: 356
la moyenne sur canal 1 est: -0.174
la variance sur canal 1 est: 0.11115
le nbre n sur canal 2 est: 356
la moyenne sur canal 2 est: -0.200
la variance sur canal 2 est: 0.11111
le nbre de data est: 356
le nbre de segment lu est: 1

==========
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 5.9652
le nombre de max dans le signal 1 est: 3

```
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 69.0 Hz a echantillon 212
 f0[1]= 67.8 Hz a echantillon 330
 dernposmax1= 212


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 6.8195
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 69.6 Hz a echantillon 211
 f0[1]= 67.2 Hz a echantillon 330
 dernposmax2= 211


la f0 choisie dans la trame du canal 1 vaut 67.8 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 600
  bisdspb[0] = 47.9394
  bisdspb[1] = 95.8789
  bisdspb[2] = 191.7578
  bisdspb[3] = 383.5155
  bisdspb[4] = 767.0311
  bisdspb[5] = 1534.0621
  bisdspb[6] = 3068.1243
ecriture des coefficients dans coeffiltX[] pour canal 1
 le banc de filtre pour le canal 1 est fait.
 ecriture des signaux filtrés dans sigfilt1X[] canal 1


 ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 67.2 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 604
  bisdspb[0] = 47.5366
  bisdspb[1] = 95.0732
  bisdspb[2] = 190.1464
  bisdspb[3] = 380.2927
  bisdspb[4] = 760.5854
  bisdspb[5] = 1521.1709
  bisdspb[6] = 3042.3418
```

```
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


commence ecriture du fichier sgrehaus.dat signal rehaussé


==================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.061
la variance sur canal 1 est: 0.17408
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.070
la variance sur canal 2 est: 0.19256
le nbre de data est: 612
le nbre de segment lu est: 2


==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 6.3421
le nombre de max dans le signal 1 est: 3
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 64.5 Hz a echantillon 78
 f0[1]= 70.0 Hz a echantillon 242
 dernposmax1= 242


 commence la transformee en ondelettes sur canal 2:
 la transformee en ondelette sur le canal 2 est faite.
 le seuil de la trfond du canal 2 est: 7.2583
 le nombre de max dans le signal 2 est: 3
 les frequences fondamentales dans la fenetre du signal 2 sont:
  f0[0]= 64.5 Hz a echantillon 78
  f0[1]= 70.0 Hz a echantillon 242
  dernposmax2= 242


 la f0 choisie dans la trame du canal 1 vaut 64.5 Hz
```

il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 630
bisdspb[0] = 45.6198
bisdspb[1] = 91.2396
bisdspb[2] = 182.4792
bisdspb[3] = 364.9583
bisdspb[4] = 729.9167
bisdspb[5] = 1459.8334
bisdspb[6] = 2919.6667
ecriture des coefficients dans coeffiltX[] pour canal 1
le banc de filtre pour le canal 1 est fait.
ecriture des signaux filtrés dans sigfilt1X[] canal 1

ecriture des signaux synthétisés du canal 1

la f0 choisie dans la trame du canal 2 vaut 64.5 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 630
bisdspb[0] = 45.6198
bisdspb[1] = 91.2396
bisdspb[2] = 182.4792
bisdspb[3] = 364.9583
bisdspb[4] = 729.9167
bisdspb[5] = 1459.8334
bisdspb[6] = 2919.6667
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.011
la variance sur canal 1 est: 0.23483
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.011

```
la variance sur canal 2 est: 0.26837
le nbre de data est: 868
le nbre de segment lu est: 3


================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 4.6983
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 65.6 Hz a echantillon 109
 f0[1]= 70.0 Hz a echantillon 268
 f0[2]= 93.0 Hz a echantillon 354
 dernposmax1= 109


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.3767
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 65.0 Hz a echantillon 109
 f0[1]= 70.0 Hz a echantillon 267
 f0[2]= 93.0 Hz a echantillon 353
 dernposmax2= 109


la f0 choisie dans la trame du canal 1 vaut 65.6 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 620
  bisdspb[0] = 46.3677
  bisdspb[1] = 92.7353
  bisdspb[2] = 185.4706
  bisdspb[3] = 370.9413
  bisdspb[4] = 741.8825
  bisdspb[5] = 1483.7650
  bisdspb[6] = 2967.5300
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1
```

```
la f0 choisie dans la trame du canal 2 vaut 65.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 626
  bisdspb[0] = 45.9907
  bisdspb[1] = 91.9814
  bisdspb[2] = 183.9627
  bisdspb[3] = 367.9255
  bisdspb[4] = 735.8510
  bisdspb[5] = 1471.7019
  bisdspb[6] = 2943.4038
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


commence ecriture du fichier sgrehaus.dat signal rehaussé


================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.104
la variance sur canal 1 est: 0.42051
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.121
la variance sur canal 2 est: 0.43428
le nbre de data est: 1124
le nbre de segment lu est: 4


================================================
 commence la transformee en ondelettes sur canal 1:
 la transformee en ondelette du canal 1 est faite.
 le seuil de la trfond du canal 1 est: 5.7065
 le nombre de max dans le signal 1 est: 3
 les frequences fondamentales dans la fenetre du signal 1 sont:
  f0[0]= 70.0 Hz a echantillon 108
  f0[1]= 64.5 Hz a echantillon 232
  dernposmax1= 232
```

```
commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 6.5339
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.0 Hz a echantillon 107
 f0[1]= 64.0 Hz a echantillon 232
 dernposmax2= 232


la f0 choisie dans la trame du canal 1 vaut 64.5 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 630
  bisdspb[0] = 45.6198
  bisdspb[1] = 91.2396
  bisdspb[2] = 182.4792
  bisdspb[3] = 364.9583
  bisdspb[4] = 729.9167
  bisdspb[5] = 1459.8334
  bisdspb[6] = 2919.6667
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


 la f0 choisie dans la trame du canal 2 vaut 64.0 Hz
  il y a 6 passe-bandes dans le banc de filtre
   dont l'ordre N est: 636
   bisdspb[0] = 45.2548
   bisdspb[1] = 90.5097
   bisdspb[2] = 181.0193
   bisdspb[3] = 362.0387
   bisdspb[4] = 724.0773
   bisdspb[5] = 1448.1547
   bisdspb[6] = 2896.3093
 ecriture des coefficients dans coeffiltX[] pour canal 2
  le banc de filtre pour le canal 2 est fait.
  ecriture des signaux filtrés dans sigfilt2X[] canal 2
```

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.143
la variance sur canal 1 est: 0.37654
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.164
la variance sur canal 2 est: 0.37452
le nbre de data est: 1380
le nbre de segment lu est: 5

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 5.7862
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 70.0 Hz a echantillon 162
f0[1]= 135.6 Hz a echantillon 221
f0[2]= 70.8 Hz a echantillon 334
dernposmax1= 221

commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 6.6354
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
f0[0]= 70.0 Hz a echantillon 162
f0[1]= 137.9 Hz a echantillon 220
f0[2]= 70.8 Hz a echantillon 333
dernposmax2= 220

la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 580
bisdspb[0] = 49.4975

bisdspb[1] = 98.9949
bisdspb[2] = 197.9899
bisdspb[3] = 395.9798
bisdspb[4] = 791.9596
bisdspb[5] = 1583.9192
bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 1
le banc de filtre pour le canal 1 est fait.
ecriture des signaux filtrés dans sigfilt1X[] canal 1

ecriture des signaux synthétisés du canal 1

la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 580
bisdspb[0] = 49.4975
bisdspb[1] = 98.9949
bisdspb[2] = 197.9899
bisdspb[3] = 395.9798
bisdspb[4] = 791.9596
bisdspb[5] = 1583.9192
bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.205
la variance sur canal 1 est: 0.40237
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.235
la variance sur canal 2 est: 0.35287
le nbre de data est: 1636
le nbre de segment lu est: 6

```
==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 5.2397
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 62.5 Hz a echantillon 91
 f0[1]= 72.7 Hz a echantillon 201
 f0[2]= 70.0 Hz a echantillon 341
 dernposmax1= 201


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.9998
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 62.5 Hz a echantillon 91
 f0[1]= 72.7 Hz a echantillon 201
 f0[2]= 70.0 Hz a echantillon 340
 dernposmax2= 201


la f0 choisie dans la trame du canal 1 vaut 62.5 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 650
  bisdspb[0] = 44.1942
  bisdspb[1] = 88.3884
  bisdspb[2] = 176.7767
  bisdspb[3] = 353.5534
  bisdspb[4] = 707.1068
  bisdspb[5] = 1414.2136
  bisdspb[6] = 2828.4272
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 62.5 Hz
```

il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 650
bisdspb[0] = 44.1942
bisdspb[1] = 88.3884
bisdspb[2] = 176.7767
bisdspb[3] = 353.5534
bisdspb[4] = 707.1068
bisdspb[5] = 1414.2136
bisdspb[6] = 2828.4272
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.038
la variance sur canal 1 est: 0.12430
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.045
la variance sur canal 2 est: 0.10635
le nbre de data est: 1892
le nbre de segment lu est: 7

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 3.6549
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 70.8 Hz a echantillon 211
f0[1]= 156.9 Hz a echantillon 262
f0[2]= 115.9 Hz a echantillon 331
dernposmax1= 211

commence la transformee en ondelettes sur canal 2:

```
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 4.1852
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.8 Hz a echantillon 210
 f0[1]= 153.8 Hz a echantillon 262
 f0[2]= 117.6 Hz a echantillon 330
 dernposmax2= 210


la f0 choisie dans la trame du canal 1 vaut 70.8 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 574
  bisdspb[0] = 50.0607
  bisdspb[1] = 100.1213
  bisdspb[2] = 200.2426
  bisdspb[3] = 400.4853
  bisdspb[4] = 800.9705
  bisdspb[5] = 1601.9410
  bisdspb[6] = 3203.8821
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


 la f0 choisie dans la trame du canal 2 vaut 70.8 Hz
  il y a 6 passe-bandes dans le banc de filtre
   dont l'ordre N est: 574
   bisdspb[0] = 50.0607
   bisdspb[1] = 100.1213
   bisdspb[2] = 200.2426
   bisdspb[3] = 400.4853
   bisdspb[4] = 800.9705
   bisdspb[5] = 1601.9410
   bisdspb[6] = 3203.8821
 ecriture des coefficients dans coeffiltX[] pour canal 2
  le banc de filtre pour le canal 2 est fait.
  ecriture des signaux filtrés dans sigfilt2X[] canal 2


  ecriture des signaux synthétisés du canal 2
```

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.017
la variance sur canal 1 est: 0.12966
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.021
la variance sur canal 2 est: 0.14600
le nbre de data est: 2148
le nbre de segment lu est: 8

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 3.0616
le nombre de max dans le signal 1 est: 5
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 166.7 Hz a echantillon 6
f0[1]= 103.9 Hz a echantillon 83
f0[2]= 190.5 Hz a echantillon 125
f0[3]= 70.0 Hz a echantillon 345
dernposmax1= 125

commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 3.4321
le nombre de max dans le signal 2 est: 5
les frequences fondamentales dans la fenetre du signal 2 sont:
f0[0]= 163.3 Hz a echantillon 6
f0[1]= 105.3 Hz a echantillon 82
f0[2]= 186.0 Hz a echantillon 125
f0[3]= 70.0 Hz a echantillon 345
dernposmax2= 125

la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 580

bisdspb[0] = 49.4975
bisdspb[1] = 98.9949
bisdspb[2] = 197.9899
bisdspb[3] = 395.9798
bisdspb[4] = 791.9596
bisdspb[5] = 1583.9192
bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 1
le banc de filtre pour le canal 1 est fait.
ecriture des signaux filtrés dans sigfilt1X[] canal 1

ecriture des signaux synthétisés du canal 1

la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 580
bisdspb[0] = 49.4975
bisdspb[1] = 98.9949
bisdspb[2] = 197.9899
bisdspb[3] = 395.9798
bisdspb[4] = 791.9596
bisdspb[5] = 1583.9192
bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.091
la variance sur canal 1 est: 0.15181
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.103
la variance sur canal 2 est: 0.15702
le nbre de data est: 2404

```
le nbre de segment lu est: 9


================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 4.6694
le nombre de max dans le signal 1 est: 3
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 70.0 Hz a echantillon 109
 f0[1]= 70.0 Hz a echantillon 286
 dernposmax1= 109


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.3287
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.0 Hz a echantillon 109
 f0[1]= 70.0 Hz a echantillon 286
 dernposmax2= 109


la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
  bisdspb[6] = 3167.8384
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
```

```
 dont l'ordre N est: 580
 bisdspb[0] = 49.4975
 bisdspb[1] = 98.9949
 bisdspb[2] = 197.9899
 bisdspb[3] = 395.9798
 bisdspb[4] = 791.9596
 bisdspb[5] = 1583.9192
 bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


commence ecriture du fichier sgrehaus.dat signal rehaussé


==================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.139
la variance sur canal 1 est: 0.22558
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.157
la variance sur canal 2 est: 0.25592
le nbre de data est: 2660
le nbre de segment lu est: 10


==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 7.3927
le nombre de max dans le signal 1 est: 3
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 70.0 Hz a echantillon 221
 f0[1]= 60.2 Hz a echantillon 354
 dernposmax1= 221


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 8.4663
```

```
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.0 Hz a echantillon 220
 f0[1]= 70.0 Hz a echantillon 354
 dernposmax2= 220


la f0 choisie dans la trame du canal 1 vaut 60.2 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 676
  bisdspb[0] = 42.5327
  bisdspb[1] = 85.0655
  bisdspb[2] = 170.1310
  bisdspb[3] = 340.2619
  bisdspb[4] = 680.5238
  bisdspb[5] = 1361.0476
  bisdspb[6] = 2722.0952
ecriture des coefficients dans coeffiltX[] pour canal 1
 le banc de filtre pour le canal 1 est fait.
 ecriture des signaux filtrés dans sigfilt1X[] canal 1


 ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
  bisdspb[6] = 3167.8384
 ecriture des coefficients dans coeffiltX[] pour canal 2
  le banc de filtre pour le canal 2 est fait.
  ecriture des signaux filtrés dans sigfilt2X[] canal 2


  ecriture des signaux synthétisés du canal 2


 commence ecriture du fichier sgrehaus.dat signal rehaussé
```

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.323
la variance sur canal 1 est: 0.20795
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.370
la variance sur canal 2 est: 0.19913
le nbre de data est: 2916
le nbre de segment lu est: 11

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 3.8920
le nombre de max dans le signal 1 est: 5
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 61.5 Hz a echantillon 95
f0[1]= 190.5 Hz a echantillon 137
f0[2]= 111.1 Hz a echantillon 209
f0[3]= 62.0 Hz a echantillon 338
dernposmax1= 209

commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 4.4566
le nombre de max dans le signal 2 est: 5
les frequences fondamentales dans la fenetre du signal 2 sont:
f0[0]= 61.1 Hz a echantillon 95
f0[1]= 190.5 Hz a echantillon 137
f0[2]= 112.7 Hz a echantillon 208
f0[3]= 62.0 Hz a echantillon 337
dernposmax2= 208

la f0 choisie dans la trame du canal 1 vaut 61.5 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 660
bisdspb[0] = 43.5143
bisdspb[1] = 87.0285
bisdspb[2] = 174.0571

```
 bisdspb[3] = 348.1141
 bisdspb[4] = 696.2282
 bisdspb[5] = 1392.4564
 bisdspb[6] = 2784.9128
ecriture des coefficients dans coeffiltX[] pour canal 1
 le banc de filtre pour le canal 1 est fait.
 ecriture des signaux filtrés dans sigfilt1X[] canal 1


 ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 61.1 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 666
  bisdspb[0] = 43.1821
  bisdspb[1] = 86.3642
  bisdspb[2] = 172.7284
  bisdspb[3] = 345.4568
  bisdspb[4] = 690.9135
  bisdspb[5] = 1381.8270
  bisdspb[6] = 2763.6541
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


 commence ecriture du fichier sgrehaus.dat signal rehaussé


==================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.333
la variance sur canal 1 est: 0.22911
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.383
la variance sur canal 2 est: 0.21182
le nbre de data est: 3172
le nbre de segment lu est: 12
```

```
==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 5.1722
le nombre de max dans le signal 1 est: 3
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 70.0 Hz a echantillon 91
 f0[1]= 70.0 Hz a echantillon 249
 dernposmax1= 249


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.9197
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.0 Hz a echantillon 91
 f0[1]= 70.0 Hz a echantillon 248
 dernposmax2= 248


la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
  bisdspb[6] = 3167.8384
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
```

```
 bisdspb[2] = 197.9899
 bisdspb[3] = 395.9798
 bisdspb[4] = 791.9596
 bisdspb[5] = 1583.9192
 bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


commence ecriture du fichier sgrehaus.dat signal rehaussé


==================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.247
la variance sur canal 1 est: 0.18937
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.285
la variance sur canal 2 est: 0.18931
le nbre de data est: 3428
le nbre de segment lu est: 13


==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 6.2036
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 86.0 Hz a echantillon -7
 f0[1]= 70.0 Hz a echantillon 207
 f0[2]= 70.0 Hz a echantillon 349
 dernposmax1= 207


 commence la transformee en ondelettes sur canal 2:
 la transformee en ondelette sur le canal 2 est faite.
 le seuil de la trfond du canal 2 est: 7.0515
 le nombre de max dans le signal 2 est: 4
 les frequences fondamentales dans la fenetre du signal 2 sont:
```

```
f0[0]= 86.0 Hz a echantillon -7
f0[1]= 70.0 Hz a echantillon 206
f0[2]= 70.0 Hz a echantillon 349
dernposmax2= 206


la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
  bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 1
 le banc de filtre pour le canal 1 est fait.
 ecriture des signaux filtrés dans sigfilt1X[] canal 1


 ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
  bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2


 commence ecriture du fichier sgrehaus.dat signal rehaussé
```

```
==================================================
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.029
la variance sur canal 1 est: 0.14852
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.031
la variance sur canal 2 est: 0.14596
le nbre de data est: 3684
le nbre de segment lu est: 14


==================================================
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 4.3930
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 70.0 Hz a echantillon 104
 f0[1]= 70.0 Hz a echantillon 254
 f0[2]= 148.1 Hz a echantillon 308
 dernposmax1= 254


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.0228
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 70.0 Hz a echantillon 103
 f0[1]= 70.0 Hz a echantillon 253
 f0[2]= 145.5 Hz a echantillon 308
 dernposmax2= 253


la f0 choisie dans la trame du canal 1 vaut 70.0 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.4975
  bisdspb[1] = 98.9949
  bisdspb[2] = 197.9899
  bisdspb[3] = 395.9798
  bisdspb[4] = 791.9596
  bisdspb[5] = 1583.9192
```

bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 1
le banc de filtre pour le canal 1 est fait.
ecriture des signaux filtrés dans sigfilt1X[] canal 1

ecriture des signaux synthétisés du canal 1

la f0 choisie dans la trame du canal 2 vaut 70.0 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 580
bisdspb[0] = 49.4975
bisdspb[1] = 98.9949
bisdspb[2] = 197.9899
bisdspb[3] = 395.9798
bisdspb[4] = 791.9596
bisdspb[5] = 1583.9192
bisdspb[6] = 3167.8384
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: 0.113
la variance sur canal 1 est: 0.12268
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: 0.128
la variance sur canal 2 est: 0.13202
le nbre de data est: 3940
le nbre de segment lu est: 15

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.

```
le seuil de la trfond du canal 1 est: 4.4465
le nombre de max dans le signal 1 est: 6
les frequences fondamentales dans la fenetre du signal 1 sont:
 f0[0]= 137.9 Hz a echantillon 55
 f0[1]= 250.0 Hz a echantillon 87
 f0[2]= 79.2 Hz a echantillon 188
 f0[3]= 186.0 Hz a echantillon 231
 f0[4]= 70.2 Hz a echantillon 345
 dernposmax1= 231


commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 5.0553
le nombre de max dans le signal 2 est: 6
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 140.4 Hz a echantillon 54
 f0[1]= 250.0 Hz a echantillon 86
 f0[2]= 79.2 Hz a echantillon 187
 f0[3]= 186.0 Hz a echantillon 230
 f0[4]= 69.6 Hz a echantillon 345
 dernposmax2= 230


la f0 choisie dans la trame du canal 1 vaut 70.2 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.6215
  bisdspb[1] = 99.2431
  bisdspb[2] = 198.4861
  bisdspb[3] = 396.9722
  bisdspb[4] = 793.9445
  bisdspb[5] = 1587.8889
  bisdspb[6] = 3175.7778
 ecriture des coefficients dans coeffiltX[] pour canal 1
  le banc de filtre pour le canal 1 est fait.
  ecriture des signaux filtrés dans sigfilt1X[] canal 1


  ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 69.6 Hz
 il y a 6 passe-bandes dans le banc de filtre
```

dont l'ordre N est: 584
bisdspb[0] = 49.1900
bisdspb[1] = 98.3801
bisdspb[2] = 196.7601
bisdspb[3] = 393.5203
bisdspb[4] = 787.0406
bisdspb[5] = 1574.0812
bisdspb[6] = 3148.1624
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========
le nbre n sur canal 1 est: 256
la moyenne sur canal 1 est: -0.008
la variance sur canal 1 est: 0.05849
le nbre n sur canal 2 est: 256
la moyenne sur canal 2 est: -0.008
la variance sur canal 2 est: 0.04933
le nbre de data est: 4196
le nbre de segment lu est: 16

==========
commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 2.8452
le nombre de max dans le signal 1 est: 4
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 145.5 Hz a echantillon -24
f0[1]= 69.6 Hz a echantillon 91
f0[2]= 80.8 Hz a echantillon 190
dernposmax1= 190

commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.

```
le seuil de la trfond du canal 2 est: 3.2571
le nombre de max dans le signal 2 est: 4
les frequences fondamentales dans la fenetre du signal 2 sont:
 f0[0]= 142.9 Hz a echantillon -24
 f0[1]= 70.2 Hz a echantillon 90
 f0[2]= 80.8 Hz a echantillon 189
 dernposmax2= 189


la f0 choisie dans la trame du canal 1 vaut 69.6 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 584
  bisdspb[0] = 49.1900
  bisdspb[1] = 98.3801
  bisdspb[2] = 196.7601
  bisdspb[3] = 393.5203
  bisdspb[4] = 787.0406
  bisdspb[5] = 1574.0812
  bisdspb[6] = 3148.1624
ecriture des coefficients dans coeffiltX[] pour canal 1
 le banc de filtre pour le canal 1 est fait.
 ecriture des signaux filtrés dans sigfilt1X[] canal 1


 ecriture des signaux synthétisés du canal 1


la f0 choisie dans la trame du canal 2 vaut 70.2 Hz
 il y a 6 passe-bandes dans le banc de filtre
  dont l'ordre N est: 580
  bisdspb[0] = 49.6215
  bisdspb[1] = 99.2431
  bisdspb[2] = 198.4861
  bisdspb[3] = 396.9722
  bisdspb[4] = 793.9445
  bisdspb[5] = 1587.8889
  bisdspb[6] = 3175.7778
ecriture des coefficients dans coeffiltX[] pour canal 2
 le banc de filtre pour le canal 2 est fait.
 ecriture des signaux filtrés dans sigfilt2X[] canal 2


 ecriture des signaux synthétisés du canal 2
```

commence ecriture du fichier sgrehaus.dat signal rehaussé

==========

le nbre n sur canal 1 est: 5
la moyenne sur canal 1 est: -0.024
la variance sur canal 1 est: 0.22459
le nbre n sur canal 2 est: 5
la moyenne sur canal 2 est: -0.006
la variance sur canal 2 est: 0.09954
le nbre de data est: 4201
le nbre de segment lu est: 17

==========

commence la transformee en ondelettes sur canal 1:
la transformee en ondelette du canal 1 est faite.
le seuil de la trfond du canal 1 est: 1.2396
le nombre de max dans le signal 1 est: 3
les frequences fondamentales dans la fenetre du signal 1 sont:
f0[0]= 75.5 Hz a echantillon 34
f0[1]= 102.6 Hz a echantillon 112
dernposmax1= 112

commence la transformee en ondelettes sur canal 2:
la transformee en ondelette sur le canal 2 est faite.
le seuil de la trfond du canal 2 est: 1.3887
le nombre de max dans le signal 2 est: 3
les frequences fondamentales dans la fenetre du signal 2 sont:
f0[0]= 76.9 Hz a echantillon 32
f0[1]= 101.3 Hz a echantillon 111
dernposmax2= 111

la f0 choisie dans la trame du canal 1 vaut 75.5 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 538
bisdspb[0] = 53.3665
bisdspb[1] = 106.7331
bisdspb[2] = 213.4662
bisdspb[3] = 426.9324
bisdspb[4] = 853.8647

bisdspb[5] = 1707.7295
bisdspb[6] = 3415.4590
ecriture des coefficients dans coeffiltX[] pour canal 1
le banc de filtre pour le canal 1 est fait.
ecriture des signaux filtrés dans sigfilt1X[] canal 1

ecriture des signaux synthétisés du canal 1

la f0 choisie dans la trame du canal 2 vaut 76.9 Hz
il y a 6 passe-bandes dans le banc de filtre
dont l'ordre N est: 528
bisdspb[0] = 54.3928
bisdspb[1] = 108.7857
bisdspb[2] = 217.5713
bisdspb[3] = 435.1427
bisdspb[4] = 870.2853
bisdspb[5] = 1740.5707
bisdspb[6] = 3481.1414
ecriture des coefficients dans coeffiltX[] pour canal 2
le banc de filtre pour le canal 2 est fait.
ecriture des signaux filtrés dans sigfilt2X[] canal 2

ecriture des signaux synthétisés du canal 2

commence ecriture du fichier sgrehaus.dat signal rehaussé

BIBLIOGRAPHIE

1. Université de Lausanne, *section linguistique de la faculté des Lettres*, [En ligne].http://www.unil.ch/ling/page13431.html (Page consultée le 9 décembre 2005).

2. Université de Laval, *département de langues, linguistique et traduction*, à Québec, Pierre Martin, [En ligne]. http://www.lli.ulaval.ca/labo2256/lexique/dico.html (Page consultée le 9 décembre 2005).

3. John R. Doller,jr, John G. Proakis, John H.L. Hansen. *Discrete time processing of speech signals*, MacMillan publishing company ISBN 0-02-328301-7.

4. Aapo Hyvärinen, Juha Karhunen, Erkki Oja (2001). *Independant component analysis*, John Wiley and sons, ISBN 0-471-40540-X.

5. Institut National Polytechnique de Grenoble. *Contribution à la séparation aveugle de sources*, Ali Mansour. [En ligne]. http://www.ensieta.fr/e3i2/Doc/AMansour.pdf (Page consultée le 9 décembre 2005) .

6. David Talkin (1995). *Speech coding and synthesis*, chapitre 14: a robust algorithm for pitch tracking, édition Elsevier.

7. Université de Regina, Saskatchewan. *Pitch extraction and fundamental frequency: history and current techniques*, David Gerhard, ISBN 0 7731 0455 0,[En ligne]. http://www2.cs.uregina.ca/~gerhard/publications/TRdbg-Pitch.pdf (Page consultée le 9 décembre 2005).

8. Stéphane Mallat. *A wavelet tour of signal processing*, Academic Press, ISBN 0-12-466605-1.

9. Jaideva C. Goswami, Andrew K. Chan. *Fundamentals of wavelets, theory, algorithms and applications*, , Wiley-intersciences, ISBN 0-471-19748-3.

10. A.K.Barros, T. Rutkowski, F. Itakura , N. Ohnishi. *Estimation of speech embedded in a reverberant and noisy environment by independent component analysis and wavelets.*

11. Shubha Kadambe et G.Faye Boudreaux-Bartels, (1992). *Application de la transformée en ondelettes (TO) pour la détection de la fréquence fondamentale (f0) des signaux de la parole* de IEEE.

12. Shubha Kadambe et G.Faye Boudreaux-Bartels. *A comparison of wavelet functions for pitch detection of speech signals*, CH2977-7/91/0000-0449 IEEE.

13. L. Qiu, S.N. Koh et H. Yang. *Pitch determination of noisy speech using Wavelet transform in time and frequency domains*. (IEEE TENCON'93 Pékin).

14. G.A. Shelby, C.M.Cooper, et R.R.Adhami. *A wavelet based speech pitch detector for tone languages*. 0-7803-2127-8/94 IEEE 1994.

15. Zhang Yanjie & Véronique Prinet. *IF estimation based on wavelet transform*. Institute of automation (IOA), Chinese Academy of Sciences (CAS).

16. Christian Gargour, Venkat Ramachandran (2001). *Traitement numérique des signaux*. ETS, ISBN 2-921145-24-3.

17. John G. Proakis, Dimitris G. Manolakis (1996). *Digital signal processing principles algorithms and applications*, (3e éd.). Prentice Hall, ISBN 0-13-373762-4.

18. Alan V. Oppenheim, Ronald W. Schafer (1989). *Discrete-time signal processing*. Prentice Hall, ISBN 0-13-216292-X.

19. Todd K. Moon, Wynn C. Stirling (2000). *Mathematical methods and algorithms for signal processing*. Prentice Hall, ISBN 0-201-36186-8.

20. Monson H. Hayes (1996). *Statistical digital signal processing and modeling*. John Wiley & sons. En ligne].http://www.ece.gatech.edu/users/mhayes/stat_dsp (Page consultée le 9 décembre 2005) .

21. Simon Haykin (2002). *Adaptive filter theory* (4e éd.) Prentice Hall, ISBN 0-13-090126-1.

22. Tamal Bose (2004). *Digital signal and image processing*. Wiley, ISBN 0-471-32727-1.

23. Rulph Chassaing (2002). *DSP applications using C and the TMS320C6x DSK*. Wiley, ISBN-0-471-20754-3.

24. Institut National des Sciences Appliquées de Lyon. Notes de cours sur le filtrage adaptatif, du professeur Hugues Benoit-Cattin, [En ligne]. http://www.creatis.insa-lyon.fr/~yougz/tsi/adaptatif.doc (Page consultée le 9 décembre 2005) .

25. Danilo P. Mandic & Andrzej Cichoki. *An online algorithm for blind extraction of sources with different dynamical structures*. (ICA2003).