Application de la transformée en nombres entiers à la conception d'algorithmes de faible complexité pour l'annulation d'échos acoustiques

Hamzé Alaeddine

Ecole doctorale des Sciences de la Matière de l'Information et de la Santé
Membre du Collège Doctoral de Bretagne Occidentale

THÈSE DE DOCTORAT
DE
L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

Mention " Sciences et Technologies de l'Information et de la Communication"

Spécialité " Traitement du Signal et des Images "

HAMZÉ ALAEDDINE

" Application de la transformée en nombres entiers à la conception d'algorithmes de faible complexité pour l'annulation d'échos acoustiques "

Soutenue le 12 Juillet 2007 devant le jury composé des membres ci-dessous désignés :

Président du jury

M.	Jean LE BIHAN	Professeur, ENIB-Brest

Rapporteurs

MM.	Jean-Pierre CANCES	Professeur, Université de Limoges
	Denis HAMAD	Professeur, Université de Littoral Côte d'Opale

Examinateurs

MM.	El Houssaïn BAGHIOUS	Maître de Conférences, Université de Bretagne Occidentale
	Nicolas BEUZELIN	Ingénieur DGA/GESMA, Brest
	Gilles BUREL	Professeur, Université de Bretagne Occidentale, Directeur de thèse
	Slimane KARTIT	PDG, Société Point Gamma Communications, Paris
	Samir SAOUDI	Professeur, E.N.S.T.-Bretagne

Travaux effectués au **L**aboratoire d'**E**lectronique et **S**ystèmes de **T**élécommunications
LEST-UMR CNRS 6165
Université de **B**retagne **O**ccidentale

Remerciements

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Electronique et Systèmes de Télécommunications (LEST) de l'Université de Bretagne Occidentale (UBO), au sein de l'équipe du Traitement du Signal pour les Télécommunications (TST), site de Quimper.

Je tiens à remercier sincèrement Monsieur Jean LE BIHAN, Professeur à l'Ecole Nationale d'Ingénieurs de Brest, qui m'a fait l'honneur d'accepter la présidence du jury.

Je remercie également Monsieur Jean-Pierre CANCES, Professeur à l'Université de Limoges, et Monsieur Denis HAMAD, Professeur à l'Université de Littoral Côte d'Opale, pour avoir accepté d'être rapporteur de mes travaux.

Je remercie vivement Monsieur Nicolas BEUZELIN, Chef de projet à la Délégation Générale pour l'Armement, Groupes d'Etudes Sous-Marines de l'Atlantique de Brest, Monsieur Slimane KARTIT, PDG de la Société Point Gamma Communications (Paris) et Monsieur Samir SAOUDI, Professeur à l'Ecole Nationale Supérieure des Télécommunications de Bretagne, qui m'ont fait l'honneur de participer au jury.

Je tiens à exprimer ma profonde gratitude à Monsieur Gilles Burel, Professeur à l'Université de Bretagne Occidentale, qui m'a accueilli au sein de son équipe dès mon DEA, et qui, par la suite a dirigé cette thèse. Sa grande culture scientifique et ses qualités humaines sont autant d'éléments qui m'ont conforté tout au long de l'avancement de mes travaux.

Mes plus sincères remerciements sont adressés à Monsieur El-Houssaïn BAGHIOUS, Maître de Conférences à l'Université de Bretagne Occidentale, pour ses conseils, sa bonne humeur, sa grande qualité scientifique et pédagogique, et pour avoir encadré mes premiers pas dans la recherche lors de mon stage DEA et tout au long de ma thèse.

Bien sûr, je remercie de tout coeur le petit noyau quimpérois de l'équipe du Traitement du Signal pour les Télécommunications du Laboratoire d'Electronique et Systèmes de Télécommunications qui m'a accueilli et offert un environnement agréable à mon développement tant personnel que professionnel.

Enfin, je dédie ce travail à ma mère, mes frères et soeurs, à Sofia, et à tous ceux qui m'ont soutenu et supporté durant ces trois longues années.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes et abrévitions

BLMS	Block Least Mean Square
BNLMS	Block Normalized **LMS**
BPNLMS	Block Proportionnate **NLMS**
BPNLMS + +	Block **PNLMS + +**
CCP	Cyclic Convolution Property
CRT	Chinese Residue Theorem
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
DTD	Double Talk Detector
EQM	Erreur Quadratique Moyenne
ERLE	Echo Return Loss Enhancement
FIR	Finite Impulse Response
FNT	Fermat Number Transform
GF	Galois Field
GSFFT	Generalized Sliding Fast Fourier Transform
GSFNT	Generalized Sliding Fermat Number Transform
GSNTT	Generalized Sliding Number Theoretic Transform
IFNT	Inverse **FNT**
IGSFFT	Inverse **GSFFT**
IGSFNT	Inverse **GSFNT**
IGSNTT	Inverse **GSNTT**
LMS	Least Mean Square
LPC	Linear Predictive Coding
LSF	Line Spectrum Frequency
MDF	Multi Delay Filter
NTT	Number Theoretic Transform
NLMS	Normalized **LMS**

PNLMS	Proportionnate **NLMS**
PNLMS $++$	Proportionnate **PNLMS** $++$
RBLMS	Robuste **BLMS**
RBNLMS	Robuste **BNLMS**
RBPNLMS	Robuste **BPNLMS**
RBPNLMS $++$	Robuste **BPNLMS** $++$
RLMS	Robuste **LMS**
RLS	Recursive Least Square
RNLMS	Robuste **NLMS**
RNS	Residue Number System
RPNLMS	Robuste **PNLMS**
RPNLMS $++$	Robuste **PNLMS** $++$
$\mathbf{SER_{post}}$	a Posteriori Signal to Echo Ratio
$\mathbf{SER_{prio}}$	a Priori Signal to Echo Ratio
SFFT	Sliding Fast Fourier Transform
SNR	Signal to Noise Ratio
$\mathbf{SNR_{post}}$	a Posteriori Signal to Noise Ratio
$\mathbf{SNR_{prio}}$	a Posteriori Signal to Noise Ratio
VAD	Voice Activity Detector

Notations Mathématiques

M^t	Transposé du vecteur ou de la matrice M
$E(\cdot)$	Espérance mathématique
$pgcd$	Plus Grand Commun Diviseur
$ent(a)$	Partie entière de a
a/b	Division Euclidienne de a par b
$a \mid b$	Résultat entier de la division Euclidienne de a par b
$rem(a/b)$	Reste de la division Euclidienne de a par b
$a \bmod q$ ou $\langle a \rangle_q$	Réduction de a par le modulo q
$\mathbb{Z}_q$	Ensemble des entiers appartenant à $\{0, q-1\}$
$GF(q)$	Corps de Galois d'ordre q
$\dot{a}$	Classe d'équivalence de a
$\equiv$	Congruence
F_t	$t^{\text{ème}}$ nombre de Fermat
$*$	Produit de convolution
$\bullet$	Multiplications élément par élément
$\otimes$	Produit de Kronecker

Introduction

La parole, moyen de communication privilégié entre les humains, constitue une grande partie des messages transmis en télécommunications. Un phénomène d'écho, qui est la réverbération du signal à l'émission, pose généralement un problème dans toutes les communications de type "PC à Téléphone" ou "Téléphone à Téléphone". Il peut être causé par les composantes électroniques des parties analogiques du système téléphonique, lors du passage de 4 à 2 fils du réseau téléphonique classique ($PSTN$: *Public Switched Telephone Network*), qui réfléchissent une partie du signal traité. L'interlocuteur s'entend alors parler avec un temps de retard.

Certains nouveaux services de télécommunications ont considéré le milieu acoustique (salle, cabine téléphonique, habitacle d'une voiture etc...) comme faisant partie de la chaîne de communication. Ces nouveaux services correspondent à l'apparition des postes à haut-parleur, des postes mains-libres et des systèmes de téléconférences (audio et visioconférences).

Des nouveaux problèmes sont apparus, du fait de la prise rapprochée de son dans le système téléphonique classiques par commutation ou téléphonie sur réseau IP (*Internet Protocol*). Parmi ces problèmes, nous pouvons citer :

- La réverbération où en plus du son direct, le ou les microphones de prise de son captent une multitude de réflexions qui brouillent le message transmis.

- Le bruit acoustique présent dans une salle ou dans une cabine téléphonique.

- Le phénomène d'écho acoustique : lors de la mise en place d'une communication bidirectionnelle entre deux salles, une boucle de transmission fermée est établie. Le signal émis par la salle distante est réémis vers cette même salle à cause du couplage existant entre le haut-parleur et le ou les microphones de prise de son d'une même salle. Si la transmission introduit un retard important (de l'ordre de plusieurs centaines de millisecondes), les personnes présentes dans une salle réentendent leur propre voix ; c'est le phénomène d'écho acoustique dû au canal acoustique de couplage qui, par définition, représente la transformation du signal diffusé par le haut-parleur et capté par les microphones de prise de son [Julien1982, Julien1984].

Lorsque l'écho acoustique est présent de façon gênante, c'est à dire clairement distinct subjectivement de son signal d'origine, un traitement spécifique, appelé "annulation d'écho acoustique", doit être impérativement mis en oeuvre pour préserver la qualité de la communication. Le but d'un tel traitement est d'estimer l'écho

acoustique entre le signal reçu (signal envoyé dans le haut-parleur) et la sortie de la salle (signal capté par le microphone) puis de retrancher une estimation de ce signal de sortie, ceci sans affecter le signal de parole locale dans le cas de double parole (les deux locuteurs parlent en même temps).

L'annulation d'écho acoustique est un problème d'identification d'un système linéaire (canal acoustique de couplage) excité par un signal de référence connu (parole alimentant le haut-parleur). Le problème est compliqué par le fait que le signal d'excitation est fortement non stationnaire et le canal acoustique de couplage varie au cours du temps (mouvements des personnes, déplacements d'objet, etc...). Pour tenir compte de ces problèmes, nous utilisons un annulateur d'écho acoustique où l'identification de la réponse impulsionnelle finie (FIR : *Finie Impulse Response*), représentant le canal acoustique de couplage, est réalisée par des algorithmes du type gradient stochastique (LMS : *Least Mean Squares*, $NLMS$: *Normalized LMS*, $PNLMS$: *Proportionnate NLMS*, $PNLMS++$, etc...). Malgré toutes les solutions trouvées pour le problème d'annulation d'écho acoustique, quelques difficultés restent à contourner. Parmi elles :

- La réponse impulsionnelle modélisant le canal acoustique est très longue, de quelques centaines de coefficients (téléphone main libre) à quelques milliers de coefficients (téléconférence). Ceci se traduit par une charge de calcul élevée quel que soit l'algorithme utilisé et une inadaptation aux processeurs *DSP* sur lesquels sont implantés les algorithmes.

- Les signaux d'entrée (généralement signaux de parole), ne sont pas stationnaires à l'échelle de la réponse impulsionnelle et présentent une importante variation de la puissance, cela rend difficile le problème de convergence des coefficients du filtre adaptatif.

- A ces difficultés, s'ajoute le cas de double parole (deux personnes émettent simultanément un signal). Ceci provoque une divergence des coefficients du filtre adaptatif et une dégradation irrémédiable sur le signal de la parole locale.

Le principal objectif de notre étude, réalisée au sein du Laboratoire d'Electronique et Système de Télécommunications (*LEST*), est d'évaluer la possibilité d'un développement en temps réel d'un système d'annulation d'écho acoustique et de son intégration à un processeur *DSP* à virgule fixe de faible complexité. Notre étude a donc été menée pour améliorer les performances de ce système et de diminuer sa complexité de calcul au maximum, tout en maintenant un équilibre entre son coût d'implantation et sa qualité. Précisons que le choix d'un *DSP* à virgule fixe provient du fait que son coût de calcul et sa consommation sont grandement réduits par rapport à un *DSP* à virgule flottante. Cependant, un inconvénient majeur de ce type de processeur *DSP* provient de la gestion des formats de données qui pose toujours quelques difficultés pour l'implantation d'algorithmes.

Ces algorithmes utilisés dans le système d'annulation d'écho acoustique représentent une charge de calcul trop importante qu'il faut chercher à réduire. Notre but est alors de développer de nouveaux algorithmes plus efficaces, appréciés en terme de compromis entre la qualité du système d'annulation d'écho acoustique et la

complexité de calcul engendré et qui ne modifient pas le traitement à réaliser. Pour cela, une étude théorique a porté sur les algorithmes d'adaptation utilisés dans le système d'annulation d'écho acoustique pour nous fournir une bonne connaissance du système et nous permettre de proposer quelques améliorations entraînant une réduction satisfaisante du délai de traitement.

L'étude principale, présentée dans ce rapport, consiste à proposer et implanter de nouveaux algorithmes performants ayant une complexité de calcul réduite et un délai algorithmique satisfaisant. Les algorithmes d'adaptation du système d'annulation d'écho acoustique ont ainsi été étudiés et modifiés pour permettre leur implantation, dans un premier temps, sur un processeur à virgule flottante en mettant en oeuvre la transformée de Fourier rapide (FFT : $Fast\ Fourier\ Transform$) et, dans un deuxième temps, sur un processeur à virgule fixe en mettant en oeuvre la transformée en nombres de Fermat (FNT : $Fermat\ Number\ Transform$) [Julien1991]. La transformée en nombres de Fermat (FNT) est une transformée en nombres entiers (NTT : $Number\ Theoretic\ Transform$) particulière, définie sur un corps de Galois d'ordre égal à un nombre de Fermat. Elle permet une réduction du nombre de multiplications nécessaires à la réalisation de certaines fonctions utilisées en traitement du signal telles que les produits de convolution. C'est dans cette optique qu'une nouvelle technique de traitement des algorithmes du filtrage adaptatif a été étudiée. Cette technique consiste à traiter les signaux par blocs d'échantillons au lieu de les traiter échantillon par échantillon pour mettre en évidence le calcul des produits de convolution.

La réduction du nombre de multiplications requises par l'implantation des algorithmes du filtrage adaptatif par la FNT, comparée à une implantation en FFT, est accompagnée par une augmentation du nombre d'opérations telles que les additions, les soustractions et les décalages de bits. Pour réduire au maximum cette augmentation, une nouvelle technique de transformation a été proposé, par analogie à la technique du Sliding Généralisé appliquée à la FFT ($GSFFT$: *Generalized Sliding Fast Fourier Transform*) [Gazor1992]. Cette technique, intitulée Sliding Généralisé FNT ($GSFNT$: *Generalized Sliding Fermat Number Transform*), est définie sur un corps de Galois. Elle a pour objet de réduire la complexité de calcul nécessaire à la mise en oeuvre d'une transformée en nombres de Fermat (FNT) en appliquant celle-ci à une succession de séquences qui diffèrent d'un certain nombre d'échantillons l'une de l'autre.

Ce rapport de thèse comprend six chapitres.

Le premier chapitre rappelle le contexte de notre étude et présente rapidement l'origine de l'écho électrique et acoustique dans les télécommunications. Les généralités sur des différentes techniques d'annulation d'écho y sont présentées. Il rappelle aussi la théorie du filtrage adaptatif, décrit et compare les différents algorithmes d'adaptations, des plus classiques aux plus récents : algorithmes du type de gradient déterministe, algorithme des moindres carrés récursif (RLS) et algorithmes du gradient stochastique (*LMS, NLMS, PNLMS, PNLMS++*).

Le deuxième chapitre est une introduction aux transformées en nombres entiers (NTT) . Nous y présentons

la définition et les propriétés fondamentales d'une transformée en nombres entiers ainsi que les différentes applications au traitement de signal. Le domaine de définition d'une NTT n'étant que des ensembles de nombres entiers, son utilisation peut favoriser les implantations d'algorithmes sur des processeurs DSP à virgule fixe. Cette transformée, connue depuis une trentaine d'années mais peu répandue en traitement du signal, présente de nombreux avantages par rapport à d'autres outils mathématiques plus populaires comme la transformée de Fourier. En effet, le passage des données dans le domaine complexe est évité, ce qui allège les calculs et supprime toute erreur d'arrondi.

Le troisième chapitre décrit la méthode de traitement des algorithmes du filtrage adaptatif (*LMS*, *NLMS*, *PNLMS*, et *PNLMS++*) par blocs d'échantillons et souligne l'intérêt que présente l'utilisation de cette méthode en traitement du signal. L'étude porte plus particulièrement sur le traitement par blocs de l'algorithme *PNLMS++* ($BPNLMS++$) qui est largement développé. Une implantation, au moyen de la FNT, de ces différents algorithmes, traités par blocs, permet de comparer leurs performances en terme de convergence et de degré de complexité des calculs.

Le quatrième chapitre présente une technique de calcul de la transformée de Fourier rapide qui consiste à traiter un ensemble de séquences dans lequel deux séquences successives diffèrent de N échantillons l'une de l'autre. Cette technique intitulée Sliding Généralisé appliquée à la FFT ($GSFFT$) présente l'intérêt de réduire la complexité de calcul d'une transformée de Fourier rapide.

Pour réduire encore plus cette complexité de calcul, nous proposons d'étudier, par analogie avec la $GSFFT$, une nouvelle technique de calcul de la transformée en nombres entiers. Nous utilisons ensuite cette nouvelle technique, intitulée $GSFNT$, pour développer et traiter par blocs l'algorithme $BPNLMS++$ avant de procéder à son implantation.

Le cinquième chapitre propose des solutions aux problèmes posés par la divergence des algorithmes du filtrage adaptatif dans le cas de double parole. Une de ces solution consiste à traiter l'annulation d'écho par un système combiné [Martin1995] constitué d'un annulateur d'écho et d'un système de variation de gain. L'ensemble de ce système s'appuie sur l'information fournie par un détecteur de double parole afin de décider d'arrêter le système d'adaptation en cas de double parole. Cependant, ce système de détection, formé par deux *VAD* (VAD : *Voice Activity Detector*) peut fournir une mauvaise décision sur la présence de double parole. Dans ce cas, l'adaptation continuera au lieu d'être arrêtée et provoque ainsi la divergence du système. Pour éviter ce problème de divergence nous proposons d'adapter les coefficients du filtre adaptatif par un algorithme plus robuste, $PNLMS++$ robuste ($RPNLMS++$), que nous développons et traitons par blocs. Une implantation au moyen de la $GSFNT$ de différents algorithmes robustes proposés ($BRNLMS$, $BRPNLMS$, $BRPNLMS++$) permet de comparer leurs performances en terme de convergence par rapport aux algorithmes non robustes ($BNLMS$, $BPNLMS$, $BPNLMS++$).

Le sixième chapitre rappelle, dans un premier temps, l'algorithme de Filtre à Délais Multiples (MDF :

Multi-Delay Filter) qui consiste à décomposer le filtre adaptatif en sous-blocs de moindre taille. Nous proposons ensuite une nouvelle structure de cet algorithme dans l'objectif d'améliorer son implantation en temps réel. Cette nouvelle structure, intitulée $MDFP$, adaptée au système d'annulation d'écho, permet de réduire la complexité du traitement à réaliser. L'accent est ensuite mis sur le calcul d'une nouvelle méthode de gestion du pas d'adaptation, ce qui permet d'améliorer les capacités de convergence du filtre adaptatif même en présence des grandes variations de la puissance du signal d'entrée.

Nous utilisons ensuite cette nouvelle méthode de gestion du pas d'adaptation et la nouvelle procédure $MDFP$ du MDF pour développer et implanter en FNT l'algorithme $BPNLMS++$.

Enfin, une dernière partie conclut ce travail de thèse et expose quelques perspectives à ces travaux et les thèmes, qu'il reste encore à aborder avant l'obtention d'un système d'annulation d'écho acoustique complet.

Chapitre 1

Annulation d'écho acoustique

1.1 L'écho dans le système des télécommunications

L'écho, qu'il soit électrique ou acoustique, a des origines différentes qu'il est nécessaire de clarifier et de souligner. Quelles que soient ces origines, l'écho peut être caractérisé par la réponse impulsionnelle associé à son trajet entre le point d'émission et d'observation. Cette réponse est très sensible et peut varier rapidement d'un instant à un autre en fonction du type d'écho.

1.1.1 L'écho électrique

Le réseau de télécommunications longue distance, internationale par exemple, est constitué de deux types de liaison téléphonique, liaison téléphonique "à deux fils" et liaison téléphonique "à quatre fils". Le passage d'une liaison deux fils à une liaison quatre fils a pour effet de générer un écho dit électrique. Ceci est dû à la désadaptation entre l'impédance présentée par le combiné téléphonique et l'impédance du canal de transmission. Nous retiendrons en outre que le problème de l'écho électrique est classiquement résolu à l'aide d'un annulateur d'écho utilisant des algorithmes du type gradient stochastique plus spécifiques.

1.1.2 L'écho acoustique

Nous avons vu dans le paragraphe précédent que l'écho électrique est provoqué par le réseau de télécommunications, ce qui n'est pas le cas pour l'écho acoustique. L'origine de l'écho acoustique provient de l'utilisation de nouveaux systèmes de télécommunications dits "mains libres". Au début des télécommunications, l'utilisateur était obligé de coller son oreille à un combiné pour entendre son interlocuteur distant. Aujourd'hui, les nouveaux systèmes de télécommunications permettent la liberté de mouvement du locuteur en restituant le son de l'interlocuteur sur un haut-parleur (figure 1.1). Le couplage acoustique, généré par

l'utilisation de tels systèmes de télécommunications mains libres, provoque certains effets indésirables comme le phénomène de l'écho acoustique ou encore l'instabilité de la boucle de communication [Gilloire1987].

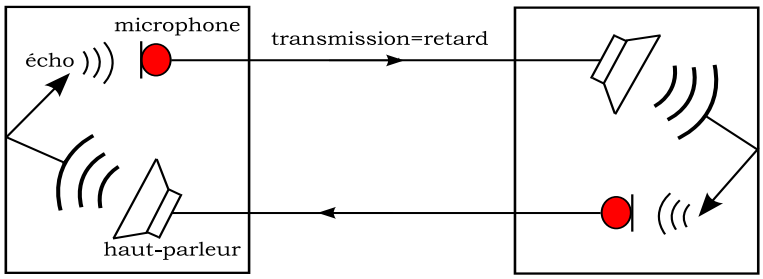

FIG. 1.1 – Exemple de communication bi-directionnelle avec écho acoustique

L'écho acoustique est provoqué par la transmission du signal émis par le haut-parleur et reçu par le microphone : cette transmission est composée d'un trajet direct et de multiples réflexions captées par le microphone, et a pour conséquence de renvoyer vers le locuteur qui a prononcé la parole dans une salle distante son propre signal. C'est donc la propagation acoustique d'une onde sonore à l'intérieur d'un volume donné qui provoque l'écho acoustique.

Le phénomène de l'écho acoustique présente des complexités du fait que ses propriétés acoustiques sont très variables en fonction de l'environnement correspondant. Il suffit de s'intéresser à quelques exemples d'utilisation de systèmes mains libres pour apprécier la difficulté du problème. Les réseaux de télécommunications actuels supportent des débits d'information considérables et autorisent la transmission simultanée et en temps réel d'images et de son. Cette avancée technologique permet par conséquent l'organisation de téléréunions entre locuteurs de sites distants en leur offrant une sensation de présence et de naturel. Les exemples les plus standards d'applications de téléréunions sont par exemple, la téléconférence et la visioconférence sur PC. Pour la téléconférence, une salle spécialement conçue pour cette application est généralement utilisée.

Lorsqu'un son est émis à l'intérieur d'une salle (ou d'une voiture), il subit des transformations physiques qui peuvent être comprises grâce aux principes de l'acoustique des salles. Des interprétations théoriques précises peuvent être obtenues en faisant appel aux domaines de l'acoustique géométrique, ondulatoire et statistique [Kuttruff1991]. Néanmoins, le phénomène physique peut être décrit simplement et succinctement comme suit. Une onde sonore, émise par un émetteur, se propage suivant les lois de l'acoustique vers un récepteur. Au cours de son trajet, l'onde subit l'influence de l'environnement acoustique dans lequel elle se propage. Le phénomène se résume, dans le cas d'une propagation dans un espace libre, à l'absorption d'une onde sonore par l'air qui dépend de paramètres (température, pression atmosphérique, etc...) variant lentement dans le temps par rapport à l'échelle de stationnarité du signal sonore. S'ajoutent à cela des phénomènes de réflexion, diffraction, diffusion, et absorption provoqués par les parois et obstacles présents dans l'espace clos. Le trajet de propagation d'une onde sonore est appelé canal acoustique. Le canal acoustique

est entièrement défini par le milieu de propagation (en l'occurrence l'air), l'espace clos (sa géométrie, ses propriétés acoustiques), la source (dans le cas de l'écho, le haut-parleur), et le récepteur (dans le cas de l'écho, le microphone). Les caractéristiques de la réponse impulsionnelle associée au trajet d'écho acoustique dépendent directement de ces différents paramètres.

La réponse impulsionnelle d'un canal acoustique se présente sous la forme d'une onde directe et d'une succession d'ondes réfléchies par les parois d'une salle particulière. Un exemple d'une réponse impulsionnelle mesurée dans une salle est représentée par la figure (1.2).

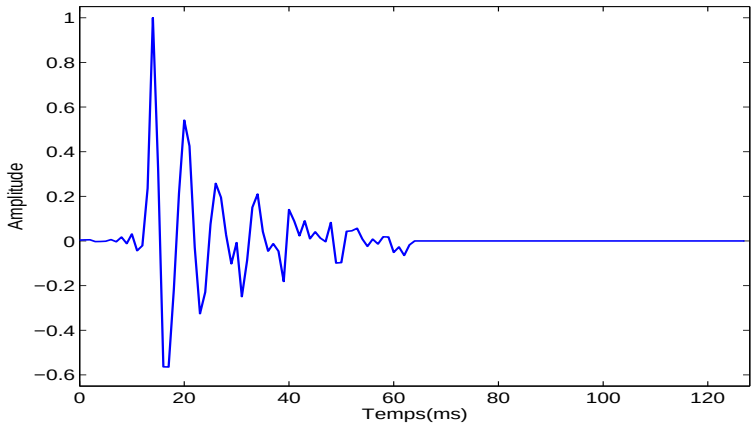

FIG. 1.2 – Exemple de réponse impulsionelle du canal acoustique

Les ondes se propagent en trajet direct (le trajet le plus court emprunté par l'onde sonore) jusqu'à ce qu'elles rencontrent un obstacle sur lequel elles se réfléchissent tout en perdant de l'énergie. En réalité, la réponse impulsionnelle de couplage acoustique est de durée infinie mais il est généralement admis que son support temporel significatif est de l'ordre de 50 à 100 ms dans une voiture, et de 250 ms à 300 ms dans une salle de téléconférence.

L'écho acoustique, résultant du couplage acoustique entre un haut-parleur et un microphone, peut donc être caractérisé par la réponse impulsionnelle du canal acoustique correspondant. Cette réponse impulsionnelle qui est très sensible et dépendante de son environnement acoustique, peut varier rapidement d'un instant à un autre, puisque la taille de la salle, le revêtement des murs, la présence d'objets ou de personnes dans la salle, etc..., sont autant de paramètres qui influent sur la nature du couplage acoustique et modifient cette réponse.

1.2 La gêne provoquée par l'écho

Avant de présenter les différentes techniques de traitement de l'écho acoustique, il est important de comprendre sous quelles conditions l'écho est perçu comme une perturbation gênante.

L'écho acoustique est présent de façon gênante pour un délai de transmission global de 30 ms. Ce délai

est largement dépassé que ce soit dans le cadre de la radiotéléphonie mains libres où le retard de transmission est de l'ordre de 180 ms ou dans des contextes de téléconférence où le traitement et la transmission de la voix introduisent un retard supérieur. Dans ce cas, un traitement spécifique doit être impérativement mis en oeuvre pour préserver la qualité de la communication.

Dans le cas de simple parole et pour des retards inférieurs à $25ms$, un système d'annulation d'écho doit fournir une atténuation de l'écho de l'ordre de 24 dB. Ce même système doit être capable de fournir une atténuation de l'écho de 40 dB pour des retards excédant 25 ms [Gilloire1994, IUTG131].

1.3 Les traitements classiques de l'écho

Dans ce paragraphe, nous proposons de décrire le concept du système d'annulation d'écho acoustique et de présenter les principales méthodes algorithmiques existantes. Nous nous intéressons plus spécifiquement à la résolution du problème posé par l'écho acoustique, résolution basée sur des techniques de filtrage adaptatif.

Les algorithmes de filtrage adaptatif sont très nombreux et ont été largement étudiés dans la littérature. Nous en rappelons les principaux en les classifiant par famille, même s'il a été démontré que tous les algorithmes adaptatifs sont liés entre eux et peuvent se déduire les uns des autres au moyen d'approximations [Benesty1991]. Nous présentons tout d'abord les algorithmes qui ont certainement suscité le plus de travaux, à savoir le LMS, RLS, $NLMS$, $PNLMS$, et $PNLMS++$. Intérêt justifié par le fait que le premier est simple à mettre en oeuvre et le dernier a une vitesse de convergence optimale.

1.3.1 Introduction au principe de l'annulation d'écho acoustique

L'annulation d'écho acoustique est un des axes d'évolution récents les plus importants dans le domaine des communications. Il s'agit d'une application des techniques d'identification des systèmes, le système à étudier étant le chemin d'écho acoustique. En télécommunications, il est souvent nécessaire d'éliminer des échos gênants. C'est notamment le cas pour la transmission de données en mode bidirectionnel simultané sur deux fils ou pour la transmission téléphonique par satellite.

Les échos proviennent de réflexions des signaux électriques. Les réflexions acoustiques peuvent aussi être gênants dans les terminaux téléphoniques à mains-libres pour les salles d'audio ou vidéoconférence [Benesty2001, Gay1998].

Une solution simple et commune vient immédiatement à l'esprit. En effet, l'utilisation de filtres adaptatifs pour estimer l'écho est devenue courante dans les applications de téléphonie. Ce procédé (figure 1.3) permet la construction d'une image de l'écho qui sera soustraite au signal reçu [Tourneur1991, Ye1991, Prado1994, Ferrara1980, Clark1981].

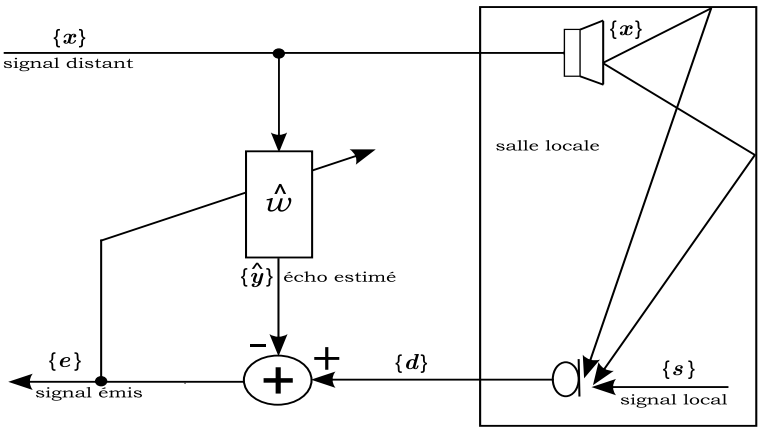

FIG. 1.3 – Principe de l'annulation d'écho acoustique

Un exemple simple est donné dans le cas d'une application téléphonique ou de téléconférence dans laquelle le signal éloigné $\{x\}$ provient d'un haut parleur. Le microphone reçoit un signal $\{d\}$ comportant un écho $\{y\}$ du signal $\{x\}$ et un signal proche $\{s\}$. Cet écho peut être modélisé comme le passage du signal $\{x\}$ à travers la fonction de transfert du local dans lequel se trouvent le haut parleur et le micro (figure 1.3). L'écho bruité est renvoyé par le microphone. Le chemin d'écho est la matérialisation de toutes les réflexions subies par le signal éloigné avant d'atteindre le microphone.

Les dispositifs d'annulation d'écho mettent en oeuvre des filtres adaptatifs dont l'adaptation du très grand nombre de coefficients est généralement réalisée grâce à des algorithmes du gradient stochastique. La rapidité et la précision de cette adaptation sont des points très critiques. En effet, les couplages acoustiques dans une salle sont caractérisés par un spectre rapidement évolutif avec de nombreux creux très prononcés. De plus, les signaux comme la parole ou le son sont fortement non stationnaires, ce qui rend le problème d'annulation d'écho plus difficile à résoudre. L'adaptation du filtre adaptatif, $\hat{w}$, est classiquement réalisée en l'absence de parole locale (parole du locuteur présent dans la salle locale). Lorsqu'une occurrence de double parole (présence à la fois de parole locale et de parole lointaine) est détectée, l'adaptation du filtre est stoppée. Le bon fonctionnement de l'annulateur d'écho repose donc implicitement sur l'emploi d'un détecteur d'activité vocale, contrôlant à la fois réception et émission. La structure typique d'un contrôle de l'écho acoustique est présentée sur la figure (1.4). En plus du filtrage réalisé par l'annulateur d'écho, un bloc de détecteur d'activité de voix locale est ajouté pour agir en cas de double parole ou de parole locale seule.

En cas de parole locale seule, le gain en réception est diminué pour se prémunir d'un retour d'écho dans la salle locale. En cas de double parole, le détecteur d'activité vocale permet de déterminer le sens de transmission à favoriser, ce qui se traduit par une augmentation du gain en réception et une diminution du gain en émission si le locuteur distant est favorisé et inversement si c'est le locuteur local qui est favorisé.

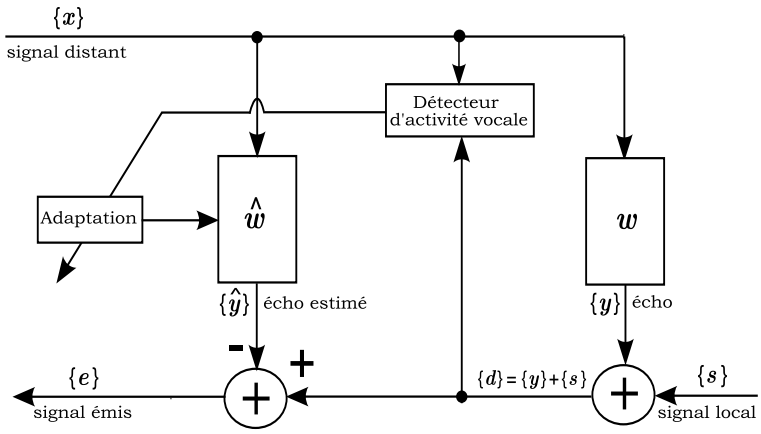

FIG. 1.4 – Structure complète d'un système d'annulation d'écho

1.4 Filtrage Adaptatif

1.4.1 Généralités sur le fitrage adaptatif

Le principe du filtrage adaptatif appliqué à l'identification d'un système inconnu, schématisé sur la figure (1.5), repose classiquement sur l'hypothèse que le système inconnu est modélisé exactement par un filtre à Réponse Impulsionnelle Finie (FIR : *Finie Impulse Response*) de L coefficients, $w_k = [w(0)\ w(1)\ \ldots\ w(L-1)]^T$. Le filtre adaptatif $\hat{w}$, modélisé par un filtre *FIR* de taille idéalement égale à celle du système inconnu, a pour rôle de fournir une estimée $\{\hat{y}\}$ du signal inconnu $\{y\}$, l'estimée étant obtenue en réalisant une opération de filtrage du signal $\{x\}$ par le système $\hat{w}$.

La technique de filtrage adaptatif se décompose classiquement en deux étapes :

1. Une étape de filtrage qui permet d'obtenir une estimée du signal inconnu en convoluant le signal d'entrée $\{x\}$ avec les coefficients du filtre adaptatif $\hat{w}$. L'erreur d'estimation $\{e\} = \{d\} - \{\hat{y}\}$ est ensuite utilisée dans la partie adaptation pour mettre à jour les coefficients du filtre.

2. Une étape d'adaptation qui permet d'ajuster les coefficients du filtre adaptatif $\hat{w}$ suivant un algorithme donné.

L'algorithme de filtrage adaptatif permet de calculer les coefficients du filtre $\hat{w}$ de façon à ce que la différence entre le signal $\{d\}$ et l'actuelle sortie du filtre $\{\hat{y}\}$ soit minimisée au sens d'un critère statistique préalablement défini. De manière générale, l'algorithme d'adaptation se présente sous la forme vectorielle suivante : [Farhang1998, Pieter1974, Emmanuel1993, Westall1993, Saeed1996, Sen1997]

$$\begin{pmatrix}\textbf{vecteur des}\\ \textbf{nouveaux}\\ \textbf{coefficients}\\ \textbf{du filtre}\end{pmatrix} = \begin{pmatrix}\textbf{vecteur des}\\ \textbf{anciens}\\ \textbf{coefficients}\\ \textbf{du filtre}\end{pmatrix} + (\textbf{pas d'adaptation}) . \begin{pmatrix}\textbf{échantillon du}\\ \textbf{signal d'erreur}\end{pmatrix} . \begin{pmatrix}\textbf{vecteur}\\ \textbf{du signal}\\ \textbf{d'entrée}\end{pmatrix}$$

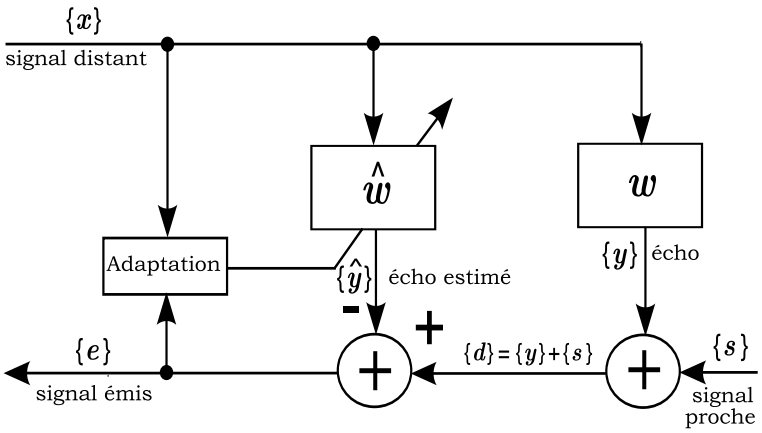

FIG. 1.5 – Schématisation de l'identification par filtrage adaptatif

1.4.2 Du filtrage de Wiener au filtrage adaptatif

De manière générale, les filtres sont utilisés dans des applications où on connaît la bande de fréquence utile ainsi que la fréquence principale. Ces filtres servent à améliorer le rapport signal sur bruit sous l'hypothèse où la bande de fréquence du bruit est supérieure à celle du signal. Dans ce cas, un filtre passe-bande centré sur la fréquence principale du signal permettra d'extraire le signal.

Un filtre adaptatif FIR est un filtre dont les coefficients sont modifiés en fonction des signaux extérieurs (signal lointain $\{x\}$ par exemple). Ce filtre permettra, à l'aide d'un algorithme du type des moindres carrés, une modélisation progressive de la réponse impulsionnelle w du chemin d'écho [Benesty2001].

Au début des années 1940, et dans le cadre de très gros efforts de recherche militaires menés au MIT (MIT : $Massachusetts\ Institue\ of\ Technology$), N. Wiener s'intéressa au problème de l'estimation d'un signal à partir d'observations bruitées d'un signal corrélé avec le signal estimé [Haykin1986]. Le filtre de Wiener développé à cette occasion permet de construire une estimation $\{\hat{y}\}$ de $\{y\}$ à partir du signal $\{x\}$. En particulier, le filtre de Wiener qui est développé à partir de concepts temporels et non fréquentiels est conçu pour minimiser l'erreur quadratique moyenne entre sa sortie $\{\hat{y}\}$ et une sortie désirée $\{d\}$, comme le montre la figure (1.6) [Kurt1976, Stephen1984, David1984].

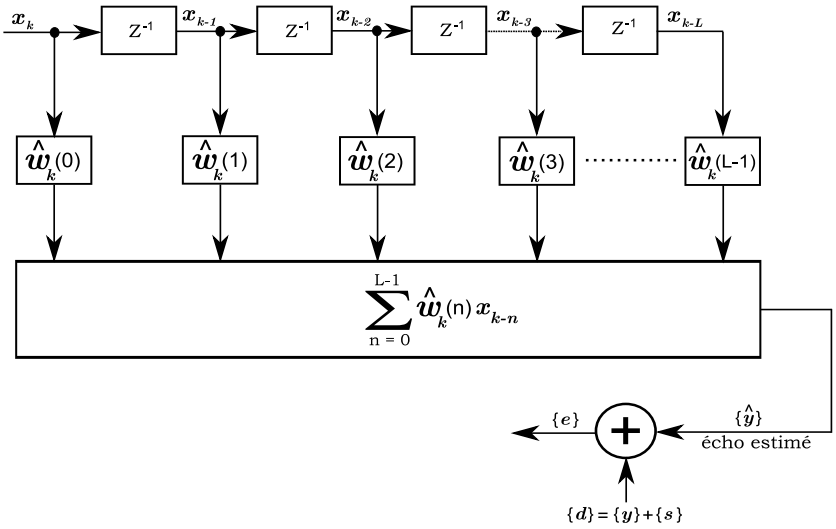

FIG. 1.6 – Schéma général d'un système d'estimation d'erreur

Plus $\{e\}$ est faible, plus l'estimation sera bonne. On cherche donc un filtre qui minimisera l'erreur quadratique moyenne. Il est pratique de minimiser $\left\{e^2\right\}$ car c'est une fonction quadratique facilement dérivable. Par ailleurs, étant donné que les signaux qui nous intéressent sont aléatoires, la fonction coût qui permettra de minimiser l'erreur quadratique moyenne (EQM) est définie par : [Feur1985]

$$\xi_k = E\left(e_k^2\right) \tag{1.1}$$

avec

$$e_k = d_k - \hat{y}_k \tag{1.2}$$

où k est l'indice d'itération.

On se limitera ici au calcul des filtres FIR. Appelons w, le filtre que nous recherchons et $\hat{w}_k$ la réponse impulsionnelle estimée donnée en notation matricielle par :

$$\hat{w}_k = \left[\begin{array}{ccccc} \hat{w}\,(0) & \hat{w}\,(1) & \hat{w}\,(2) & \ldots & \hat{w}\,(L-1) \end{array} \right]^T \tag{1.3}$$

où L désigne la longueur de ce filtre.

Le filtrage du signal x par les coefficients $\hat{w}_k$ du filtre permet ainsi l'obtention d'une estimation de l'écho $\hat{y}_k$: [Farhang1998, Shankar1983, Kazuo1990]

$$\hat{y}_k = \sum_{n=0}^{L-1} \hat{w}_k\,(n)\, x_{k-n} \tag{1.4}$$

L'estimation de cet écho $\hat{y}_k$ peut encore s'écrire, en introduisant la notation matricielle :

$$\begin{aligned} \hat{y}_k &= \hat{w}_k^T \chi_k \\ \hat{y}_k &= \chi_k^T \hat{w}_k \end{aligned} \tag{1.5}$$

avec

$$\chi_k = \begin{bmatrix} x_k & x_{k-1} & x_{k-2} & \dots & x_{(k-(L-1))} \end{bmatrix}^T \tag{1.6}$$

En faisant l'hypothèse que le signal $\{x\}$ est stationnaire, et si on introduit l'équation (1.5) dans l'équation (1.1), on peut écrire la fonction coût de la manière suivante :

$$\begin{aligned} \xi_k &= E\left[\left(d_k - \hat{w}_k^T \chi_k\right)^2\right] \\ \xi_k &= E\left[d_k^2 - 2\hat{w}_k^T \chi_k d_k + \hat{w}_k^T \chi_k \chi_k^T \hat{w}_k\right] \\ \xi_k &= E\left[d_k^2\right] - 2\hat{w}_k^T E\left[\chi_k d_k\right] + \hat{w}_k^T E\left[\chi_k \chi_k^T\right] \hat{w}_k \\ \xi_k &= E\left[d_k^2\right] - 2\hat{w}_k^T \Phi_{xd} + \hat{w}_k^T \Phi_{xx} \hat{w}_k \end{aligned} \tag{1.7}$$

où Φ_{xx} est une matrice d'autocorrélation de taille $L \times L$ définie par :

$$\Phi_{xx} = E\left[\chi_k \chi_k^T\right] \tag{1.8}$$

et où Φ_{xd} est un vecteur d'intercorrélation de taille L définie par :

$$\Phi_{xd} = E\left[\chi_k d_k\right] \tag{1.9}$$

Les équations (1.1) et (1.2) montrent que pour un filtre FIR, la fonction coût EQM dépend de la réponse impulsionnelle $\hat{w}_k$. Le minimum de la fonction coût est obtenu en cherchant les conditions d'annulation de sa dérivée par rapport aux variables que sont les L points de la réponse impulsionnelle du filtre.

Introduisons l'opérateur gradient $\bigtriangledown$:

$$\bigtriangledown \xi_k = \begin{bmatrix} \frac{\partial \xi_k}{\partial \hat{w}_k(0)} & \frac{\partial \xi_k}{\partial \hat{w}_k(1)} & \dots & \frac{\partial \xi_k}{\partial \hat{w}_k(L-1)} \end{bmatrix}^T \tag{1.10}$$

On a alors (en dérivant l'équation 1.7) :

$$\nabla \xi_k = -2\Phi_{xd} + 2\Phi_{xx} \hat{w}_k \tag{1.11}$$

Le gradient s'annule pour une réponse impulsionnelle optimale $\hat{w}_{opt}$ définie par : [Thomas1974, Thomas1981]

$$\Phi_{xx}\hat{w}_{opt} = \Phi_{xd} \tag{1.12}$$

Le filtre ainsi défini est appelé filtre FIR de Wiener optimal [Bernard1975, Bernard1976, Simon1999, George1999]. Il permet d'obtenir une erreur quadratique minimale entre d_k et $\hat{y}_k$ donnée par :

$$\xi_{min} = E\left[d_k^2\right] - \hat{w}_{opt}^T \Phi_{xd} \tag{1.13}$$

La mise en oeuvre d'un filtre optimal de Wiener demande la connaissance des caractéristiques du signal, 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 stratégie d'annulation d'écho acoustique repose sur une estimation du canal acoustique de l'écho en identifiant la réponse impulsionnelle du bouclage acoustique qui est retranchée du signal émis. La réponse varie au cours du temps et impose un filtrage adaptatif. Le filtrage adaptatif a pour objet d'approcher ces filtres optimaux. Pour cela, nous utiliserons des algorithmes d'optimisation.

1.5 Algorithmes de base du filtrage adaptatif

1.5.1 Introduction

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 (1.7).

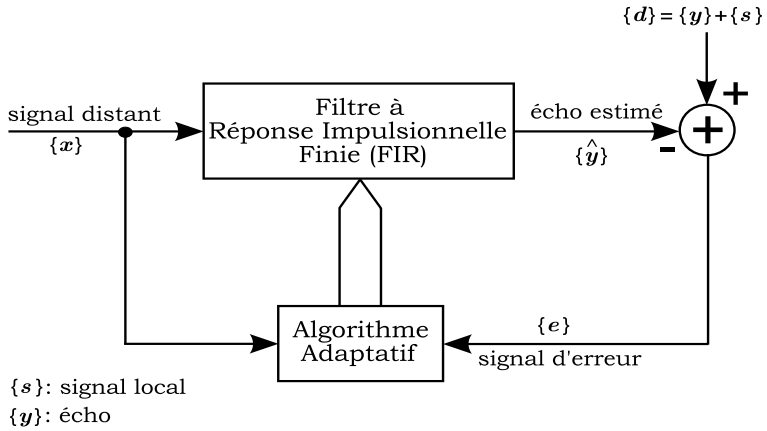

FIG. 1.7 – Schéma général d'un système de filtrage adaptatif

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.

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 principal rôle d'un algorithme adaptatif est d'ajuster un vecteur de paramètres (ici $\hat{w}_k$) pour un objectif bien défini (minimisation de $l'EQM$).

L'objectif est l'obtention d'algorithmes efficaces pour l'annulation d'écho acoustique. La famille des algorithmes des moindres carrés récursifs est la plus utilisée.

1.5.2 Algorithmes du gradient déterministe

Le filtre de Wiener donné par l'équation (1.12) et qui permet de calculer le filtre de Wiener optimal, $\hat{w}_{opt}$, conduit à résoudre un système de L équations à L inconnues :

$$\Phi_{xx}\hat{w}_{opt} = \Phi_{xd}$$

Il peut être préférable de résoudre ce système par une méthode itérative, notamment en se souvenant que la fonction de coût est quadratique, ce qui entraîne que le minimum est unique.

La méthode du gradient consiste à :

- Choisir un vecteur nul $\hat{w}_0$.

- Obtenir, à partir d'un vecteur $\hat{w}_k$ donné, un vecteur $\hat{w}_{k+1}$ par incrémentation de $\hat{w}_k$ dans la direction opposée du gradient de la fonction coût ξ.

$$\hat{w}_{k+1} = \hat{w}_k - \frac{1}{2}\mu \bigtriangledown \xi \mid_{\hat{w}=\hat{w}_k} \tag{1.14}$$

où μ est le pas d'adaptation de l'algorithme gradient qui contrôle la convergence du filtre adaptatif et

$$\bigtriangledown\xi \mid_{\hat{w}=\hat{w}_k} = -2\Phi_{xd} + 2\Phi_{xx}\hat{w}_k \tag{1.15}$$

La forme explicite de l'algorithme du gradient déterministe est donc :

$$\hat{w}_{k+1} = \hat{w}_k + \mu\left(\Phi_{xd} - \Phi_{xx}\hat{w}_k\right) \tag{1.16}$$

Le paragraphe suivant porte sur l'étude de la convergence de l'algorithme du gradient déterministe, c'est à dire sur son aptitude à être stable et à tendre vers la solution optimale $\hat{w}_{opt}$.

1.5.2.1 Stabilité de l'algorithme du gradient déterministe

La stabilité de la récurrence imposée par l'équation (1.16), contrôlée par le pas d'adaptation μ, dépend également de la matrice d'autocorrélation du vecteur d'entrée χ_k. Pour s'en convaincre, il suffit d'étudier la convergence du vecteur d'écart entre les coefficients du filtre et la solution optimale, défini par : [Haykin1986]

$$\upsilon_k = \hat{w}_k - \hat{w}_{opt} \tag{1.17}$$

En utilisant le fait que $\Phi_{xx}\hat{w}_{opt} = \Phi_{xd}$, la récurrence donnée par l'équation (1.16) devient :

$$\hat{w}_{k+1} = \hat{w}_k + \mu\left(\Phi_{xx}\hat{w}_{opt} - \Phi_{xx}\hat{w}_k\right)$$

$$\hat{w}_{k+1} = \hat{w}_k + \mu\Phi_{xx}\left(\hat{w}_{opt} - \hat{w}_k\right) \tag{1.18}$$

d'où

$$\begin{aligned} \upsilon_{k+1} &= \hat{w}_{k+1} - \hat{w}_{opt} \\ &= \hat{w}_k + \mu\Phi_{xx}\left(\hat{w}_{opt} - \hat{w}_k\right) - \hat{w}_{opt} \\ &= \hat{w}_k - \mu\Phi_{xx}\upsilon_k - \hat{w}_{opt} \\ &= \upsilon_k - \mu\Phi_{xx}\upsilon_k = \left(I_L - \mu\Phi_{xx}\right)\upsilon_k \end{aligned} \tag{1.19}$$

Φ_{xx}, matrice d'autocorrélation définie positive, est diagonalisable par une transformation unitaire de similarité, c'est à dire $\Phi_{xx} = Q_L \wedge_L \left(Q_L\right)^T$, où $Q_L\left(Q_L\right)^T = I_L$ et $\wedge_L$ est une matrice diagonale dont les éléments $\{\lambda_1, \lambda_2, \ldots, \lambda_L\}$ sont les valeurs propres de Φ_{xx} et Q_L est la matrice unitaire des vecteurs propres associés aux valeurs propres de Φ_{xx}. L'équation (1.19) s'exprime de manière équivalente, en adoptant la notation $r_k = \left(Q_L\right)^T \upsilon_k$:

$$r_{k+1} = \left(I_L - \mu\wedge_L\right) r_k \tag{1.20}$$

Ainsi, l'étude de la stabilité du système est ramenée à l'étude de la convergence de L suites géométriques définies par :

$$\begin{aligned} &r_{k+1}(n) = (1 - \mu\lambda_n)\, r_k(n), \;\; n = 0, 1, \ldots, L-1 \\ &r_k(n) = (1 - \mu\lambda_n)^k\, r_0(n) \end{aligned} \tag{1.21}$$

Chacune de ces suites définit un mode de l'algorithme caractérisé par la valeur propre λ_n réelle positive (puisque Φ_{xx} est définie positive). Une condition suffisante assurant la stabilité du système est donc :

$$|1 - \mu\lambda_n| < 1 \tag{1.22}$$

Nous en déduisons alors la condition de convergence de l'algorithme du gradient déterministe :

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{1.23}$$

où λ_{max} désigne la plus grande valeur propre de la matrice d'autocorrélation du signal d'entrée.

1.5.2.2 Vitesse de convergence

Le temps de convergence Γ_n d'un mode donné λ_n est défini comme le temps que met la suite géométrique associée pour décroître exponentiellement de sa valeur initiale $r_0(n)$ à la valeur $r_0(n) \times \frac{1}{\mathbf{e}}$, $\mathbf{e}$ désignant la base du logarithme népérien. Ce temps de convergence est relié au pas d'adaptation et à la valeur propre λ_n de la façon suivante :

$$\Gamma_n = \frac{-1}{ln(1 - \mu\lambda_n)} \tag{1.24}$$

Pour de très faibles valeurs du pas d'adaptation, on peut faire l'approximation suivante :

$$\Gamma_n \approx \frac{1}{\mu\lambda_n}, \ \mu \ll 1 \tag{1.25}$$

Ce résultat est intéressant puisqu'il prouve que pour de faibles valeurs du pas d'adaptation, le temps de convergence est inversement proportionnel au pas d'adaptation. Plus le pas d'adaptation est faible, plus le temps de convergence est long. De plus, dans ce cas, le temps de convergence est inversement proportionnel à la valeur propre correspondante.

1.5.2.3 Erreur quadratique moyenne

En utilisant le même procédé, il est possible d'exprimer l'erreur quadratique moyenne en fonction des valeurs propres de la matrice d'autocorrélation du signal d'entrée, soit : [Feur1985]

$$\xi_k = \xi_{min} + \sum_{n=0}^{L-1} \lambda_n \left(r_k(n)\right)^2 = \xi_{min} + \sum_{n=0}^{L-1} \lambda_n \left(1 - \mu\lambda_n\right)^{2k} \left(r_0(n)\right)^2 \tag{1.26}$$

Ainsi, lorsque l'algorithme du gradient déterministe est convergent, c'est à dire que le pas d'adaptation μ a une valeur dans le domaine de convergence défini par l'équation (1.23), la limite de l'erreur quadratique moyenne, quand k tend vers l'infini, est ξ_{min} et ce, quelles que soient les conditions initiales.

La relation (1.26) montre également que, pour un pas d'adaptation fixé, la convergence vers ξ_{min} est dépendante de la convergence des séries géométriques $\left(\left(1 - \mu\lambda_n\right)^{2k}\right)$. La convergence la plus rapide étant assurée pour le mode correspondant à λ_{max}. Le temps de convergence global Γ de l'algorithme du gradient déterministe est donc compris dans l'intervalle suivant :

$$\frac{-1}{ln\left(1-\mu\lambda_{max}\right)} < \Gamma < \frac{-1}{ln\left(1-\mu\lambda_{min}\right)} \tag{1.27}$$

Ce résultat souligne donc l'influence de la dispersion des valeurs propres de la matrice d'autocorrélation du signal d'entrée sur le temps de convergence de l'algorithme qui est borné par les valeurs propres les plus faibles.

L'algorithme du gradient déterministe n'a pas d'intérêt dans notre application car il nécessite la connaissance de la matrice d'autocorrélation. L'étude théorique de cet algorithme, rappelée en grandes lignes dans ce paragraphe, est cependant très utile puisqu'une analogie peut être établie facilement pour l'étude du comportement des algorithmes du type stochastique.

1.5.3 Algorithmes du gradient stochastique LMS

La popularité de l'algorithme du gradient stochastique ou LMS ($Least\ Mean\ Square$), s'explique par sa simplicité de mise en oeuvre, sa charge de calcul réduite qui est de l'ordre de $2L$ opérations par échantillon et sa relative facilité d'analyse mathématique.

Comme il a été vu au paragraphe précédent, la minimisation de $l'EQM$ à la sortie peut être réalisée à l'aide de l'algorithme du gradient stochastique si on dispose de certaines statistiques sur les signaux d'entrée. Cet algorithme itératif, connu aussi sous le nom des moindres carrés, consiste à minimiser l'erreur quadratique moyenne ainsi obtenue, soit : [Shankar1983, David1984, Kazuo1990]

$$\bigtriangledown\left(e_k^2\right) = \bigtriangledown\left(d_k - \hat{y}_k\right)^2$$

L'erreur entre le signal estimé et l'écho est donnée par :

$$e_k = d_k - \hat{y}_k = d_k - \chi_k^T \hat{w}_k$$

L'algorithme LMS découle directement de l'algorithme du gradient déterministe présenté dans le paragraphe précédent. En pratique, nous ne disposons que très rarement du gradient exact $\bigtriangledown\xi$ donné par l'équation (1.15). Une estimation est donc nécessaire. Dans le cas de l'algorithme LMS, l'estimation du gradient est réalisée en remplaçant l'équation (1.14) par : [Bernard1975, Bernard1976, Gritton1984, Sen1997, George1999]

$$\hat{w}_{k+1} = \hat{w}_k - \frac{1}{2}\mu\nabla\left(e_k^2\right) \tag{1.28}$$

où μ est le pas d'adaptation qui contrôle la convergence du filtre adaptatif.

Le gradient $\nabla\left(e_k^2\right)$ est donné par :

$$\nabla\left(e_k^2\right) = \frac{\partial\left(e_k^2\right)}{\partial\hat{w}_k} = 2e_k\frac{\partial e_k}{\partial\hat{w}_k} = -2\chi_k e_k \tag{1.29}$$

A partir des équations (1.28) et (1.29), l'algorithme LMS peut s'écrire :

$$\hat{w}_{k+1} = \hat{w}_k + \mu\chi_k e_k \tag{1.30}$$

$$e_k = d_k - \hat{y}_k = d_k - \chi_k^T\hat{w}_k \tag{1.31}$$

L'algorithme du gradient stochastique tire appellation du fait qu'il réalise implicitement une estimation du gradient par itération sur les données instantanées [Macchi1998].

La complexité algorithmique de l'algorithme LMS est connue : chaque itération k comporte $(2L+1)$ multiplications et $(2L)$ additions. Le tableau (1.1) résume les opérations nécessaires pour l'algorithme LMS à chaque itération k.

Etape de l'algorithme	Nombre d'additions	Nombre de multiplications
$e_k = d_k - \chi_k^T\hat{w}_k$	L	L
$\hat{w}_{k+1} = \hat{w}_k + 2\mu\chi_k^T e_k$	L	L+1
Total par itération	2L	2L+1

TAB. 1.1 – Complexité algorithmique de l'algorithme LMS

1.5.3.1 Convergence en moyenne vers la solution de Wiener

Il suffit d'étudier l'espérance mathématique du vecteur d'écart des coefficients par rapport à la solution optimale de Wiener et de reprendre la même démarche que celle utilisée pour l'algorithme du gradient déterministe. En faisant l'hypothèse que le vecteur d'entrée χ_k est statistiquement indépendant du vecteur des coefficients du filtre $\hat{w}_k$, il est possible de se ramener à une expression similaire à celle donnée par l'équation (1.20). Par conséquent, la convergence en moyenne de l'algorithme du gradient est garantie pour un pas d'adaptation μ satisfaisant l'équation :

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{1.32}$$

où λ_{max} est la plus grande valeur propre de la matrice d'autocorrélation Φ_{xx} du signal d'entrée donnée par l'équation (1.8) [Bernard1975, Bernard1976, Sen1997]. La dépendance par rapport aux valeurs propres de la matrice d'autocorrélation est identique à celle mentionnée pour l'algorithme du gradient déterministe.

1.5.3.2 Convergence en moyenne quadratique

L'étude de la convergence en moyenne quadratique, consistant à étudier la condition de convergence de $\sigma_k^2(e) = E\left(e_k^2\right)$, mène à une condition de convergence plus stricte que celle obtenue dans le cas de la convergence en moyenne. En effet, il est possible de montrer que, dans le cas particulier d'un bruit blanc en signal d'entrée, et en ayant fait au préalable une hypothèse d'indépendance entre l'erreur et les données en entrée, la variance de l'erreur s'exprime de la façon suivante : [Haykin1986]

$$\sigma_{k+1}^2(e) = \sigma_k^2(e)\left(1 - 2\mu\sigma^2(x) + \mu^2 L\sigma^4(x)\right) + 2\mu\sigma^2(x)\,\sigma_{opt}^2 \tag{1.33}$$

où σ_{opt}^2 désigne la variance de l'erreur optimale. De cette expression, découlent plusieurs résultats importants. La convergence en moyenne quadratique n'a lieu que si le pas d'adaptation satisfait la condition : [Farhang1998, Shankar1983, Shu1988, Bernard1985, Odile1995, Simon1996]

$$\mu < \frac{2}{L\sigma^2(x)} \tag{1.34}$$

Le pas d'adaptation μ_{opt} assurant la convergence en moyenne quadratique la plus rapide est donné par :

$$\mu_{opt} = \frac{2}{L\sigma^2(x)} \tag{1.35}$$

A l'infini, la variance de l'erreur $\sigma_k^2(e)$ se stabilise à une valeur constante $\sigma_\infty^2(e)$ qui est donnée par :

$$\sigma_\infty^2(e) = \frac{2\sigma_{opt}^2}{2 - \mu L\sigma^2(x)} \tag{1.36}$$

Le désajustement final, défini par $\tilde{\xi} = \frac{\sigma_\infty^2(e) - \sigma_{opt}^2}{\sigma_{opt}^2}$, pour de très faibles valeurs du pas d'adaptation, peut être approximé par :

$$\tilde{\xi} \approx \frac{1}{2}\mu L\sigma^2(x) \tag{1.37}$$

Ces différents résultats permettent de démontrer en outre que les propriétés de convergence de l'algorithme LMS dépendent de la statistique du signal d'entrée (équation 1.35) et que l'erreur d'identification est proportionnelle au pas d'adaptation (équation 1.37). En tenant compte du résultat de l'équation (1.25), cela signifie clairement qu'il est nécessaire de faire un compromis entre vitesse de convergence et niveau de l'erreur résiduelle. Les résultats précédents s'étendent au cas d'un signal d'entrée corrélé (un signal de parole, par exemple), signal pour lequel la dispersion des valeurs propres de la matrice d'autocorrélation n'altère que davantage les performances de convergence.

Pour les signaux non-stationnaires, il est difficile de suivre les variations du signal d'entrée avec l'adapta-

tion du filtre par l'algorithme LMS, ce qui donne une convergence lente.

1.5.4 Algorithme des Moindres Carrés Récursif (RLS)

Une solution, permettant de suivre les non-stationnarités du signal, est donnée par l'algorithme des moindres carrés récursifs RLS (*Recursive Least Squares*). Dans la méthode des moindres carrés récursive, au lieu de minimiser un critère statistique établi sur l'erreur commise en estimant un signal $\{d\}$, on minimise, à chaque itération k, la somme pondérée des carrés des erreurs commises depuis l'instant initial. Dans ce cas la fonction coût ξ_k est donnée par : [Ljung1983, Haykin1986]

$$\xi_k = \sum_{n=0}^{k} (d_n - \hat{y}_n)^2 \tag{1.38}$$

L'estimation du signal $\{d\}$ au moyen de la méthode des moindres carrés, en utilisant une réponse impulsionnelle $\hat{w}_k$, est obtenue lorsque la fonction coût ξ_k est minimisée.

La réponse impulsionnelle est donc fonction des échantillons disponibles et non pas d'une moyenne statistique générale. Par analogie avec Wiener, elle est donnée par la relation :

$$R_{k,xx}\hat{w}_k = R_{k,xd} \tag{1.39}$$

où

$$R_{k,xx} = \sum_{n=0}^{k} \chi_n \chi_n^T \tag{1.40}$$

et

$$R_{k,xd} = \sum_{n=0}^{k} \chi_n d_n \tag{1.41}$$

La réponse impulsionnelle du filtre est donc à modifier à chaque nouvelle itération. Pour limiter le nombre de calculs, nous utilisons une équation récursive donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + R_{k,xx}^{-1}\chi_k e_k \tag{1.42}$$

où

$$R_{k,xx}^{-1} = \frac{1}{\delta}\left[R_{k-1,xx}^{-1} - \frac{R_{k-1,xx}^{-1}\chi_k^T\chi_k R_{k-1,xx}^{-1}}{\left(\delta + \chi_k R_{k-1,xx}^{-1}\chi_k^T\right)}\right] \tag{1.43}$$

$R_{0,xx} = \frac{1}{\delta} I_L$, $0 < \delta < 1$

$\hat{w}_0 = 0$

Ces équations (1.42) et (1.43) sont connues sous le nom de l'algorithme *RLS*.

Dans un système d'annulation d'écho acoustique, un algorithme adaptatif doit répondre à deux critères que sont la vitesse de convergence et la complexité de calcul. Si un algorithme répondait à ces deux critères simultanément, il serait systématiquement utilisé.

L'algorithme *RLS*, moins stable et plus difficile à mettre en application, nécessite plus d'opérations que l'algorithme *LMS* mais il présente l'avantage de converger plus rapidement que ce dernier.

Un algorithme permettant de répondre simultanément aux deux critères que sont la simplicité de calcul et la convergence rapide a été proposé et étudié par [Kazuo1990]. Il s'agit de l'algorithme $NLMS$ (*Normalized Least Mean Square*) dont le principe est décrit par le paragraphe qui suit.

1.5.5 Algorithme LMS Normalisé (NLMS)

L'algorithme $NLMS$ (*Normalized Least Mean Square*) consiste à normaliser le pas d'adaptation μ dans l'algorithme LMS par rapport à l'énergie du signal d'entrée pour réduire au minimum l'effet de la variation de la puissance du signal d'entrée et de rendre ainsi la convergence plus au moins uniforme en passant d'une étape d'adaptation à une autre.

Dans l'équation (1.30), le pas d'adaptation μ est alors remplacé par un pas d'adaptation μ_k défini à chaque itération par :

$$\mu_k = \frac{\mu}{\chi_k^T \chi_k + \beta} \tag{1.44}$$

La mise à jour des coefficients du filtre adaptatif par l'algorithme $NLMS$ est alors donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \mu_k \chi_k e_k = \hat{w}_k + \mu \frac{\chi_k e_k}{\chi_k^T \chi_k + \beta} \tag{1.45}$$

où β est un facteur permettant de suivre plus ou moins rapidement les variations d'énergie dans le signal d'entrée $\{x\}$.

La convergence de cet algorithme est garantie pour un pas d'adaptation $0 < \mu_k < \frac{2}{\lambda_{max}}$.

L'intérêt de l'algorithme $NLMS$ par rapport à l'algorithme LMS est de rendre l'algorithme indépendant de la variance du signal d'entrée. Cependant, la distribution des valeurs propres de la matrice d'autocorrélation du signal d'entrée n'est en rien modifiée. Ceci implique précisément la même dépendance, dans les deux cas, de la convergence vis à vis de la statistique du signal d'entrée.

Pour les signaux stationnaires tels que le bruit blanc ou non-stationnaires tels que la parole, l'algorithme $NLMS$ apporte une amélioration significative sur le taux de convergence par rapport au LMS grâce à la normalisation du pas d'adaptation.

Cet algorithme peut être plus complexe que l'algorithme LMS mais il reste toujours l'un des algorithmes

les plus simples à mettre en application. Il est souvent utilisé dans la technologie d'annulation d'écho avec ses différentes versions présentées dans les paragraphes qui suivent.

Un des inconvénients de l'algorithme $NLMS$ par rapport au LMS est l'augmentation de la complexité de calcul au niveau du nombre de multiplications. Le tableau (1.2) résume le nombre d'opérations nécessaires dans l'algorithme $NLMS$ à chaque itération k.

Etape de l'algorithme	Nombre d'additions	Nombre de multiplications
$e_k = d_k - \chi_k^T \hat{w}_k$	L	L
$\hat{w}_{k+1} = \hat{w}_k + \mu \frac{\chi_k^T e_k}{\chi_k^T \chi_k + \beta}$	L+1	2L+1
Total par itération	2L+1	3L+1

TAB. 1.2 – Complexité algorithmique de l'algorithme NLMS

1.5.6 Algorithme Proportionné Normalisé LMS (PNLMS)

L'algorithme $PNLMS$ représente une nouvelle technique de filtrage adaptatif dans le cas d'annulation d'écho acoustique. Cette technique consiste à adapter chaque étape à une valeur désirée avec un taux de convergence qui varie d'une étape d'adaptation à l'autre. Cet algorithme exploite la faible densité des réponses impulsionnelles pour réaliser une adaptation sensiblement plus rapide que celle réalisée par l'algorithme conventionnel $NLMS$.

L'algorithme $PNLMS$ nécessite plus d'opérations que l'algorithme $NLMS$ mais il présente l'avantage de converger plus rapidement que ce dernier.

L'algorithme $PNLMS$ résulte directement de l'algorithme $NLMS$, décrit précédemment, en remplaçant le pas d'adaptation μ_k par : [Duttweiler2000, Gay1998, Benesty2001]

$$\mu_k = \frac{\mu G_k}{\chi_k^T G_k \chi_k + \beta} \tag{1.46}$$

où, $G_k = diag \left[\begin{array}{cccc} g_k(0), & \ldots, & g_k(L-1) \end{array} \right]$ est la matrice diagonale $(L \times L)$ avec $g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\Sigma_{m=0}^{L-1}\gamma_k(m)}$, où $\gamma_k(n) = max\{\rho\nu_k, |\hat{w}_k(n)|\}, n \in \{0, \ldots, L-1\}$ et où $\nu_k = max\{\delta, |\hat{w}_k(0)|, \ldots, |\hat{w}_k(L-1)|\}$. Les termes ρ et δ sont respectivement choisis égaux à $\frac{5}{L}$ et à 10^{-2}.

L'algorithme $PNLMS$ est alors défini par les équations :

$$e_k = d_k - \hat{y}_k = d_k - \chi_k^T \hat{w}_k \tag{1.47}$$

$$\hat{w}_{k+1} = \hat{w}_k + \mu_k \chi_k e_k = \hat{w}_k + \mu \frac{G_k \chi_k e_k}{\chi_k^T G_k \chi_k + \beta} \tag{1.48}$$

Si la réponse impulsionnelle du filtre adaptatif est dispersive, dans ce cas là, la convergence de l'algorithme $PNLMS$ peut être réellement plus lente que l'algorithme $NLMS$. Pour limiter ce problème de convergence

lente, nous utilisons l'algorithme $PNLMS++$ combinaison des deux algorithmes $NLMS$ et $PNLMS$, dont le principe est décrit dans le paragraphe qui suit.

1.5.7 Algorithme PNLMS++

L'algorithme $PNLMS++$ est obtenu à partir des deux algorithmes précédents $NLMS$ et $PNLMS$ comme le montre la figure (1.8) [Duttweiler2000, Gay1998, Benesty2001].

Pour les itérations de numéro impair (k impair), on procède de la même manière que dans le cas de l'algorithme $PNLMS$ et pour les itérations de numéro pair (k pair), on procède de la même manière que dans le cas de l'algorithme $NLMS$.

L'alternance entre les deux algorithmes $NLMS$ et $PNLMS$ permet de rendre l'algorithme $PNLMS++$ moins sensible aux variations de la réponse impulsionnelle de l'écho.

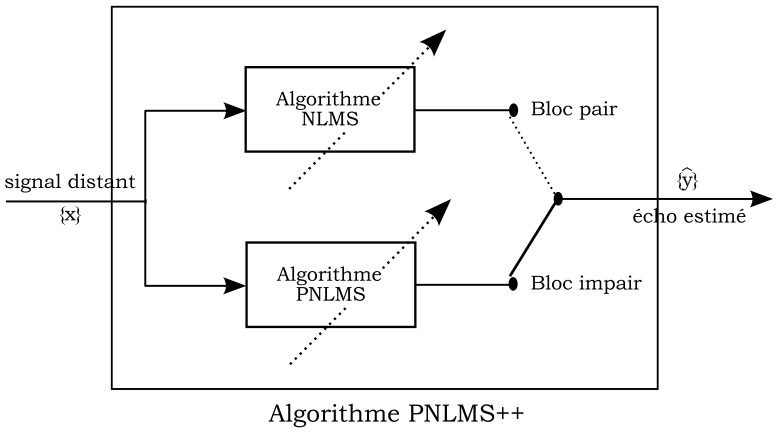

FIG. 1.8 – Principe d'adaptation par l'algorithme PNLMS++

1.6 Comparaison des performances des différents algorithmes (NLMS, PNLMS, PNLMS++)

Des simulations numériques ont été réalisées pour évaluer les performances des algorithmes de filtrage adaptatif (*NLMS, PNLMS* et *PNLMS++*) à l'aide du logiciel de programmation *MatLab*.

Les différents algorithmes sont évalués avec un signal de parole et une réponse impulsionnelle de chemin d'écho présentés respectivement par les figures (1.9-a) et (1.9-b).

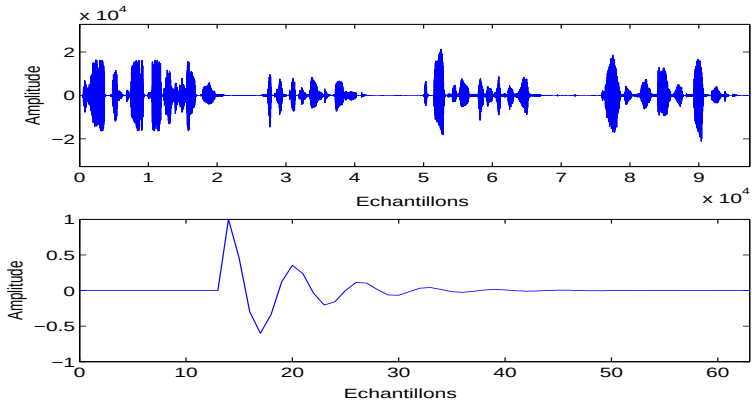

FIG. 1.9 – (a) : Signal d'entrée (b) : Réponse impulsionnelle du chemin d'écho

Nous avons utilisé des mesures objectives comme la convergence du filtre adaptatif, mesurée en dB, par :

$$N_m = 10log_{10}\left[\frac{\|w-\hat{w}\|^2}{\|w\|^2}\right] \tag{1.49}$$

w et $\hat{w}$ désignent respectivement la réponse impulsionnelle et la réponse impulsionnelle estimée du chemin d'écho.

Les résultats des simulations de cette comparaison sont représentés par la figure (1.10). Ils sont obtenus en choisissant L, longueur du filtre adaptatif égale à 64 et le pas d'adaptation $\mu = 0.1$. Les autres paramètres sont donnés par : $\beta = 10^2$, $\delta = 10^{-2}$, $\rho = \frac{5}{L}$.

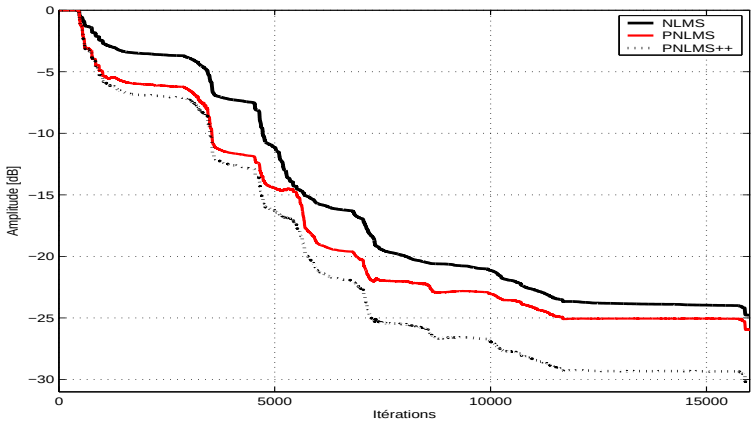

FIG. 1.10 – Convergence des coefficients du filtre adaptatif par les différentes méthodes d'adaptation

A partir de cette figure, nous constatons que l'algorithme $PNLMS++$ fournit une meilleure convergence par rapport aux autres algorithmes, $NLMS$ et $PNLMS$. Il est alors intéressant d'utiliser cet algorithme pour rendre au maximum l'écho résiduel inaudible à la sortie du système d'annulation d'écho acoustique.

1.7 Conclusion

Ce chapitre rappelle le principe de l'annulation d'écho acoustique et les différentes méthodes algorithmiques existantes. En effet, le problème de l'écho acoustique est classiquement résolu à l'aide d'un annulateur d'écho en identifiant progressivement la Réponse Impulsionnelle Finie (*FIR : Finite Impulse Response*) d'un filtre adaptatif à partir des algorithmes de type moindres carrés (LMS : *Least Mean Squares*).

Il rappelle aussi les différents types de l'algorithme des moindres carrés (LMS, $NLMS$, $PNLMS$, et $PNLMS++$) qui sont utilisés dans l'annulateur d'écho. Ces algorithmes de filtrage adaptatif présentent une charge de calcul globale importante.

L'objectif principal est alors de réduire cette charge de calcul en traitant les différents algorithmes adaptatifs (LMS, $NLMS$, $PNLMS$, et $PNLMS++$) par blocs d'échantillons au lieu de les traiter, échantillon par échantillon.

Ce traitement par blocs, qui fait appel au calcul d'une série de produits de convolution, peut être entièrement réalisé, dans un premier temps, par la mise en oeuvre de la transformée de Fourier rapide (FFT : *Fast Fourier Transform*), puis, dans un deuxième temps, par la transformée en nombres entiers (NTT : *Number Theoretic Transform*) et en particulier par la transformée en nombres de Fermat (FNT : *Fermat Number Transform*). Cette dernière transformée, qui présente de nombreux avantages par rapport à la FFT, est décrite par le chapitre suivant.

Chapitre 2

Les transformées en nombre entiers

L'implantation temps réel des algorithmes de traitement de signal est soumise à un certain nombre de contraintes telles que le temps d'exécution et la facilité de mise en oeuvre. Parmi les processeurs disponibles sur le marché, les DSP à virgule flottante, sur lesquels les calculs mathématiques sont relativement faciles à mettre en oeuvre, présentent toutefois des inconvénients de consommation et de prix. L'implantation d'algorithmes sur des processeurs à virgule fixe, répondant mieux aux contraintes technologiques et économiques, est plus délicate à réaliser. Pour s'affranchir de certaines difficultés de programmation sur ce dernier type de DSP, nous allons introduire un outil mathématique, appelé transformée en nombre entiers (NTT : $Number\ Theoretic\ Transform$), fort appréciable dans l'optique d'une implantation complète d'un système d'annulation d'écho acoustique en virgule fixe.

Les transformées en nombres entiers restent très peu utilisées dans le domaine du traitement de signal. Néanmoins, cet outil semble être intéressant pour réduire la complexité d'algorithmes standards, tel que les fonctions de filtrage. En effet, la transformée en nombres entiers, initialement développée pour permettre le calcul rapide de la convolution, présente deux avantages principaux par rapport à une implantation basée sur la transformée de Fourier discrète (DFT : $Discrete\ Fourier\ Transform$) :

• Tous les calculs s'effectuent sur l'ensemble $\mathbb{Z}$ des entiers relatifs, ce qui est particulièrement intéressant pour une implantation sur processeur DSP à virgule fixe, en terme de coûts de calcul et de performances.

• L'utilisation d'une transformée en nombres entiers évite le passage en complexe inhérent aux transformées de Fourier, ce qui permet une précision de calcul améliorée. Les calculs dans $\mathbb{Z}$ étant exacts, toute erreur d'arrondi est supprimée [Julien1991]. De plus, la multiplication complexe requiert des multiplications de réels tandis qu'une multiplication dans $\mathbb{Z}$ est une opération simplifiée.

Dans ce chapitre, nous introduisons la notion de transformée en nombres entiers et de transformée en nombres de Fermat (FNT : $Fermat\ Number\ Transform$) en particulier. Quelques algorithmes rapides liés à la transformée de Fourier sont considérés et mis en application pour des transformées en nombres entiers. Pour

chaque algorithme la connaissance d'opérations arithmétiques étant nécessaire, plusieurs résultats connus du domaine de la théorie des nombres sont rappelés et quelques aspects sur la représentation binaire des nombres entiers appartenant à un ensemble d'entiers $\mathbb{Z}$ sont abordés. Pour conclure, une partie résumera les différents points, abordés par la suite, qui seront importants pour l'application de transformée en nombres de Fermat à certains algorithmes adaptatifs du système d'annulation d'écho acoustique.

2.1 Rappels arithmétiques

Un anneau est un système algébrique, noté $\left(\begin{array}{cccc} A, & + , & \times \end{array} \right)$, se composant d'un ensemble A d'éléments auquel sont associées les opérations arithmétiques d'addition et de multiplication, dont les résultats seront toujours des éléments de l'ensemble de départ.

Propriété 2.1 *Soit $\mathbb{Z}$ un anneau d'entiers ; pour $\left(\begin{array}{cc} x, & y \end{array} \right) \in \mathbb{Z}^2$ alors $\left(\begin{array}{cc} x+y, & xy \end{array} \right) \in \mathbb{Z}^2$*

Notons $\mathbb{Z}_q$ un anneau d'ordre égal à un entier q quelconque, contenant l'ensemble des entiers $\{0, 1, \ldots, q-1\}$. Un entier relatif x appartenant à $\mathbb{Z}$ est alors défini dans $\mathbb{Z}_q$ comme étant le reste r da la division euclidienne de x par q, notée x/q. Les entiers $\left\{ \begin{array}{cc} x, & r \end{array} \right\}$ sont dits congrus modulo q :

$$\langle x = r + kq \equiv r \rangle_q \tag{2.1}$$

où k appartient à $\mathbb{Z}$ et $\equiv$ représente la notation de congruence. Dans l'équation (2.1) et par la suite, l'opération de réduction par le modulo q est notée $\langle . \rangle_q$ ou mod q. Précisons que si $r = 0$ alors q divise $x = kq$; cette notion arithmétique est représentée par $q \mid x$.

Propriété 2.2 *Soient les entiers $\left(\begin{array}{cc} x, & y \end{array} \right) \in \mathbb{Z}^2$; x est congru à y modulo q si et seulement si $q \mid (x-y)$*

La congruence modulo q est en fait une relation d'équivalence dans $\mathbb{Z}$. La classe d'équivalence de a, notée $\dot{a}$, est alors donnée par :

$$\dot{a} = \left\{ b \in \mathbb{Z}_q / \; \exists \, k \in \mathbb{Z}, a = b + kq / \; \langle a \equiv b \rangle_q \right\} \tag{2.2}$$

Les q classes d'équivalence $\dot{a}$ pour a appartenant à $\{0, 1, ..., q-1\}$ sont alors toutes distinctes et le groupe $\mathbb{Z}/\mathbb{Z}_q = \{\dot{a},\ 0 \leq a < q\}$ représentera l'ensemble de ces classes. Tous les résultats d'opérations arithmétiques modulo q, d'addition de soustraction et de multiplication, seront alors définis dans $\mathbb{Z}/\mathbb{Z}_q$.

Propriété 2.3 *Pour les entiers $\left(\begin{array}{cc} x, & y \end{array} \right) \in \left(\mathbb{Z}/\mathbb{Z}_q \right)^2$ alors $\left(\begin{array}{ccc} x+y, & x-y, & xy \end{array} \right) \in \left(\mathbb{Z}/\mathbb{Z}_q \right)^3$*

La division étant aussi un opérateur possible dans un anneau, l'inverse des éléments doit pour cela exister dans $\mathbb{Z}/\mathbb{Z}_q$. L'inverse b d'un entier a appartenant à $\mathbb{Z}/\mathbb{Z}_q$ est alors défini comme étant le plus petit entier positif qui vérifie $\langle ab \equiv 1 \rangle_q$.

Propriété 2.4 *L'inverse d'un élément existe dans $\mathbb{Z}/\mathbb{Z}_q$ si et seulement si l'élément et l'ordre de l'anneau $\mathbb{Z}_q$, égal à l'entier q, sont premiers entre eux.*

Si q est un nombre premier, tout élément non-nul de l'anneau $\mathbb{Z}_q$ possèdera un inverse appartenant à $\mathbb{Z}_q$. L'opération de division étant alors définie sur l'anneau $\mathbb{Z}/\mathbb{Z}_q$, ce dernier devient, par ce fait, un corps fini d'ordre q appelé corps de Galois et noté $GF\ (q)$ (annexe A).

2.1.1 Détermination d'un inverse

Il existe différentes méthodes pour calculer l'inverse d'un entier sur un ensemble $\mathbb{Z}_q$ d'ordre égal à un nombre premier q. Selon les conditions initiales de calcul, la meilleure méthode devra être préalablement choisie. La méthode la plus utilisée est toutefois l'algorithme d'Euclide qui reste valable quels que soient l'entier a à inverser et l'ordre q du corps de Galois $GF\ (q)$.

Proposition 2.1 *Le calcul de l'inverse d'un entier a dans l'ensemble $\mathbb{Z}_q$ consiste à résoudre l'équation* (*voir* annexe B) :

$$\langle ax + kq \equiv 1 \rangle_q \tag{2.3}$$

où $k \in \mathbb{Z}$ et x représente un entier inconnu égal à l'inverse de a modulo q.

La solution de cette équation peut être obtenue par l'algorithme d'Euclide étendu, décrit par la suite [Aho1974]. D'autres méthodes existent pour déterminer un inverse mais certaines ne sont utilisables que pour des conditions très particulières. Ainsi pour des valeurs de q égales à des nombres de Fermat, des calculs plus efficaces d'inverses sont possibles. Les deux propositions suivantes sont faites pour les cas particuliers où le modulo sera égal à un des nombres de Fermat qui s'expriment sous la forme $q = 2^{2^t} + 1$ avec $t \in \mathbb{N}$.

Proposition 2.2 *L'inverse d'un entier $a \in \mathbb{Z}_q$ avec $q = 2^{2^t} + 1$ premier et $t \in \mathbb{N}$ est donné par* :

$$\left\langle a^{-1} = a^{2^{2^t-1}} a^{2^{2^t-2}} ... a^{2^1} a^{2^0} \right\rangle_q \tag{2.4}$$

Démonstration 2.1

La congruence $\left\langle x^{q-1} \equiv 1 \right\rangle_q$ étant facilement vérifiable, quel que soit l'entier x appartenant à l'ensemble d'entiers $\mathbb{Z}_q$ d'ordre q, nombre premier quelconque, le développement de l'équation (2.4) peut être déduit par récurrence. Posons la congruence suivante :

$$\left\langle x^{-1} = x^{-1}.1 \equiv x^{-1} x^{q-1} = x^{q-2} = x^{2^{2^t}-1} \right\rangle_q \tag{2.5}$$

• Pour $t = 0$, nous avons :

$$\left\langle x^{-1} = x^{2^{2^t}-1} = x^{2^1-2^0} = \frac{x^{2^1}}{x^{2^0}} = x^{2^0} \right\rangle_q$$

• Pour $t = 1$, nous obtenons, à partir de l'égalité $x^{2^2} = x^{2^1} x^{2^1}$, la congruence suivante :

$$\left\langle x^{-1} = x^{2^{2^t}-1} = x^{2^2-2^0} = \frac{x^{2^1} x^{2^1}}{x^{2^0}} = x^{2^1} x^{2^0} \right\rangle_q$$

• Pour $t > 1$, nous déduisons alors le développement de $x^{2^{2^t}-1}$:

$$\begin{cases} \left\langle x^{-1} \equiv x^{2^{2^t}-1} = x^{2^{2^t}-2^0} = \frac{x^{2^{2^t}-1} x^{2^{2^t}-1}}{x^{2^0}} = \frac{x^{2^{2^t}-1} x^{2^{2^t}-2} x^{2^{2^t}-2}}{x^{2^0}} \right\rangle_q \\ \left\langle x^{-1} \equiv \frac{x^{2^{2^t}-1} x^{2^{2^t}-2} \ldots x^{2^2}}{x^{2^0}} = x^{2^{2^t}-1} x^{2^{2^t}-2} \ldots x^{2^1} \frac{x^{2^1}}{x^{2^0}} = x^{2^{2^t}-1} x^{2^{2^t}-2} \ldots x^{2^1} x^{2^0} \right\rangle_q \end{cases}$$

L'équation (2.4) a donc été démontrée. Ce calcul permettra de déterminer l'inverse d'un entier dans l'ensemble $\mathbb{Z}_{2^{2^t}+1}$ et nécessitera $2\left(2^{2^t}-1\right)$ multiplications. Une autre méthode, plus contraignante, qui n'est valable que pour des entiers $a \in \mathbb{Z}_{2^{2^t}+1}$ égaux à une puissance de 2, ce qui assure une valeur entière à $log_2(a)$, est présentée dans la proposition suivante.

Proposition 2.3 *Pour tout entier de la forme $a = 2^k$ avec $k \in \mathbb{N}^\star$, l'inverse de a modulo $q = 2^{2^t}+1$ est alors obtenu par* :

$$\left\langle a^{-1} \equiv -2^{2^t-k} = -2^{2^t-log_2(a)} \right\rangle_{2^{2^t}+1} \tag{2.6}$$

Démonstration 2.2

Partant de la congruence $\left\langle 2^{2^t} \equiv -1 \right\rangle_{2^{2^t}+1}$, l'équation (2.6) peut être démontrée ainsi :

$$\begin{cases} \left\langle 2^{2^t} 2^{-k} \equiv -1.2^{-k} \right\rangle_{2^{2^t}+1} \\ \left\langle a^{-1} = 2^{-k} \equiv -2^{2^t-k} \right\rangle_{2^{2^t}+1} \end{cases}$$

Dans le cas d'implantation numérique binaire, ce calcul d'inverse ne requiert pas de multiplication mais uniquement des décalages de bits. Notons que cette dernière proposition sera utilisée dans la détermination de la transformée inverse en nombres de Fermat.

2.1.1.1 Algorithme d'Euclide étendu

La solution de l'équation (2.3) est calculée par l'algorithme d'Euclide étendu décrit par les relations de récurrences suivantes (annexe B). Tant que $S_i \neq 0$, les opérations suivantes sont répétées :

$$\begin{cases} Q_{i+1} = ent\left(\frac{S_{i-1}}{S_i}\right) \\ S_{i+1} = rem\left(\frac{S_{i-1}}{S_i}\right) = \langle S_{i-1}\rangle_{S_i} \\ x_{i+1} = x_{i-1} - Q_{i+1}x_i \\ i = i+1 \end{cases} \tag{2.7}$$

où les opérateurs $ent(\cdot)$ et $rem(\cdot)$ représentent la partie entière et le reste d'une division. Les conditions initiales de l'algorithme sont données par $S_0 = q$, $S_1 = a$, $x_0 = 0$, $x_1 = 1$ et $i = 1$. Lorsqu'à la $i^{ème}$ itération la valeur de S_i est nulle, l'inverse de a est donné par $\left\langle a^{-1} = x_{i-1}\right\rangle_q$. Notons que cet algorithme nécessite un nombre de multiplications de l'ordre de $2log_2 q$.

2.1.2 Système des résidus

Pour certaines applications, la dynamique demandée aux intervalles de données peut être élevée. Une réponse possible à cette difficulté est de travailler sur des Systèmes de Résidus (RNS : *Residue Number System*) où le Théorème du Reste Chinois (CRT : *Chinese Residue Theorem*) pourra s'appliquer aux différentes séquences de données [Aho1974].

Pour tout entier q, $\mathbb{Z}_q$ désigne l'anneau des résidus modulo q. Soient q_1 et q_2 deux nombres entiers premiers entre eux, la fonction f appliquée sur un entier $x \in \mathbb{Z}_{q_1 q_2}$ est un isomorphisme d'anneaux.

$$\begin{aligned} f : \mathbb{Z}_{q_1 q_2} &\longmapsto \mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \\ x &\longmapsto \left(x \, mod \, q_1, \quad x \, mod \, q_2 \right) \end{aligned}$$

Une généralisation s'effectue sans peine. Soit $q = \Pi_{i=1}^{k} q_i$ le produit de k nombres entiers premiers entre eux deux à deux, l'anneau $\mathbb{Z}_q$ sera isomorphe à l'anneau produit $\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \ldots \times \mathbb{Z}_{q_{k-1}} \times \mathbb{Z}_{q_k}$. Tout entier peut alors être représenté par ses résidus modulo des entiers premiers entre eux.

2.1.2.1 Théorème du reste chinois

Le Théorème du Reste Chinois ou Théorème Chinois apparaît pour la première fois dans le traité de *Juzhang Suhanshu* écrit entre l'an 208 et 473 après *J.C.* sous la forme d'un problème :

Nous souhaitons connaître le nombre d'objets en notre possession. Si nous les comptons par paquets de trois, il en reste deux ; si nous les comptons en paquets de cinq, le reste est trois ; si nous les comptons par paquets de sept, le reste est deux. Combien avons nous d'objets ?

Pour information, la réponse est égale au nombre vingt-trois. Si l'existence de ce traité en ces termes reste invérifiable, une certitude est l'utilisation par les armées chinoises de ce théorème pour dénombrer leurs soldats. Des valeurs de reste étaient déterminées en fonction du nombre d'hommes alignés par colonne et

permettaient une estimation exacte et rapide de l'effectif des troupes.

Tout entier $x \in \mathbb{Z}_q$ peut être représenté de manière unique par un ensemble de résidus $\{x_1, \ldots, x_k\}$ obtenus par réduction modulo un ensemble d'entiers premiers deux à deux $\{q_1, \ldots, q_k\}$ tel que $q = \Pi_{i=1}^{k} q_i$. La valeur initiale de x pourra ensuite être reconstruite grâce au Théorème Chinois :

$$x = \left\langle \sum_{i=1}^{k} \hat{q}_i \hat{q}_i^{-1} x_i \right\rangle_q \tag{2.8}$$

avec $\hat{q}_i = q/q_i$ et $\hat{q}_i^{-1}$ son inverse modulo q_i tel que $\left\langle \hat{q}_i \hat{q}_i^{-1} = 1 \right\rangle_{q_i}$.

2.1.2.2 Exemple

Pour illustrer l'utilisation du Théorème du Reste Chinois, le problème décrit par le traité de *Juzhang Suhanshu* est détaillé. Soient X et Q deux ensembles de nombres entiers contenant respectivement les restes et leur modulo associé :

$$\begin{cases} X = \left\{ \begin{array}{cccc} x_1, & x_2, & x_3 \end{array} \right\} = \left\{ \begin{array}{cccc} 2, & 3, & 2 \end{array} \right\} \\ Q = \left\{ \begin{array}{cccc} q_1, & q_2, & q_3 \end{array} \right\} = \left\{ \begin{array}{cccc} 3, & 5, & 7 \end{array} \right\} \end{cases}$$

La valeur de l'entier x, représentant le nombre d'objets possédés, appartient à l'anneau d'ordre $q = 3 \times 5 \times 7 = 105$. Pour déterminer le résultat final, des calculs intermédiaires sont nécessaires. Dans un premier temps, les valeurs $\hat{q}_i$ sont calculées :

$$\begin{cases} \hat{q}_1 = 105/3 = 5 \times 7 = 35 \\ \hat{q}_2 = 105/5 = 3 \times 7 = 21 \\ \hat{q}_3 = 105/7 = 3 \times 5 = 15 \end{cases}$$

Dans une seconde étape, leur inverse $\hat{q}_i^{-1}$ sont déduits :

$$\begin{cases} \left\langle \hat{q}_1^{-1} = 35^{-1} \equiv 2^{-1} \equiv 2 \right\rangle_3 \\ \left\langle \hat{q}_2^{-1} = 21^{-1} \equiv 1^{-1} = 1 \right\rangle_5 \\ \left\langle \hat{q}_3^{-1} = 15^{-1} \equiv 1^{-1} = 1 \right\rangle_7 \end{cases}$$

La valeur initiale x est alors retrouvée par :

$$x = \left\langle \sum_{i=1}^{3} \hat{q}_i \hat{q}_i^{-1} x_i \right\rangle_{q=105} = \langle 35 \times 2 \times 2 + 21 \times 1 \times 3 + 15 \times 1 \times 2 \rangle_{q=105} = 23$$

Ce résultat peut bien sûr être vérifié :

$$\begin{cases} x_1 = \langle 23 = 7 \times 3 + 2 \rangle_3 = 2 \\ x_2 = \langle 23 = 4 \times 5 + 3 \rangle_5 = 3 \\ x_3 = \langle 23 = 3 \times 7 + 2 \rangle_7 = 2 \end{cases}$$

Cet exemple illustre bien l'utilisation du Théorème du Reste Chinois pour la reconstruction d'un nombre ou d'une séquence originale à partir d'un ensemble de résidus. Notons que cette représentation bijective de nombres par un système de résidus (RNS) permet d'effectuer tous les calculs souhaités sur x de manière indépendante sur les ensembles de résidus avant de reconstituer le résultat final.

2.2 Les transformées en nombres entiers (NTT)

En 1972, *Rader* proposait des transformées définies sur un ensemble d'entiers d'ordre égal à un nombre de *Fermat* pour calculer, sans erreur d'arrondi, des produits de convolutions de séquences de nombres entiers [Rader1972]. Plus tard, *Agarwal et Burrus* [Agarwal1974] ont montré pour certaines longueurs de séquences que les transformées en nombres de Fermat peuvent être mises en oeuvre à l'aide d'additions, de soustractions, et de décalages de bits essentiellement, et en faisant très peu appel à des opérations de multiplication. L'application de ces transformées, malgré quelques contraintes arithmétiques, s'avère alors plus rapide qu'une implantation de transformées de Fourier conventionnelles. Avant de détailler les transformées en nombres de Fermat, différentes définitions liées aux transformées en nombres entiers et aux corps de Galois $GF\ (q)$ sont données. Afin de réduire la complexité de calcul des transformées en nombres entiers, le problème du choix du modulo q sera discuté.

2.2.1 Définition générale d'une NTT

Une transformée en nombres entiers (NTT) présente la même forme que la transformée de Fourier discrète (DFT : $Discrete\ Fourier\ Transform$) [Julien1991]. Cependant son domaine de définition n'est plus l'ensemble des complexes $\mathbb{C}$ mais un corps de Galois GF. Le calcul des transformées en nombres entiers utilise donc essentiellement des opérations arithmétiques dans le corps $GF\ (q)$ d'ordre q premier ou pseudo-premier. La racine primitive d'une transformée de Fourier d'ordre M dans $\mathbb{C}$, définie par le terme exponentiel $e^{j\frac{2\pi}{M}}$, est alors remplacée par la racine $M^{ème}$ de l'unité du corps de Galois $GF\ (q)$ représentée par le terme générateur α tel que :

$$\left\langle \alpha^M = 1 \right\rangle_q \tag{2.9}$$

Ainsi, la transformée en nombres entiers d'une séquence $x = \{x_n\}_{n=0}^{M-1}$, composée de M éléments x_n

appartenant au corps de Galois $GF\ (q)$ d'ordre q, est une séquence $\mathbf{X} = \{X_k\}_{k=0}^{M-1}$, dont les composantes X_k sont définies elles aussi dans le corps $GF\ (q)$ et déterminées par :

$$X_k = \left\langle \sum_{n=0}^{M-1} x_n \alpha^{nk} \right\rangle_q \tag{2.10}$$

où $k = 0, 1, ..., M-1$ et α représente le terme générateur d'ordre M, égal à la longueur de séquence de la transformée, du corps $GF\ (q)$. L'ordre M de l'élément α, avec $0 < M \leq q-1$, étant la valeur du plus petit entier positif p pour lequel $\langle \alpha^p = 1 \rangle_q$, remarquons que de manière générale, le terme générateur α ne sera pas nécessairement une racine primitive, ce qui entraînera que tous les termes $\left\{ \begin{array}{cccc} \alpha, & \alpha^2, & \ldots, & \alpha^{q-1} \end{array} \right\}$ ne seront pas toujours distincts.

Si la longueur de transformée M et le modulo q sont premiers entre eux, il existe un inverse M^{-1} dans le corps de Galois $GF\ (q)$ tel que $\left\langle M.M^{-1} \equiv 1 \right\rangle_q$. L'inverse d'une transformée en nombres entiers existe alors si $M \mid (q-1)$ et est donnée par :

$$x_n = \left\langle M^{-1} \sum_{k=0}^{M-1} X_k \alpha^{-nk} \right\rangle_q \tag{2.11}$$

avec $n = 0, 1, \ldots, M-1$.

Le modulo q pouvant être égal à un nombre premier p $(q = p)$, à une puissance m d'un nombre premier p $(q = p^m)$ ou à un produit de puissances m_i de nombres p_i premiers entre eux $\left(q = \Pi_{i=1}^{k} p_i^{m_i}\right)$; les conditions que doivent respecter les paramètres α, M et q pour qu'une transformée en nombres entiers et son inverse existent dans leur domaine de définition $GF\ (q)$ [Agarwal1974] sont les suivants :

• pgcd $\left(\begin{array}{cc} \alpha, & q \end{array} \right)$ = pgcd $\left(\begin{array}{cc} M, & q \end{array} \right) = 1$, soit $\langle \alpha^M = 1 \rangle_q$

• Si q est premier, M doit diviser $(q-1)$. Sinon q peut s'écrire $q = \Pi_{i=1}^{k} p_i^{m_i}$ et l'ordre M devra diviser le plus grand commun diviseur, diminué de l'unité : $M \mid$ pgcd $\left(\begin{array}{cc} p_i - 1, & p_j - 1 \end{array} \right)$ $\forall\ (i,j) \in \{1, 2, \ldots, k\}^2$ et $i \neq j$

• pgcd $\left(\begin{array}{cc} (\alpha^i - 1), & q \end{array} \right) = 1$ si $i \mid M$, $\forall i \in \{1, 2, \ldots, M-1\}$

Notons que de manière similaire à la transformée de Fourier discrète, le calcul d'une transformée en nombres entiers de longueur M peut être envisagé avec la mise en oeuvre d'un ordre de M^2 multiplications et de $M\,(M-1)$ additions. La multiplication par des puissances de la racine unité α étant l'opération arithmétique la plus complexe rencontrée dans le calcul d'une transformée en nombres entiers, l'efficacité de l'implantation d'une NTT sur processeur DSP et plus encore d'une intégration $VLSI$ (*Very Large Scale Integration*) dépendra en grande partie de l'optimisation de telles multiplications. Les valeurs données à α seront alors de préférence prises égales à des puissances de 2 pour permettre le remplacement des multiplications par α par de simples décalages de bits [Duhamel1982].

De plus, le choix de l'ordre de l'ensemble de définition $\mathbb{Z}_q$ des transformées en nombres entiers est très important pour une programmation optimale. Un choix de q approprié doit être fait de manière à fournir un large choix de longueurs de transformées possibles, qui correspondront au mieux aux séquences de données à traiter, ainsi qu'à garantir les performances et la faible complexité des architectures mises en oeuvre pour les opérations en nombres entiers réduites par le modulo q. D'un simple point de vue d'implantation, la réduction par un modulo q d'un nombre entier codé en binaire restera peu complexe à exécuter pour des valeurs de q égales à une puissance de deux ou si sa représentation binaire contient peu de 1. Ainsi, les opérations dans $\mathbb{Z}_{2^m}$ seront très simples à réaliser, mais 2 étant un facteur de $q = 2^m$, la longueur maximale de transformée dans cet anneau serait égale à 1. Les mêmes conclusions seront tirées pour tout anneau d'ordre pair. Ces modulos ne sont donc pas du tout intéressants pour l'application de transformées en nombres entiers. Le choix de q égal à un entier impair est donc inévitable. Tout nombre entier naturel impair pouvant s'écrire sous la forme $q = ca^b + 1$ avec c, a et b entiers, si les valeurs de q sont premières alors les longueurs de transformée M associées seront obtenues telles que M divise ca^b. Sachant que les transformées de Fourier peuvent être calculées plus efficacement si la longueur de transformée M est composée, le choix de ces paramètres entraînera une requête de l'ordre de $(a-1)\,Mlog_aM$ multiplications et d'autant d'additions. En effet, la longueur M étant exprimée comme une puissance d'un entier, la transformée en nombres entiers peut être décomposée en plusieurs transformées de petites tailles et implantée à l'aide des mêmes structures mises oeuvre pour une transformée de Fourier rapide [Blahut1985]. Notons seulement que plus la valeur de a sera petite plus l'implantation sera efficace. *Chevillat* donna ainsi une table [Chevillat1978] de modulos de cette forme codés sur 8 à 16 bits. Certains de ces modulos sont composés et la plupart d'entre eux est premier. Par exemple, un des nombres de Chevillat, $q = 2.3^9 + 1$ permet une longueur de transformée de $3^9 = 19683$ dans le corps de Galois $GF\,(39367)$. Cependant, la représentation binaire de q (1001100111000111) ne semble pas convenir à une exécution simple de la réduction modulo q.

2.2.2 Propriétés d'une transfomation en nombres entiers

Pour déterminer une transformée en nombres entiers directe et inverse, la racine $M^{ème}$ de l'unité α définie dans $GF\,(q)$ doit être prédéfinie en respectant toujours l'équation (2.9).

Propriété 2.5 *Pour un terme générateur α d'ordre M de $GF\,(q)$, tous les produits nk, exposants de α dans l'équation* (2.10)*, sont calculés avec une réduction modulo M :*

$$\left\langle \alpha^{nk} = \alpha^{\langle nk \rangle_M} \right\rangle_q \tag{2.12}$$

Cette relation est bien sûr valable dans l'équation (2.11) *où $\left\langle \alpha^{-nk} = \alpha^{\langle -nk \rangle_M} \right\rangle_q$. Une transformation inverse s'effectuera alors uniquement avec des multiplications de puissances positives de α modulo q.*

Cette réécriture fait ressortir une plus grande symétrie des matrices de transformations et permet une simplification de leur construction.

Exemple 2.1

Prenons l'écriture de la matrice de transformation directe de dimension $M = 4$:

$$\left[\alpha^{nk}\right] \bmod q = \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 \end{bmatrix} \bmod q = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha^1 & \alpha^2 & \alpha^3 \\ 1 & \alpha^2 & 1 & \alpha^2 \\ 1 & \alpha^3 & \alpha^2 & \alpha^1 \end{bmatrix} \bmod q$$

Notons que seules deux opérations sont nécessaires pour définir cette matrice.

L'ensemble des propriétés, généralement associées aux transformées de Fourier discrètes existent aussi pour toutes les transformées en nombres entiers. En considérant une séquence $\{x_n\}$ et sa transformation en nombres entiers $\{X_k\}$, différentes propriétés [Agarwal1974] sont énoncées ci-dessous, de manière non-exhaustive.

Propriété 2.6 : ***linéarité***

La transformée en nombres entiers (définie sur un ensemble $\mathbb{Z}$ d'ordre q) de la somme de deux séquences $\{x_1\}$ et $\{x_2\}$ est égale à la somme sur $\mathbb{Z}_q$ de leurs transformées

$$T(x_1 + x_2) = \langle T(x_1) + T(x_2) \rangle_q$$

où T représente une transformation en nombres entiers directe ou inverse

Propriété 2.7 : ***périodicité***

Si la séquence $\{x_n\}$ est périodique et de période m, telle que $x_n = x_{n+m}$, alors sa transformée en nombres entiers $\{X_k\}$ est aussi périodique et de même période, c'est à dire $X_k = X_{k+m}$.

Propriété 2.8 : ***symétrie et antisymétrie***

• *Si la séquence $\{x_n\}$ est symétrique : $x_{m-n} = x_{m+n}$ alors sa transformée en nombres entiers $\{X_k\}$ est aussi symétrique : $X_{m-k} = X_{m+k}$*

• *Si la séquence $\{x_n\}$ est antisymétrique : $x_{m-n} = -x_{m+n}$ alors sa transformée en nombres entiers $\{X_k\}$ est aussi antisymétrique : $X_{m-k} = -X_{m+k}$*

Propriété 2.9 : ***orthogonalité***

L'élément générateur α forme un ensemble orthogonal :

$$\sum_{k=0}^{M-1} \alpha^{nk}\alpha^{-mk} = \sum_{k=0}^{M-1} \alpha^{(n-m)k} = \begin{cases} M & si \quad \langle n = m \rangle_M \\ 0 & sinon \end{cases}$$

Propriété 2.10 : ***décalage***

Si la séquence $\{x_n\}$ *est soumise à un décalage de* m *échantillons alors sa NTT est donnée par* :

$$T\left(\{x_{n+m}\}\right) = \left\{T\left(\{x_n\}\right)\alpha^{-mk}\right\} = \left\{X_k\alpha^{-mk}\right\} \tag{2.13}$$

où T *représente une transformation en nombres entiers directe ou inverse.*

Précisons avant d'énoncer la dernière propriété qu'un calcul de convolution cyclique de deux fonctions quelconques g et h, définies dans l'ensemble $\mathbb{R}$ ou $\mathbb{Z}$, correspond au calcul d'une intégrale qui exprime leur quantité de chevauchement en fonction d'un décalage de temps $(t-\tau)$:

$$(g * h)(t) = \int_{-\infty}^{+\infty} g(\tau)\, h(t-\tau)\, d\tau \tag{2.14}$$

où le symbole * représente l'opération de convolution qui correspond à un filtrage de g par la réponse impulsionnelle h ou inversement. Dans notre cas où les fonctions g et h seront deux séquences discrètes, d'une durée finie, le produit de convolution s'exprimera sous la forme d'une somme :

$$(g * h)(n) = \sum_{k=0}^{M-1} g(k)\, h(n-k) = \sum_{k=0}^{M-1} g(n-k)\, h(k) \tag{2.15}$$

avec $k = 0, 1, \ldots, M-1$ où M représente la longueur des deux séquences g et h.

Nous pouvons maintenant indiquer que les transformées en nombres entiers possèdent la Propriété de Convolution Cyclique (*CCP* : *Cyclic Convolution Property*) qui définit la transformée d'une convolution de deux séquences comme étant égale au produit de leur transformée.

Propriété 2.11 : ***convolution cyclique***

Soit la transformation en nombres entiers T *et son inverse* T^{-1}, *la convolution de deux séquences de nombres entiers* $\{g(n)\}_{n=0}^{M-1}$ *et* $\{h(n)\}_{n=0}^{M-1}$ *s'écrit* :

$$\{y(n)\} = (g * h)(n) = T^{-1}\left(T\left(g(n)\right) \bullet T\left(h(n)\right)\right) \tag{2.16}$$

où $n = 0, 1, \ldots, M-1$ *et l'opérateur* $\bullet$ *désigne ici une multiplication élément par élément.*

Démonstration 2.3

Les deux séquences $\{g(n)\}_{n=0}^{M-1}$ et $\{h(n)\}_{n=0}^{M-1}$ peuvent être exprimées au moyen de leurs transformées en nombres entiers, directe et inverse, définies sur l'ensemble $\mathbb{Z}_q$, par :

$$\begin{cases} g(n) = \left\langle M^{-1} \sum_{k=0}^{M-1} \left(\sum_{k=0}^{M-1} g(n) \alpha^{nk} \right) \alpha^{-nk} \right\rangle_q = \left\langle M^{-1} \sum_{k=0}^{M-1} G(k) \alpha^{-nk} \right\rangle_q \\ h(n) = \left\langle M^{-1} \sum_{k=0}^{M-1} \left(\sum_{k=0}^{M-1} h(n) \alpha^{nk} \right) \alpha^{-nk} \right\rangle_q = \left\langle M^{-1} \sum_{k=0}^{M-1} H(k) \alpha^{-nk} \right\rangle_q \end{cases}$$

D'après l'équation (2.15), la convolution cyclique s'écrit :

$$y(n) = (g * h)(n) = \left\langle M^{-1} \sum_{\tau=0}^{M-1} g(\tau) \sum_{k=0}^{M-1} H(k) \alpha^{-k(n-\tau)} \right\rangle_q$$

En inversant l'écriture, on obtient :

$$y(n) = \left\langle M^{-1} \sum_{k=0}^{M-1} H(k) \left(\sum_{\tau=0}^{M-1} g(\tau) \alpha^{k\tau} \right) \alpha^{-kn} \right\rangle_q = \left\langle M^{-1} \sum_{k=0}^{M-1} H(k) G(k) \alpha^{-kn} \right\rangle_q$$

Cela permet bien de retrouver la relation donnée par l'équation (2.16). La validité de cette propriété sera garantie par la relation (2.9) qui lie les trois paramètres α, M et q.

Notons ici que la relation de congruence $\langle -a \equiv q - a \rangle_q$, valable dans un ensemble d'entiers $\mathbb{Z}$ d'ordre q, indique que tout entier négatif $(-a)$ est représenté par un entier positif. Ainsi, pour éviter une double interprétation des entiers dans le résultat du calcul de convolution cyclique, donné par l'équation (2.16), les composantes $y(n)$ devront respecter la borne maximale$\mid y(n) \mid \leq \frac{q}{2}$. Pour pallier tout dépassement ou *overflow* dans le calcul de $\{y(n)\}_{n=0}^{M-1}$, des limites sur les amplitudes des éléments $g(n)$ et $h(n)$ doivent être fixées. S'il n'y a aucune connaissance à priori sur les différentes valeurs $g(n)$ et $h(n)$, alors les éléments $y(n)$ devront être bornés par :

$$\mid y(n) \mid_{max} \leq M \mid g(n) \mid_{max} \mid h(n) \mid_{max} \tag{2.17}$$

Si au moins l'une des deux séquences est connue, les bornes des $y(n)$ pourront être affinées telles que :

$$\mid y(n) \mid \leq \mid g(n) \mid_{max} \sum_{n=0}^{M-1} \mid h(n) \mid \leq \frac{q}{2} \tag{2.18}$$

Il faudra tenir compte de cette contrainte à chaque calcul de convolution en fonction des résultats a priori. La contrainte pourra être assouplie si le domaine de définition des $y(n)$ est connu par avance.

De nombreuses transformées en nombres entiers existent. Certaines d'entres elles acceptent des structures rapides similaires à celles mises en oeuvre pour la FFT. Nous allons introduire plus particulièrement la transformée en nombres de Fermat (FNT) qui semble être la transformée la plus adaptée aux opérations de

traitement numérique. Ainsi, son application à certains algorithmes laisse entrevoir une réduction importante de la complexité de calcul du système d'annulation d'écho acoustique.

2.2.3 Transformée en Nombres de Fermat (FNT)

Dans cette partie, les transformées en nombres entiers sont définies dans un ensemble dont l'ordre est de la forme $2^b + 1$ avec $b \in \mathbb{N}$. Si b est une puissance de 2, alors il appartient à l'ensemble des nombres de Fermat. Les nombres de Fermat s'écrivent donc sous la forme $F_t = 2^{2^t} + 1$ avec $t \in \mathbb{N}$ et offrent de nombreuses possibilités pour les valeurs de M.

La transformée en nombres entiers définie dans le corps de Galois $GF\left(F_t\right)$, est appelée transformée en nombres de Fermat (FNT : $Fermat\ Number\ Transform$). Le plus grand avantage d'une FNT est sa capacité à proposer, pour des F_t premiers, des longueurs M de transformée très composées vérifiant $M \mid \left(F_t - 1\right)$. Notons cependant que seuls les cinq premiers nombres de Fermat sont premiers :

$$\begin{cases} F_0 = 2^1 + 1 = 3 \\ F_1 = 2^2 + 1 = 5 \\ F_2 = 2^4 + 1 = 17 \end{cases} \quad \begin{cases} F_3 = 2^8 + 1 = 257 \\ F_4 = 2^{16} + 1 = 65537 \end{cases}$$

Les suivants étant considérés comme pseudo-premiers, pourront toutefois être utilisés pour le calcul de transformées [Rosen1993], mais la longueur M de ces transformées devra diviser le plus grand commun diviseur de leurs facteurs diminués de l'unité.

$$\begin{cases} F_5 = 2^{32} + 1 = 4294967297 = 641 \times 6700417 \\ F_6 = 2^{64} + 1 = 274177 \times 67280421310721 \end{cases}$$

Une transformée en nombres de Fermat de longueur quelconque $M = 2^{t+1}$ pourra être calculée en utilisant seulement $\frac{M}{2} log_2 M$ multiplications et $M log_2 M$ additions modulo F_t, pour une racine primitive $\alpha = 2$ qui vérifie la relation suivante :

$$\left\langle 1 = (-1)^2 \equiv \left(F_t - 1\right)^2 = \left(2^{2^t}\right)^2 = 2^{2^{t+1}} \right\rangle_{F_t} \tag{2.19}$$

Le terme générateur α étant essentiel pour réduire la complexité de calcul d'une FNT, le choix de α différent d'une racine primitive mais égal à une puissance de deux reste intéressant en arithmétique binaire car les multiplications par une puissance de 2 seront réalisées par des décalages de bits. Les longueurs possibles M de transformées et les termes générateurs α associés sont alors respectivement donnés par $M = 2^{t+1-i}$ et $\left\langle \alpha = 2^{2^i} \right\rangle_{F_t}$ avec $(i,t) \in \mathbb{N}^2$ tels que $0 \leq i < t$ [Julien1991]. Les transformées en nombres de Fermat, directe et inverse, peuvent donc s'écrire :

$$X_k = \left\langle \sum_{n=0}^{2^{t+1-i}-1} x_n 2^{\langle 2^i nk \rangle_M} \right\rangle_{F_t} \tag{2.20}$$

$$x_n = \left\langle -2^{2^t-1-(t-i)} \sum_{k=0}^{2^{t+1-i}-1} X_k 2^{\langle -2^i nk \rangle_M} \right\rangle_{F_t} \tag{2.21}$$

avec k, $n = 0, 1, \ldots, 2^{t+1-i} - 1$. Dans l'équation (2.21) de la transformée inverse, la longueur de séquence M étant égale à une puissance de 2, son inverse est donné par $\left\langle M^{-1} = -2^{2^t-t+i-1} \right\rangle_{F_t}$, en effet :

$$\begin{cases} \left\langle F_t - 1 = 2^{2^t} \equiv -1 \right\rangle_{F_t} \\ \left\langle 2^{2^t} 2^{-(t-i+1)} \equiv -2^{-(t-i+1)} \right\rangle_{F_t} \\ \left\langle M^{-1} = 2^{-(t+1-i)} \equiv -2^{2^t-t+i-1} \right\rangle_{F_t} \end{cases}$$

Agarwal et *Burrus* [Agarwal1974] ont montré qu'il était possible, en conservant une valeur de α égale à une puissance de 2, de doubler la longueur de transformée $M = 2^{t+2}$ en posant $i = -1$. Le terme générateur α est alors défini comme étant la solution de la congruence suivante :

$$\left\langle \alpha^2 \equiv 2 \right\rangle_{F_t} \tag{2.22}$$

Une valeur de α existe pour tout nombre de Fermat F_t avec $t \geq 2$ et peut être exprimée telle que $\left\langle \sqrt{2} = 2^{2^{t-2}} \left(2^{2^{t-1}} - 1 \right) \right\rangle_{F_t}$ à partir de la démarche suivante :

$$\begin{cases} \left(\sqrt{2}\right)^2 = 2 = (-1).2.(-1) \\ \left\langle \left(\sqrt{2}\right)^2 = -2.2^{2^t} = 2^{2^{t-1}} \left(1 - 1 - 2.2^{2^{t-1}} \right) \right\rangle_{F_t} \\ \left\langle \left(\sqrt{2}\right)^2 = \left(2^{2^{t-2}} \right)^2 \left(\left(2^{2^{t-1}} \right)^2 - 2.2^{2^{t-1}} + 1^2 \right) \right\rangle_{F_t} \\ \left\langle \left(\sqrt{2}\right)^2 = \left(2^{2^{t-2}} \left(2^{2^{t-1}} - 1 \right) \right)^2 \right\rangle_{F_t} \end{cases}$$

t	b	modulo F_t	M pour $\alpha = 4$	M pour $\alpha = 2$	M pour $\alpha = \sqrt{2}$	M_{max}	α pour M_{max}
0	1	$2^1 + 1$	–	2	–	2	2
1	2	$2^2 + 1$	2	4	–	2^2	2 ou 3
2	4	$2^4 + 1$	4	8	16	2^4	$\sqrt{2}$ ou 3
3	8	$2^8 + 1$	8	16	32	2^8	3
4	16	$2^{16} + 1$	16	32	64	2^{16}	3
5	32	$2^{32} + 1$	32	64	128	2^7	$\sqrt{2}$
6	64	$2^{64} + 1$	64	128	256	2^8	$\sqrt{2}$

TAB. 2.1 – Paramètres possibles pour l'implantation de la FNT

La notation $\sqrt{2}$ est bien entendue utilisée uniquement pour simplifier les écritures car la fonction racine

carrée n'a aucune signification dans un corps de Galois. Cette décomposition en puissance de 2 montre que les multiplications par une puissance n de $\sqrt{2}$ peuvent être réalisées par un décalage de bits si n est pair ou par 2 décalages et une addition si n est impair :

$$\begin{cases} \left\langle \left(\sqrt{2}\right)^n = 2^{\frac{n}{2}} \right\rangle_{F_t} & \text{si} \quad n \quad \text{pair} \\ \left\langle \left(\sqrt{2}\right)^n = 2^{\frac{n-1}{2}} 2^{2^{t-2}} \left(2^{2^{t-1}} - 1\right) \right\rangle_{F_t} & \text{si} \quad n \quad \text{impair} \end{cases}$$

Pour les FNT n'ayant pas atteint leur longueur de transformée maximale, nous avons vérifié, de manière analogue, que ce résultat peut être étendu aux valeurs de i négatives suivantes. Pour un modulo égal au quatrième ou cinquième nombre de Fermat $\left(\quad t = 3 \quad \text{ou} \quad 4 \quad \right)$, la longueur de séquence $M_{max} = 2^b$ pourra alors être obtenue pour la racine primitive $\alpha = 3$ du corps $GF\ (F_t)$ ou par la décomposition d'une puissance de 2 telle que $\left\langle \alpha = 2^{\frac{1}{2b}} = \sqrt[2b]{2} \right\rangle_{F_t}$ (voir table 2.2)

Proposition 2.4 *Le choix de* $M = 2^{t+1-i}$ *et de* α *tel que* $\left\langle \alpha^{2^{1-i}} = 2 \right\rangle_{F_t}$ *peut être associé à des valeurs de* i *négatives et entières, appartenant à l'ensemble* $\left\{ \quad -4, \quad -3, \quad -2 \quad \right\}$ *pour* $t \geq 3$ *et aux valeurs entières de l'intervalle compris entre* -12 *et* -5 *pour* $t = 4$.

t	b	modulo F_t	M pour $\alpha = \sqrt[4]{2}$	M pour $\alpha = \sqrt[8]{2}$	M pour $\alpha = \sqrt[16]{2}$	M_{max}	α pour M_{max}
3	8	$2^8 + 1$	64	128	256	2^8	$\sqrt[16]{2}$ ou3
4	16	$2^{16} + 1$	128	256	512	2^{16}	3

TAB. 2.2 – Autres paramètres possibles pour l'implantation de la FNT

Agarwal et *Burrus* [Agarwal1974] ont vérifié que la transformée en nombres de Fermat admet la Propriété de Convolution Cyclique et ont insisté sur le fait que le calcul d'une FNT requiert environ $Mlog_2M$ opérations simples (décalages de bits, additions) mais aucune multiplication, alors qu'une transformée de Fourier discrète nécessite un nombre de multiplications complexes de l'ordre de $\frac{M}{2} log_2 \frac{M}{2}$.

Exemple 2.2

Pour illustrer l'utilisation de la propriété de convolution cyclique (CCP) de la transformée en nombres de Fermat (FNT), un exemple est traité pour le calcul de produit de convolution entre deux séquences de longueur M, $f = \left[\quad 2, \quad -2, \quad 1, \quad 0 \quad \right]$ et $g = \left[\quad 1, \quad 2, \quad 0, \quad 0 \quad \right]$. En considérant le choix $t = 2$, $F_2 = 17$ et $M = 4$, nous appliquerons la racine $\alpha = 4$ qui répond à l'égalité $\left\langle 4^4 = 1 \right\rangle_{17}$. La matrice de transformation s'écrit alors :

$$\left\langle \left[\alpha^{nk}\right]\right\rangle_{17} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 4^2 & 4^3 \\ 1 & 4^2 & 4^4 & 4^6 \\ 1 & 4^3 & 4^6 & 4^9 \end{bmatrix} \; mod\, 17$$

Sachant que $\left\langle 2^{nk}\right\rangle_{17} = \left\langle 2^{\langle nk\rangle_4}\right\rangle_{17}$, la matrice de transformation est obtenue par :

$$\left\langle \left[\alpha^{nk}\right]\right\rangle_{17} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 4^2 & 4^3 \\ 1 & 4^2 & 1 & 4^2 \\ 1 & 4^3 & 4^2 & 4 \end{bmatrix} \; mod\, 17 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} \; mod\, 17$$

L'inverse de M dans $GF\ (17)$ étant égal à $\left\langle M^{-1} = 4^{-1} = 13\right\rangle_{17}$, la matrice de transformation inverse $\left\langle M^{-1}\left[\alpha^{-nk}\right]\right\rangle_{17}$ s'écrit :

$$\left\langle M^{-1}\left[\alpha^{-nk}\right]\right\rangle_{17} = 4^{-1}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4^{-1} & 4^{-2} & 4^{-3} \\ 1 & 4^{-2} & 4^{-4} & 4^{-6} \\ 1 & 4^{-3} & 4^{-6} & 4^{-9} \end{bmatrix} \; mod\, 17$$

$$= 13\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4^3 & 4^2 & 4 \\ 1 & 4^2 & 1 & 4^2 \\ 1 & 4 & 4^2 & 4^3 \end{bmatrix} \; mod\, 17$$

soit

$$\left\langle M^{-1}\left[\alpha^{-nk}\right]\right\rangle_{17} = 13\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 13 & 16 & 4 \\ 1 & 16 & 1 & 16 \\ 1 & 4 & 16 & 13 \end{bmatrix} \; mod\, 17$$

La transformée en nombres de Fermat de f et de g sont alors données respectivement par F et G à l'aide des matrices de transformation directe $\left\langle \left[\alpha^{nk}\right]\right\rangle_{17}$:

$$F = \left\langle \left[\alpha^{nk}\right] f^T \right\rangle_{17} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 1 \\ 0 \end{bmatrix} mod\, 17 = \begin{bmatrix} 1 \\ 10 \\ 5 \\ 9 \end{bmatrix} mod\, 17$$

$$G = \left\langle \left[\alpha^{nk}\right] g^T \right\rangle_{17} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} mod\, 17 = \begin{bmatrix} 3 \\ 9 \\ 16 \\ 10 \end{bmatrix} mod\, 17$$

Le produit de convolution $h = f * g$ étant égal à la transformée inverse de H, produit élément par élément de F et G :

$$H = \langle F \bullet G \rangle_{17} = \begin{bmatrix} 1 \\ 10 \\ 5 \\ 9 \end{bmatrix} \bullet \begin{bmatrix} 3 \\ 9 \\ 16 \\ 10 \end{bmatrix} mod\, 17 = \begin{bmatrix} 3 \\ 5 \\ 12 \\ 5 \end{bmatrix} mod\, 17$$

où l'opérateur $\bullet$ désigne la multiplication élément par élément. La convolution h est alors obtenue par :

$$h = \left\langle M^{-1} \left[\alpha^{-nk}\right] H \right\rangle_{17} = 13 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 13 & 16 & 4 \\ 1 & 16 & 1 & 16 \\ 1 & 4 & 16 & 13 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 12 \\ 5 \end{bmatrix} mod\, 17 = \begin{bmatrix} 2 \\ 2 \\ -3 \\ 2 \end{bmatrix} mod\, 17$$

Ce résultat est bien sûr identique à celui obtenu par le calcul de convolution conventionnel donné par la somme suivante :

$$h_k = \sum_{n=0}^{3} f_n g_{\langle k-n \rangle_M} \text{ avec } k = 0, 1, 2, 3. \tag{2.23}$$

Cet exemple illustre bien que les approximations telles que les troncatures ou les arrondis de nombres n'existent pas dans une arithmétique en nombres entiers. De plus, les séquences de départ satisfaisant la condition émise par l'équation (2.18), les différents dépassements observés durant les étapes intermédiaires du calcul n'ont pas d'effet sur le résultat final. Notons qu'une généralisation de la borne d'amplitude $\pm A$ des éléments de chaque séquence à convoluer, pour un modulo égal à un nombre de Fermat $F_t = 2^{2^t} + 1$ et une longueur de séquence $M = 2^b$, est donnée par $A \leq 2^{\frac{2^t - b - 1}{2}}$ [Alfredsson1996].

2.3 Principe d'implantation d'une FNT

Pour la fin de ce chapitre et les suivants, notre étude sera axée sur l'implantation et sur les applications liées aux transformées en nombres de Fermat. Le choix de la FNT s'explique par sa grande modularité et sa relative simplicité de réalisation. Les propriétés de la transformée en nombres de Fermat ont, en effet, permis d'envisager une exécution $VLSI$ ($Very\ Large\ Scale\ Integration$) des opérations arithmétiques, dans un ensemble de nombres entiers d'ordre égal à un nombre de Fermat, traditionnellement mises en application à l'aide des circuits logiques binaires.

De nombreuses manières de représenter les nombres entiers de l'ensemble $\mathbb{Z}_{2^b+1}$ par des mots binaires de $b+1$ bits existent. La complexité et les performances des architectures pour les opérations arithmétiques dépendront entre autres de la représentation choisie. Les représentations les plus connues sont celles proposées par *McClellan* [McClellan1976] et *Leibowitz* [Leibowitz1976] ($Diminished-1\ Representation$). En utilisant ces représentations, les opérations telles que l'addition, la multiplication par deux, peuvent être assez facilement effectuées en $VLSI$.

Notons que les circuits semi-conducteur $CMOS$ sont, aujourd'hui, les plus utilisés dans les architectures $VLSI$. Cette technologie attrayante permet une implantation dense, un rendement élevé, une faible dissipation de puissance et un bas coût [Weste1993].

2.3.1 Implantation Butterfly

De la même façon qu'il existe une version rapide de la transformée de Fourier (FFT : $Fast\ Fourier\ Transform$), il existe une version rapide de la transformée en nombres entiers. Les structures $VLSI$ de la FFT, de type $butterfly$ par exemple, peuvent alors être adoptées pour l'exécution en temps réel d'une transformée en nombres de Fermat rapide.

Comme mentionné précédemment, une transformée en nombres de Fermat de dimension $M = 2^n$ avec $0 \leq n \leq b = 2^t$ peut être calculée en utilisant le principe de l'algorithme FFT, basé sur la décomposition suivante [Cooley1965].

$$\left\langle X_k = \sum_{n=0}^{M-1} x_n \alpha^{nk} = \sum_{n=0}^{\frac{M}{2}-1} x_{2n} \alpha^{2nk} + \sum_{n=0}^{\frac{M}{2}-1} x_{2n+1} \alpha^{(2n+1)k} \right\rangle_{F_t} \tag{2.24}$$

$$\left\langle X_k = \sum_{n=0}^{\frac{M}{2}-1} x_{2n} \alpha^{2nk} + \alpha^k \sum_{n=0}^{\frac{M}{2}-1} x_{2n+1} \alpha^{2nk} = G_k + \alpha^k H_k \right\rangle_{F_t} \tag{2.25}$$

avec $k = 0, 1, \ldots, M-1$. Les vecteurs $\{G_k\}$ et $\{H_k\}$ sont respectivement les transformées en nombres de Fermat des séquences $\{x_{2n}\}_{n=0}^{\frac{M}{2}-1}$ et $\{x_{2n+1}\}_{n=0}^{\frac{M}{2}-1}$ de longueur $\frac{M}{2}$.

Sachant que $\left\langle \alpha^M = 1 \right\rangle_{F_t}$, la relation $\left\langle \alpha^{k+\frac{M}{2}} = -\alpha^k \right\rangle_{F_t}$ est établie de la manière suivante :

$$\begin{cases} \left\langle \alpha^M = 1 = (-1)^2 = \left(\alpha^{\frac{M}{2}}\right)^2 \right\rangle_{F_t} \\ \left\langle \alpha^{\frac{M}{2}} = -1 \right\rangle_{F_t} \\ \left\langle \alpha^{k+\frac{M}{2}} = -\alpha^k \right\rangle_{F_t} \end{cases}$$

La transformée en nombres de Fermat $\{X_k\}$ complète, de longueur M, peut alors être exprimée par :

$$\begin{cases} X_k = G_k + \alpha^k H_k \\ X_{k+\frac{M}{2}} = G_k - \alpha^k H_k \end{cases}$$

pour $k = 0, 1, \ldots, \frac{M}{2} - 1$. Ce découpage de la séquence en deux parties peut être répété pour décomposer la transformée de longueur $M = 2^b$ en $\left\{4, 8, \ldots, \frac{M}{2}\right\}$ parties. Toute FNT peut alors être calculée, à l'aide de $log_2\left(2^b\right)$ décompositions, comme $\frac{Mb}{2}$ transformées de longueur de séquence $M = 2$.

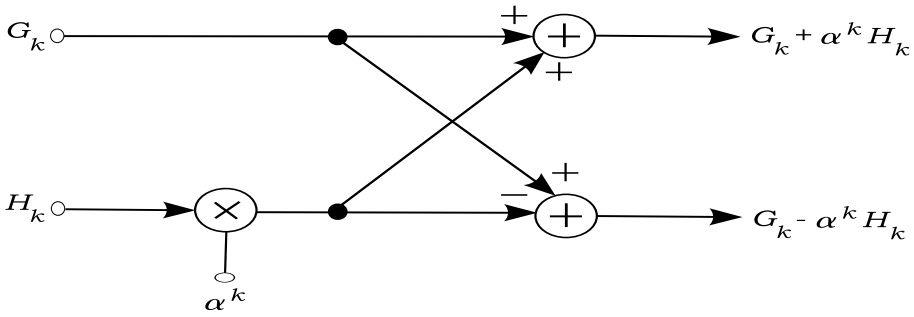

FIG. 2.1 – Décomposition Butterfly

La figure (2.1) en illustre l'implantation symétrique, appelée *Butterfly* ou en *papillon.* De cette structure, qui requiert deux additions et une multiplication, sont tirés le nombre total d'opérations requises pour une transformée en nombres de Fermat. Comme indiqué précédemment, les nombres d'additions et multiplications mis en oeuvre pour une transformée en nombres de Fermat sont respectivement de l'ordre de $Mlog_2M$ et $\frac{M}{2}log_2M$.

2.3.2 Implantation des opérations binaires sur un corps de Galois

Dans cette partie, des représentations d'opérations réduites par un modulo de la forme $2^b + 1$, par des fonctions binaires simples, sont présentées. Notons que dans l'ensemble $\mathbb{Z}_{2^b+1}$, les mots binaires sont représentés par une longueur fixe de $b + 1$ bits. Toutefois, il serait très intéressant de ne pouvoir utiliser qu'un processeur de b bits où seule la représentation de la valeur 2^b poserait problème. Ce nombre ayant une probabilité très faible d'apparaître (de l'ordre de 2^{-b}) pour des données décorrélées [Agarwal1974], il est possible de supprimer cette valeur en la remplaçant par un de ses voisins 0 ou $2^b - 1$. Bien sûr que les faibles erreurs occasionnées devront être préalablement jugées acceptables en fonction du type d'application mis en

oeuvre. Si ce n'est pas le cas, un bit supplémentaire sera nécessaire pour représenter 2^b ce qui compliquera l'intégration matérielle (*hardware*) sur une carte *DSP*.

Une représentation binaire dans $\mathbb{Z}_{2^b+1}$ des opérations de réduction par le modulo 2^b+1, de négation, d'addition ou de multiplications sont détaillées ci-après. Notons que certaines de ces opérations seront bien évidemment impliquées dans le calcul des transformées en nombres de Fermat directe et inverse. Pour la suite de cette partie, posons un nombre entier $u = (u_{b-1}\, u_{b-2}\, \ldots\, u_1\, u_0)_2$ représenté par un mot de b bits tel que $u = \sum_{n=0}^{b-1} u_n 2^n$ où $u_n \in \mathbb{Z}_2 = \{0, 1\}$. Dans un souci de clarté, les opérations binaires seront à chaque fois illustrées dans $\mathbb{Z}_{2^4+1}$, avec des entiers codées sur 4 bits ; l'extension des résultats au cas général de $\mathbb{Z}_{2^b+1}$ est évidente.

2.3.2.1 La réduction modulo $2^b + 1$

Toute opération dans un ensemble $\mathbb{Z}_{2^b+1}$ étant suivie d'une réduction par le modulo 2^b+1, il est alors très important que cette procédure soit la plus simple et rapide possible. Pour certaines opérations, la réduction du modulo pourra même être intégrée directement dans le calcul.

Proposition 2.5 *Sachant que* $\langle 2^b \equiv -1 \rangle_{2^b+1}$, *le résidu modulo* 2^b+1 *d'un entier* $u \geq 2^b$, *écrit sur* $(b+1)$ *bits est obtenu en retranchant la retenue* $u_b = 1$ *au mot* $u = (u_{b-1}\, u_{b-2}\, \ldots\, u_1\, u_0)_2$.

$$\left\langle u = u_b 2^b + \sum_{n=0}^{b-1} u_n 2^n = \sum_{n=0}^{b-1} u_n 2^n - u_b 2^0 = \sum_{n=0}^{b-1} u_n 2^n - 1 \right\rangle_{2^b+1} \tag{2.26}$$

La soustraction de 1 *sera en fait réalisée par l'addition du complément à deux de un*

$$\left\langle u = \sum_{n=0}^{b-1} (u_n + 1)\, 2^n \right\rangle_{2^b+1} \tag{2.27}$$

Comme indiqué précédemment, dans le cas où $u = 2^b$, *nous pourrons aussi choisir d'arrondir* u *à* 0.

Exemple 2.3

Prenons l'exemple de l'entier $u = 26$ à réduire par un modulo égal au deuxième nombre de Fermat $2^{2^2} + 1 = 17$, avec $b = 4$. La représentation binaire de l'entier $u = 26$ sur $b + 1 = 5$ bits est donnée par :

$$26 = \{11010\}_2$$

La réduction modulo 17 de cet entier, en ne tenant compte que des $b = 4$ premiers bits, est réalisée par :

$$\left\langle 26 = \left\{ \begin{array}{c} 1010 \\ -0001 \end{array} \right\}_2 = \left\{ \begin{array}{c} 1010 \\ +1111 \end{array} \right\}_2 = \{1001\}_2 = 9 \right\rangle_{17}$$

2.3.2.2 La négation

Pour effectuer l'opération de changement de signe dans un ensemble $\mathbb{Z}_{2^b+1}$, la démarche est basée sur l'écriture en mode binaire de tout nombre entier u négatif sous la forme :

$$u = -\sum_{n=0}^{b-1} u_n 2^n = \sum_{n=0}^{b-1} \overline{u_n} 2^n - \left(2^b - 1\right) \tag{2.28}$$

où $\overline{u_n}$ représente le complément du bit u_n du mot u.

Proposition 2.6 *Sachant que* $\left\langle 2^b \equiv -1 \right\rangle_{2^b+1}$, *l'opération de négation de tout entier, appartenant à un ensemble d'entiers d'ordre égal à* $2^b + 1$, *peut être réalisé par l'équation suivante* :

$$\left\langle u = -\sum_{n=0}^{b-1} u_n 2^n = \sum_{n=0}^{b-1} \overline{u_n} 2^n - \left(2^b - 1\right) = \sum_{n=0}^{b-1} \overline{u}_n 2^n + 2 \right\rangle_{2^b+1} \tag{2.29}$$

Exemple 2.4

Donnons ici l'exemple d'un entier $u = 4$ modulo 17 dont le signe veut être inversé :

$$\left\langle -4 = 17 - 4 = \left\{ \begin{array}{c} 1011 \\ +0010 \end{array} \right\}_2 = \{1101\}_2 = 13 \right\rangle_{17}$$

2.3.2.3 L'addition et la soustraction

Considérons maintenant l'addition $\langle u + v \rangle_{2^b+1}$ où les entiers u et v, écrits sur b bits, appartiennent à $\mathbb{Z}_{2^b+1}$, soit $0 \leq u,\, v < 2^b$:

$$w = u + v = \sum_{n=0}^{b-1} u_n 2^n + \sum_{n=0}^{b-1} v_n 2^n \tag{2.30}$$

Proposition 2.7 *Si* $w < 2^b$, *alors* w *est obtenu simplement par l'équation* (2.30). *Sinon l'addition sera suivie d'une réduction modulo* $2^b + 1$ *tel que* :

$$\left\langle w = \sum_{n=0}^{b-1} \left(u_n + v_n\right) 2^n - 1 \right\rangle_{2^b+1} \tag{2.31}$$

Proposition 2.8 *La soustraction* $\langle u - v \rangle_{2^b+1}$ *sera, elle, effectuée en deux temps. La négation de* v *est calculée et additionnée à* u *comme indiqué dans la proposition* 2.7.

2.3.2.4 La multiplication par une puissance de 2

Comme il a été déjà précisé précédemment, les multiplications par des puissances de deux sont particulièrement faciles à mettre en oeuvre en arithmétique binaire. En effet, la multiplication par 2 d'un nombre binaire $u = \sum_{n=0}^{b-1} u_n 2^n$ est réalisée par un décalage de bit vers la gauche, en ne tenant compte que des $(b-1)$ premiers bits :

$$2u = (u_{b-2}\, u_{b-3}\, \ldots\, u_1\, u_0\, 0)_2 \tag{2.32}$$

Proposition 2.9 *La multiplication par une puissance β de 2 dans l'ensemble $\mathbb{Z}_{2^b+1}$, avec $0 \leq 2^\beta u \leq 2^b - 1$, pourra donc être obtenue par β décalages de bits sur u :*

$$\left\langle u = \sum_{m=\beta}^{b-1} u_{(m-\beta)} 2^m \right\rangle_{2^b+1} \tag{2.33}$$

Si $2^\beta u \geq 2^b$, alors la procédure de réduction modulo $2^b + 1$ doit être effectuée. La retenue 1 sera alors soustraite pour maintenir l'opération à l'ensemble $\mathbb{Z}_{2^b+1}$.

Notons que le calcul d'une transformée inverse peut demander des multiplications du type $2^{-\beta}$. Dans ce cas, la puissance négative sera remplacée par la congruence $\left\langle 2^{-\beta} \equiv -2^{b-\beta} \right\rangle_{2^b+1}$. Un nombre de décalages égal à $b - \beta$ et une négation du résultat intermédiaire sont alors nécessaires pour exécuter l'opération.

Exemple 2.5

Deux possibilités existent pour le calcul binaire $\left\langle 2^\beta u \right\rangle_{2^b+1}$. Prenons par exemple l'opération $\left\langle 11.2^3 = 3 \right\rangle_{17}$, avec $u = 11$ et $\beta = 3$, qui peut être calculée par 2 méthodes :

• Soit chaque décalage est dissocié et la réduction par le modulo est effectuée dès que nécessaire :

$$\begin{cases} \left\langle 11.2 = 2.\{1011\}_2 = \{10110\}_2 = \left\{ \begin{array}{c} 0110 \\ -0001 \end{array} \right\}_2 = \left\{ \begin{array}{c} 0110 \\ +1111 \end{array} \right\}_2 = \{0101\}_2 \right\rangle_{17} \\ \left\langle 11.2^2 = 2.\{0101\}_2 = \{1010\}_2 \right\rangle_{17} \\ \left\langle 11.2^3 = 2.\{1010\}_2 = \{10100\}_2 = \left\{ \begin{array}{c} 0100 \\ -0001 \end{array} \right\}_2 = \left\{ \begin{array}{c} 0100 \\ +1111 \end{array} \right\}_2 = \{0011\}_2 = 3 \right\rangle_{17} \end{cases}$$

• Soit les décalages de $\beta = 3$ bits à gauche sont pris en compte de manière globale tel que :

$$\left\{ \left\langle 11.2^3 = 2^3.\{1011\}_2 = \{1011000\}_2 \right\rangle_{17} \right.$$

La réduction modulo 17 est donnée par :

$$\left\{ \left\langle 11.2^3 = \{1011000\}_2 = \left\{ \begin{array}{c} 1000 \\ -0101 \end{array} \right\}_2 = \left\{ \begin{array}{c} 1000 \\ +1011 \end{array} \right\}_2 = \{0011\}_2 = 3 \right\rangle_{17}$$

2.3.2.5 La multiplication

Pour optimiser les opérations de multiplications, un produit standard d'entiers quelconques $(u,\, v) \in \mathbb{Z}^2_{2^b+1}$ peut être exécuté comme une sommation de sous-produits.

Proposition 2.10 *L'opération de multiplication exécute* $\langle w = uv \rangle_{2^b+1}$ *avec* $u = \sum_{n=0}^{b-1} u_n 2^n$ *et* $v = \sum_{n=0}^{b-1} v_n 2^n$ *où* $(u_n,\, v_n) \in \mathbb{Z}_2^2$ *tel que* :

$$\langle w = uv \rangle_{2^b+1} = \left\langle \sum_{n=0}^{b-1} u_n \left(2^n v\right) = \sum_{n=0}^{b-1} u_n \left(2^n \sum_{m=0}^{b-1} v_m 2^m \right) \right\rangle_{2^b+1} \tag{2.34}$$

Si $w \geq 2^b$, *alors la procédure de réduction modulo* $2^b + 1$ *doit bien sûr être effectuée.*

Exemple 2.6

Prenons l'exemple d'une multiplication simple telle que $\langle w = 9 \times 13 = 15 \rangle_{17}$:

$$\left\{ \begin{array}{l} \left\langle 9 \times 13 = \{1001\}_2 \times \{1101\}_2 = 2^0 \{1101\}_2 + 2^3 \{1101\}_2 = \{1101\}_2 + \{1101000\}_2 \right\rangle_{17} \\ \left\langle 9 \times 13 = \{1101\}_2 + \left\{ \begin{array}{c} 1000 \\ -0110 \end{array} \right\}_2 = \{1101\}_2 + \left\{ \begin{array}{c} 1000 \\ +1010 \end{array} \right\}_2 \right\rangle_{17} \\ \langle 9 \times 13 = \{1101\}_2 + \{0010\}_2 = \{1111\}_2 = 15 \rangle_{17} \end{array} \right.$$

2.3.2.6 Le calcul de puissance

Pour calculer la puissance $\langle w = u^v \rangle_{2^b+1}$, avec $u = \sum_{n=0}^{b-1} u_n 2^n$ et $v = \sum_{n=0}^{r-1} v_n 2^n$ où $(u_n,\, v_n) \in \mathbb{Z}_2^2$, la méthode la plus utilisée est appelée *binary method* [Knuth1969] et permet d'écrire w tel que :

$$\left\langle w = u^v = \left(\left(\left(\left(u^{v_{r-1}} \right)^2 u^{v_{r-2}} \right)^2 \ldots u^{v_2} \right)^2 u^{v_1} \right)^2 u^{v_0} \right\rangle_{2^b+1} \tag{2.35}$$

La valeur de w sera donc obtenue au moyen des puissances de 2 et des multiplications. Dans le cas du calcul de puissance de la racine unité d'une transformée en nombres de Fermat, la volonté de choisir α égal à une puissance de 2 est clairement justifiable pour permettre le remplacement de toutes les multiplications mises en oeuvre par des décalages.

2.4 Application des transformées en nombres entiers

Une transformée en nombres entiers admet plusieurs propriétés intéressantes qui permettent de l'appliquer à des algorithmes rapides utilisés en traitement de signal. Les calculs très courants de convolution, de corrélation et de multiplication de matrices sont abordés. Notons que ces trois algorithmes sont valables quelle que soit la transformée utilisée $\left(\begin{array}{cc} FFT, & FNT \end{array} \right)$.

Nous présentons dans cette partie trois fonctions importantes pour lesquelles la mise en oeuvre de la transformée en nombres de Fermat (FNT) est très intéressante. Il est entendu que cette description n'est pas exhaustive, d'autres applications mettant en jeu la Propriété de Convolution Cyclique de la FNT existent, telles que la multiplication de grands entiers.

2.4.1 Convolution et corrélation de séquences

Les produits de convolution et de corrélation sont deux opérations essentielles en traitement de signal. La convolution de deux séquences $\{x_n\}_{n=0}^{M-1}$ et $\{h_n\}_{n=0}^{M-1}$ est donnée par :

$$y_k = \sum_{n=0}^{M-1} x_n h_{\langle k-n \rangle_M} \quad \text{avec} \quad k = 0, 1, \ldots, M-1 \tag{2.36}$$

Notons que la convolution et la corrélation sont mathématiquement équivalentes, la corrélation de $\{x_n\}_{n=0}^{M-1}$ et $\{h_n\}_{n=0}^{M-1}$ étant obtenue par la convolution de $\{x_n\}_{n=0}^{M-1}$ avec $\{h_{-n}\}_{n=0}^{M-1}$.

Comme précisé dans la partie précédente de ce chapitre, les transformées en nombres entiers possèdent la Propriété de Convolution Cyclique (CCP) [Agarwal1975]. Cette méthode de calcul de convolution, particulièrement efficace pour des longueurs de séquences composées, est souvent plus rapide que celle du calcul direct de l'équation (2.36), ce qui lui vaut le nom de *convolution rapide*. Néanmoins, il existe des algorithmes permettant d'améliorer encore cette procédure, tels que la méthode *over-lap add* qui permet de sectionner une longue séquence pour calculer plus facilement la convolution sur des sous-séquences plus courtes [Rabiner1975].

2.4.1.1 Calcul de convolution rapide

Le calcul de convolution de deux séquences de M échantillons, en utilisant la Propriété de Convolution Cyclique, exige l'application de trois transformées en nombre entiers de longueur $2M$, les séquences étant étendues par des zéros. Pour éviter l'addition de zéros, un algorithme est présenté ci-dessous [Shu1988]. Pour calculer la convolution de deux séquences de longueur M, $\{x(n)\}_{n=0}^{M-1}$ et $\{h(n)\}_{n=0}^{M-1}$, la procédure, donnée par la figure (2.2), est appliquée.

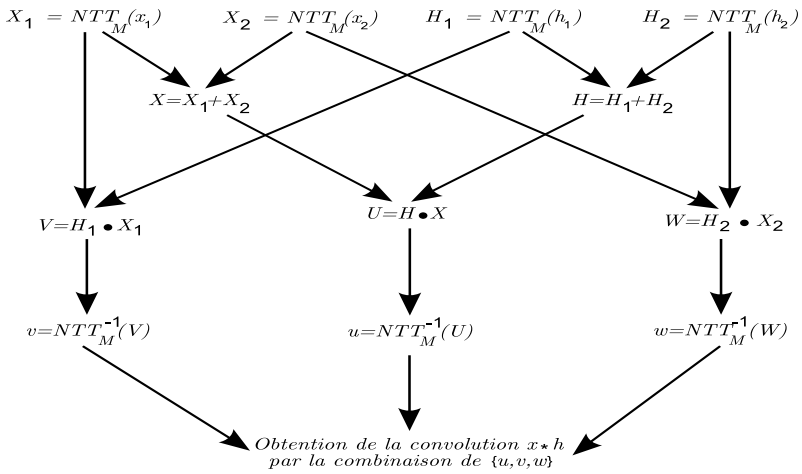

FIG. 2.2 – Procédure de convolution rapide

Algorithme 2.1

Deux séquences $\{x_1\}$ et $\{x_2\}$ sont formées à partir de $\{x(n)\}_{n=0}^{M-1}$:

$$\begin{cases} x_1(n) = \begin{cases} x(n) & 0 \le n < \frac{M}{2} \\ 0 & \frac{M}{2} \le n < M \end{cases} \\ x_2(n) = \begin{cases} 0 & 0 \le n < \frac{M}{2} \\ x(n) & \frac{M}{2} \le n < M \end{cases} \end{cases}$$

Deux autres vecteurs, notés $\{h_1(n)\}_{n=0}^{M-1}$ et $\{h_2(n)\}_{n=0}^{M-1}$, sont définis de la même manière à partir de $\{h(n)\}_{n=0}^{M-1}$. En posant $\{X_1\}$, $\{X_2\}$, $\{H_1\}$ et $\{H_2\}$ les transformées en nombres entiers respectives de $\{x_1\}$, $\{x_2\}$, $\{h_1\}$ et $\{h_2\}$ définies sur $\mathbb{Z}_q$, alors les transformées de $\{x\}$ et $\{h\}$ sont données par :

$$\begin{cases} X(k) = \langle X_1(k) + X_2(k) \rangle_q \\ H(k) = \langle H_1(k) + H_2(k) \rangle_q \end{cases}$$

avec $k = 0, 1, \ldots, M-1$.

Trois nouvelles séquences, notées $\{U\}$, $\{V\}$ et $\{W\}$, peuvent maintenant être calculées par :

$$\begin{cases} U(k) = \langle H(k) \bullet X(k) \rangle_q \\ V(k) = \langle H_1(k) \bullet X_1(k) \rangle_q \\ W(k) = \langle H_2(k) \bullet X_2(k) \rangle_q \end{cases}$$

où $k = 0, 1, \ldots, M-1$ et l'opérateur $\bullet$ désigne des multiplications élément par élément. La convolution $y = x * h$ est alors obtenue par transformations inverses de $\{U\}$, $\{V\}$ et $\{W\}$:

$$\begin{cases} y(n) = v(n) & 0 \leq n < \frac{M}{2} \\ y(n) = \langle u(n) - w(n) \rangle_q & \frac{M}{2} \leq n < M \\ y(n) = \langle u(n-M) - v(n-M) \rangle_q & M \leq n < \frac{3M}{2} \\ y(n) = w(n) & \frac{3M}{2} \leq n < 2M \end{cases}$$

Par rapport à un calcul de produit de convolution par l'équation (2.16) de la CCP qui requiert trois transformées en nombres entiers (NTT) de longueur de séquences $2M$, cet algorithme fera appel à sept transformées de dimension M. Cette procédure est alors très avantageuse pour une implémentation simple en évitant la manipulation de longues séquences de données. Elle l'est d'autant plus que certaines NTT, telle que la FNT, sont implantées sans opération complexe ni multiplication.

2.4.1.2 Algorithme rapide d'autocorrélation

La méthode classique pour calculer une corrélation entre deux signaux utilise la Propriété de Convolution Cyclique. Une autocorrélation d'une séquence de M échantillons génère $2M-1$ échantillons en sortie et requiert la mise en oeuvre de transformées de dimension $2M$. Cependant, comme l'algorithme précédent, une méthode de calcul rapide de l'autocorrélation existe et permet la suppression de l'extension de séquences par des zéros. Le coût de calcul en sera particulièrement réduit. Ci-dessous nous décrivons un algorithme efficace [Xu1992], qui permet la segmentation des séquences de données d'entrée.

Algorithme 2.2

Pour calculer l'autocorrélation r_{xx}, donnée par l'équation (2.37) d'une séquence $\{x(j)\}_{j=0}^{M-1}$ pour un retard maximal p, la méthode présentée est basée sur une segmentation non-uniforme de la séquence d'origine et calcule plusieurs autocorrélations de dimension réduite n_i.

$$r_{xx}(k) = \sum_{j=0}^{M-1} x(j)\, x(k+j) \tag{2.37}$$

Une étape importante pour l'efficacité de cet algorithme est le choix des longueurs n_i, qui devront permettre la mise en oeuvre de transformées en nombres entiers, telles que :

$$M \leq \sum_{i=1}^{i_{max}} (n_i - p) \text{ avec } 0 < n_i < n_{i+1} \tag{2.38}$$

Notons que le choix correspondant à $n_i = n_{i+1}$ reste possible si $n_{i+1} = n_{i_{max}}$. Nous pouvons ensuite former $2i_{max}$ nouvelles séquences $\{x_{i,1}\}$ et $\{x_{i,2}\}$, pour $i = 1, \ldots, i_{max}$, telles que :

$$x_{i,2}(j) = \begin{cases} x(j+d_i) & \text{pour } 0 \leq j < n_i - p \\ 0 & \text{pour } \quad n_i - p \leq j < n_i \end{cases} \tag{2.39}$$

$$\text{et} \begin{cases} \text{si } i \neq i_{max} : \; x_{i,1}(j) = x(j+d_i) & \text{pour } 0 \leq j < n_i \\ \text{si } i = i_{max} : \; x_{i,1}(j) = x_{i,2}(j) & \text{pour } \quad 0 \leq j < n_i \end{cases} \tag{2.40}$$

où $d_1 = 0$ et $d_i = \sum_{k=1}^{i-1}(n_k - p)$ pour $i \in [2, i_{max}]$. Les transformées en nombres entiers peuvent maintenant être appliquées à la nouvelle écriture de la corrélation r_{xx}.

$$r_{xx}(k) = \sum_{i=1}^{i_{max}} \sum_{j=0}^{n_i - 1} x_{i,1}(j)\, x_{i,2}(k+j) \;\; \text{avec } 0 \leq k \leq p-1 \tag{2.41}$$

Pour $i = 1, \ldots, i_{max}$, les transformées respectives $\{X_{i,1}\}$ et $\{X_{i,2}\}$ des séquences $\{x_{i,1}\}$ et $\{x_{i,2}\}$ sont calculées pour un modulo q. Ensuite, la multiplication élément par élément est effectuée sur l'ensemble d'entiers $\mathbb{Z}_q$:

$$Y_i(j) = \langle X_{i,1}(j) \bullet X_{i,2}(n_i - j) \rangle_q \tag{2.42}$$

où $X_{i,1}(n_i) = X_{i,2}(0)$ avec $j = 0, \ldots, n_i - 1$. Par la suite, la transformée inverse $\{y_i\}$ de $\{Y_i\}$ offre la possibilité d'obtenir les coefficients d'autocorrélation à l'ordre p, qui seront égaux à :

$$r_{xx}(k) = \left\langle \sum_{i=1}^{i_{max}} y_i(k) \right\rangle_q \;\; \text{avec } k = 0, 1, \ldots, p-1 \tag{2.43}$$

En comparant cette méthode à celle utilisant la Propriété de Convolution Cyclique, les deux mettant en oeuvre la transformée en nombres entiers, ce calcul d'autocorrélation requiert deux fois moins de multiplications, de l'ordre de $\sum_{i=1}^{i_{max}} n_i$.

2.4.2 Multiplication de matrices

Une autre application d'importance possible des transformées en nombres de Fermat est la multiplication de deux matrices. En effet, une des méthodes de calcul rapide de produit de matrices [Yagle1995] consiste à réécrire la multiplication matricielle en produit de convolution de séquences composées par les coefficients des matrices.

Algorithme 2.3

Le produit $C(c_{ij}) = A(a_{ij})\, B(b_{ij})$ de deux matrices A et B de dimension $(M \times M)$ est écrit sous forme polynomiale $R(x) = P(x)\, Q(x)$, les coefficients des polynômes étant donnés par les éléments a_{ij} et b_{ij} des

matrices :

$$\begin{cases} p_{i+jM} = a_{ij} \\ q_{M(M-1-i+jM)} = b_{ij} \end{cases}$$

avec $0 \leq i,\ j < M$. Les coefficients q_n, avec $0 \leq n < M^3 - M$, non définis par les éléments b_{ij} de la matrice B étant égaux à zéro, les polynômes $P(x)$ et $Q(x)$ sont alors définis par :

$$P(x) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} p_{i+jM} x^{i+jM} \tag{2.44}$$

$$Q(x) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} q_{M(M-1-i+jM)} x^{M(M-1-i+jM)} \tag{2.45}$$

Ainsi, les coefficients r_k du polynôme $R(x)$ sont obtenus par la multiplication polynomiale suivante :

$$R(x) = P(x)\,Q(x) = \sum_{k=0}^{M^3+M^2-M-1} r_k x^k \tag{2.46}$$

La construction de la matrice C est alors possible en choisissant ses éléments c_{ij} tels que :

$$c_{ij} = r_{M^2-M+i+jM^2} \tag{2.47}$$

où $0 \leq i, j < M$ et r_k les éléments du vecteur obtenu soit par l'équation (2.46), soit par un produit de convolution. En effet, une multiplication polynomiale étant équivalente à un produit de convolution, $R(x)$ pourra être déterminé en effectuant la convolution de deux vecteurs contenant les coefficients des polynômes $P(x)$ et $Q(x)$.

En utilisant, pour le calcul de la convolution, des transformées en nombres entiers de longueur M^3+M^2-M et l'équation (2.16) de la propriété de convolution cyclique, le produit de deux matrices $(M \times M)$ peut s'effectuer avec moins de M^3 multiplications. Notons que cet algorithme de multiplication matricielle pourra être grandement simplifié dans le cas de matrices diagonales ou symétriques et plus encore avec l'utilisation de la matrice de *Toeplitz*.

2.4.2.1 Exemple de produit matriciel

Soit deux matrices de dimension (2×2), $A = \begin{bmatrix} 2 & 3 \\ 1 & 5 \end{bmatrix}$ et $B = \begin{bmatrix} 4 & 7 \\ 8 & 10 \end{bmatrix}$, dont nous souhaitons calculer le produit $C = AB$ à l'aide de l'algorithme précédent. En écrivant les deux matrices sous forme polynomiale, nous déterminons les $P(x)$ et $Q(x)$ suivants :

$$\begin{cases} P(x) = 2 + x + 3x^2 + 5x^3 \\ Q(x) = 8 + 4x^2 + 10x^4 + 7x^6 \end{cases}$$

Le produit $R(x)$ peut alors s'obtenir par la convolution $R = P * Q$ des deux séquences P et Q, définies par les coefficients des deux polynômes précédents, telles que :

$$\begin{cases} P = \begin{bmatrix} 2 & 1 & 3 & 5 \end{bmatrix} \\ Q = \begin{bmatrix} 8 & 0 & 4 & 0 & 10 & 0 & 7 \end{bmatrix} \end{cases}$$

Le produit de convolution R, calculé à l'aide de transformées en nombres entiers et de l'équation (2.16) de la propriété de convolution cyclique, fournit alors la séquence suivante :

$$R = \begin{bmatrix} 16 & 8 & \mathbf{32} & \mathbf{44} & 32 & 30 & \mathbf{44} & \mathbf{57} & 21 & 35 \end{bmatrix}$$

Le résultat pouvant être vérifié par le produit $R(x) = P(x) \times Q(x)$:

$$R(x) = 16 + 8x + \mathbf{32}x^2 + \mathbf{44}x^3 + 32x^4 + 30x^5 + \mathbf{44}x^6 + \mathbf{57}x^7 + 21x^8 + 35x^9$$

A partir du vecteur R, la matrice C résultante du produit matriciel peut donc être déduite : $C = \begin{bmatrix} 32 & 44 \\ 44 & 57 \end{bmatrix}$. Ce résultat étant bien évidemment vérifiable par :

$$\begin{bmatrix} 2 & 3 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} 4 & 7 \\ 8 & 10 \end{bmatrix} = \begin{bmatrix} 2 \times 4 + 3 \times 8 & 2 \times 7 + 3 \times 10 \\ 1 \times 4 + 5 \times 8 & 1 \times 7 + 5 \times 10 \end{bmatrix} = \begin{bmatrix} 32 & 44 \\ 44 & 57 \end{bmatrix}$$

Précisons que les transformées en nombres entiers utilisées dans cet exemple sont des transformées en nombres de Fermat de longueur $M = 16$ échantillons et de terme générateur $\alpha = 2$ pour un modulo $F_3 = 257$. L'ordre du corps de Galois $GF(F_3)$ a été choisi de manière à éviter tout dépassement dans le calcul de convolution.

2.5 Conclusion

Nous allons résumer les points abordés de ce chapitre qui sont essentiels à l'application souhaitée de la transformée en nombres de Fermat. Pour maintenir une bonne précision des calculs, toutes les opérations arithmétiques seront réalisées sur des mots de 16 bits. Les calculs des transformées en nombres de Fermat intervenant dans le système d'annulation d'écho acoustique seront alors effectués dans le corps de Galois $GF\ (F_4)$ d'ordre égal au cinquième nombre de Fermat $F_4 = 2^{16} + 1 = 65537$.

Le terme générateur α étant essentiel pour réduire la complexité de calcul d'une FNT, nous nous sommes imposés un choix de α égal à une puissance de deux. Ainsi, les multiplications par α, en arithmétique binaire, seront réalisées par des décalages de bits. Pour un α et un modulo F_4 fixés, la longueur M doit respecter la condition pgcd $(\alpha,\, q)$ = pgcd $(M,\, q) = 1$ nécessaire à l'existence dans le corps de Galois $GF\ (F_4)$ de la transformée en nombres de Fermat et de son inverse. Le modulo F_4 étant premier, les longueurs possibles de transformées doivent diviser la valeur maximale $M_{max} = 2^b = 2^{16} = 65536$ égale à $F_4 - 1$. Ainsi, les longueurs de toutes les FNT utilisées sont de la forme $M = 2^{t+1-i}$. Elles sont associées au terme générateur $\alpha = 2^{2^i}$ et limitées aux entiers $-1 \leq i < 5$. Les transformées en nombres de Fermat directes et inverses, de longueur $M = 2^{t+1-i}$, sont donc implantées selon les relations suivantes :

$$X_k = \left\langle \sum_{n=0}^{M-1} x_n 2^{\left\langle 2^i nk \right\rangle_M} \right\rangle_{F_4} \tag{2.48}$$

$$x_n = \left\langle -2^{2^t-1-(t-i)} \sum_{k=0}^{M-1} X_k 2^{\left\langle -2^i nk \right\rangle_M} \right\rangle_{F_4} \tag{2.49}$$

avec $k,\ n = 0, 1, \ldots, 2^{t+1-i} - 1$.

Une transformée en nombres de Fermat admet la propriété de la convolution cyclique qui permet de l'appliquer à des algorithmes rapides utilisés dans le système d'annulation d'écho acoustique. Dans cette optique, une proposition de traiter les algorithmes du filtrage adaptatif par blocs d'échantillons au lieu d'échantillon par échantillon fait intervenir une série de produits de convolution. Ces produits de convolution peuvent être entièrement réalisés par la transformation en nombres entiers [Lee1985] et en particulier par la transformation en nombres de Fermat.

Chapitre 3

Traitement par bloc des algorithmes d'annulation d'écho

3.1 Introduction

La mise en oeuvre des différents algorithmes appliqués au système d'annulation d'écho acoustique représente une charge de calcul globale importante. C'est pourquoi, des problèmes de temps et de complexité d'exécution se posent toujours lorsqu'on cherche à implanter un algorithme d'annulation d'écho dans un processeur temps réel. Pour ces raisons, il est alors nécessaire de réduire au maximum les coûts des opérations à traiter pour pouvoir envisager une implantation des algorithmes d'annulation d'écho acoustique dans un microprocesseur de traitement de signal. En effet, si le nombre d'opérations mises en jeu devient trop élevé pour le processeur *DSP* choisi, if faudra chercher à simplifier ce traitement. L'idée est de proposer de nouveaux algorithmes plus efficaces qui ne modifient pas le traitement à réaliser.

Dans cette partie, nous allons présenter une variante des algorithmes adaptatifs (*LMS*, *NLMS*, *PNLMS*, et *PNLMS++*) présentés dans le premier chapitre. Au lieu de procéder à l'adaptation des coefficients à chaque nouvelle itération, nous proposons d'utiliser l'adaptation de ces algorithmes seulement tous les N échantillons constituant un bloc, $N \geq 0$ désignant la taille du bloc d'adaptation. Ceci implique une réduction de la charge de calculs d'un facteur N, au moins pour la partie adaptation.

Ce traitement par blocs, qui fait intervenir une série de produits de convolution, peut être réalisé, avec une complexité réduite de calcul, par la mise en oeuvre de la transformée de Fourier rapide (FFT : *Fast Fourier Transform*).

Bien que les processeurs *DSP* récents aient une puissance de calcul de plus en plus importante, les multiplications restent bien plus complexes qu'une réalisation d'additions ou de décalages de bits, c'est

pourquoi, nous nous intéressons plus particulièrement au nombre de multiplications pour l'évaluation du coût de calcul des différentes procédures. Précisons que la plupart des estimations sur les nombres d'opérations nécessaires au traitement des fonctions des algorithmes adaptatifs ne sera donnée qu'à titre indicatif et comparatif.

Pour une complexité de calcul réduite au maximum, nous proposons d'implanter ces différents algorithmes traités par bloc, par la mise en oeuvre de transformées en nombres de Fermat (FNT).

Nous présenterons dans ce chapitre des mesures objectives pour comparer les nouveaux algorithmes implantés par les deux méthodes, la FNT et la FFT.

3.2 Algorithme du gradient stochastique par bloc (BLMS)

3.2.1 Introduction

Le filtrage numérique par bloc a été étudié en détail par Burrus, Mitra, et al. [Burrus1971, Gnanasekaran1977, Ferrara1980, Clark1981]. Au lieu de procéder à l'adaptation des coefficients à chaque nouvel échantillon, ce qui revient à adapter le filtre 16000 fois par seconde pour une fréquence d'échantillonnage $f_e = 16\ kHz$, l'algorithme $BLMS$ (*Bloc Least Mean Squares*) consiste à adapter les coefficients du filtre seulement tous les N échantillons. Ceci implique une réduction du temps d'exécution lorsqu'on cherche à implanter cet algorithme de filtrage dans un processeur temps réel.

Comme nous avons vu précédemment, le filtrage adaptatif est un filtre numérique de réponse impulsionnelle finie (RIF) qui permet d'estimer le signal d'entrée $\{x\}$ au moyen de ses coefficients $\hat{w}_k$ pour obtenir un écho estimé $\hat{y}_k$ suivant la relation :

$$\hat{y}_k = \sum_{n=0}^{L-1} \hat{w}_k\,(n)\, x_{k-n} \tag{3.1}$$

où L désigne l'ordre du filtre RIF.

La mise à jour des coefficients du filtre adaptatif est réalisée selon l'équation :

$$\hat{w}_{k+1} = \hat{w}_k + \mu \chi_k e_k \tag{3.2}$$

μ désigne le pas d'adaptation qui contrôle la convergence du filtre adaptatif, $\hat{w}_k$ et χ_k deux vecteurs représentant respectivement le filtre d'adaptation et le signal d'entrée.

$$\hat{w}_k = \left[\begin{array}{cccc} \hat{w}_k\,(0) & \hat{w}_k\,(1) & \ldots & \hat{w}_k\,(L-1) \end{array} \right]^T$$

$$\chi_k = \begin{bmatrix} x_k & x_{k-1} & \dots & x_{k-L+1} \end{bmatrix}^T \tag{3.3}$$

L'erreur e_k, appelée aussi écho résiduel e_k, à l'instant k est donnée par la différence entre l'entrée désirée d_k et l'écho estimé $\hat{y}_k$:

$$\begin{aligned} e_k &= d_k - \hat{y}_k \\ &= d_k - \sum_{n=0}^{L-1} \hat{w}_k(n)\, x_{k-n} \end{aligned} \tag{3.4}$$

où k est l'indice d'échantillons.

Dans cette partie, nous allons décrire la procédure de traitement du filtrage adaptatif par bloc. La formulation mathématique de cette procédure est obtenue sous forme vectorielle en traitant les N échantillons de l'intervalle $[kN;\, kN+N-1]$ à chaque itération k désignant l'indice du bloc $(k \in \mathbb{N})$.

Soit $\hat{w}_k$ le filtre adaptatif de longueur L conçu pour estimer le chemin de retour acoustique et mis à jour à chaque itération k.

Nous considérons les vecteurs de données $\tilde{d}_k$, $\tilde{y}_k$ et ϵ_k de longueur N tels que :

$$\tilde{d}_k = \begin{bmatrix} d_{kN} & d_{kN+1} & \dots & d_{kN+N-1} \end{bmatrix}^T \tag{3.5}$$

$$\tilde{y}_k = \begin{bmatrix} \hat{y}_{kN} & \hat{y}_{kN+1} & \dots & \hat{y}_{kN+N-1} \end{bmatrix}^T \tag{3.6}$$

$$\epsilon_k = \tilde{d}_k - \tilde{y}_k = \begin{bmatrix} e_{kN} & e_{kN+1} & \dots & e_{kN+N-1} \end{bmatrix}^T \tag{3.7}$$

La sortie du filtre adapté par un algorithme traité par bloc est donné par : [Burrus1971]

$$\begin{aligned} \tilde{y}_k &= \begin{bmatrix} x_{kN} & x_{kN-1} & \dots & x_{kN-L+1} \\ x_{kN+1} & x_{kN} & \dots & x_{kN-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{kN+N-1} & x_{kN+N-2} & \dots & x_{kN+N-L} \end{bmatrix} \begin{bmatrix} \hat{w}_k(0) \\ \hat{w}_k(1) \\ \vdots \\ \hat{w}_k(L-1) \end{bmatrix} \\ &= \Re_k \hat{w}_k \end{aligned} \tag{3.8}$$

où $\Re_k$ est une matrice de Toeplitz de taille $(L \times N)$.

Le calcul de l'écho estimé, $\tilde{y}_k$, a été réalisé par convolution discrète selon le produit matriciel donné par

l'équation (3.8). Cet écho estimé va nous permettre de définir la structure en bloc de l'algorithme LMS $(BLMS)$.

3.2.2 Algorithme LMS par bloc (BLMS)

Le filtre adaptatif est mis à jour par une technique de minimisation de l'erreur quadratique moyenne entre sa sortie $\{\tilde{y}\}$ et une sortie désirée, $\left\{\tilde{d}\right\}$. La fonction de coût associée est l'erreur quadratique instantanée évaluée sur un bloc de données : [Gnanasekaran1977, Ferrara1980, Clark1981, Gersho1983]

$$J\left(\hat{w}_k\right) = \left(\epsilon_k^T \epsilon_k\right) \tag{3.9}$$

Les coefficients du filtre adaptatif $\hat{w}_k$ sont alors mis à jour selon l'équation :

$$\hat{w}_{k+1} = \hat{w}_k - \frac{1}{2}\mu_B \bigtriangledown_B J\left(\hat{w}_k\right) \tag{3.10}$$

$\bigtriangledown_B J\left(\hat{w}_k\right)$ est le gradient de la fonction de coût J par rapport au vecteur $\hat{w}_k$. Il est défini par :

$$\bigtriangledown_B J\left(\hat{w}_k\right) = \bigtriangledown_B \left(\epsilon_k^T \epsilon_k\right) = \frac{\partial\left(J\left(\hat{w}_k\right)\right)}{\partial \hat{w}_k} \tag{3.11}$$

où μ_B est le pas d'adaptation.

Afin de calculer ce terme de gradient, nous développons l'expression de la fonction de coût $J\left(\hat{w}_k\right)$:

$$\begin{aligned} J\left(\hat{w}_k\right) &= \left(\epsilon_k^T \epsilon_k\right) \\ &= \sum_{j=kN}^{(k+1)N-1} \left(e_j\right)^2 \\ &= \sum_{j=kN}^{(k+1)N-1} \left(d_j - \hat{y}_j\right)^2 \end{aligned} \tag{3.12}$$

Par dérivation de l'équation (3.12) et en reconnaissant les expressions respectives de $\tilde{y}_k$ et du vecteur d'erreur ϵ_k des équations (3.8) et (3.7), nous en déduisons que le gradient $\bigtriangledown_B J\left(\hat{w}_k\right)$ admet comme expression :

$$\bigtriangledown_B J\left(\hat{w}_k\right) = \frac{\partial\left(J\left(\hat{w}_k\right)\right)}{\partial \hat{w}_k} = \frac{\partial\left(\sum_{j=kN}^{(k+1)N-1}\left(e_j\right)^2\right)}{\partial \hat{w}_k}$$

$$= \frac{\partial\left(\sum_{j=kN}^{(k+1)N-1}\left(d_j - \hat{y}_j\right)^2\right)}{\partial \hat{w}_k} = \frac{\partial\left(\sum_{j=kN}^{(k+1)N-1}\left(d_j - \chi_j \hat{w}_k\right)^2\right)}{\partial \hat{w}_k} \tag{3.13}$$

$$= -2\left(\sum_{j=kN}^{(k+1)N-1}\left(d_j - \chi_j\hat{w}_k\right)(\chi_j)\right) = -2\left(\sum_{j=kN}^{(k+1)N-1}\chi_j e_j\right)$$

alors

$$\bigtriangledown_B\left(\epsilon_k^T\epsilon_k\right) = -2\Re_k^T\epsilon_k \tag{3.14}$$

Les coefficients du filtre adaptatif $\hat{w}_k$ sont alors mis à jour selon l'équation et le principe décrit par la figure (3.1) :

$$\hat{w}_{k+1} = \hat{w}_k - \frac{1}{2}\mu_B\bigtriangledown_B\left(\epsilon_k^T\epsilon_k\right) = \hat{w}_k + \mu_B\Re_k^T\epsilon_k \tag{3.15}$$

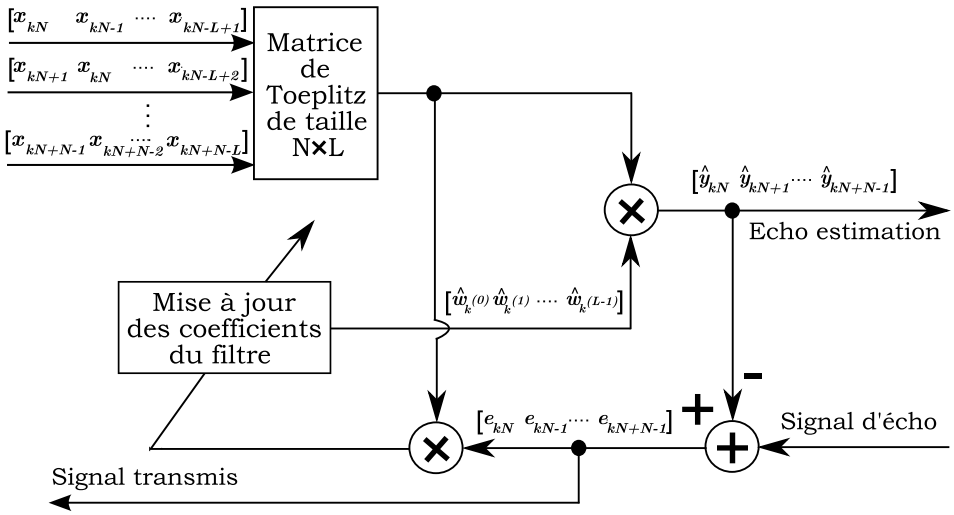

FIG. 3.1 – Système de filtrage adaptatif par bloc

L'étude de la convergence en moyenne vers la solution de Wiener montre que le domaine de stabilité de l'algorithme $BLMS$ est identique à celui du LMS [Clark1981]. Par conséquent, la convergence en moyenne de l'algorithme $BLMS$ est garantie pour un pas d'adaptation μ_B satisfaisant la condition suivante :

$$0 < \mu_B < \frac{2}{\lambda_{max}} \tag{3.16}$$

où λ_{max} est la plus grande valeur propre de la matrice d'autocorrélation Φ_{xx} donnée par (1.8) [Bernard1975, Bernard1976, Sen1997].

L'algorithme LMS temporel par blocs est très simple à mettre en place. Son coût de calcul est fortement conditionné par le calcul des produits matriciels des équations (3.8) et (3.14) qui sont de l'ordre de $N \times L$. L'équation (3.8) correspond à une convolution et peut donc être implémentée de manière rapide par la transformée de Fourier. Il en est de même pour l'équation (3.14) qui correspond à une corrélation croisée.

Pour une complexité de calcul réduite, nous proposerons, par la suite, de remplacer la transformée de Fourier par la mise en oeuvre de la transformée en nombres de Fermat (FNT).

3.2.3 BLMS par la transformée de Fourier rapide

L'algorithme $BLMS$ peut être rendu moins coûteux en temps de calcul si on choisit de réaliser le calcul matriciel, qui est très lourd du point de vue calculatoire (équation 3.8 et 3.14), par le calcul de la convolution. En opérant de la sorte, nous tirons alors avantage des propriétés de convolution circulaire de la transformée de Fourier discrète (TFD) et de la rapidité de calcul par la transformée de Fourier rapide (FFT). Cet algorithme appelé Fast $BLMS$, a été introduit par Clark [Clark1981].

Les valeurs du signal d'écho $\tilde{y}_k$ sont estimées, comme nous l'avons précédemment vu plus haut, au moyen d'une convolution linéaire représentée matriciellement par l'équation (3.8), $\tilde{y}_k = \Re_k \hat{w}_k$. Le calcul de ces valeurs nécessite des opérations très lourdes.

Le principe général des algorithmes de calcul de convolution rapide consiste à remplacer le calcul matriciel par la convolution circulaire. En effet, une convolution circulaire dans le domaine temporel est équivalente à une multiplication terme à terme dans le domaine de la TFD, ce qui permet de réduire la complexité de calcul dans les algorithmes faisant appel à la TFD.

Pour cette raison, nous allons calculer les coefficients du filtre adaptatif de l'algorithme $BLMS$ de manière à pouvoir utiliser la propriété de la convolution circulaire : [Clark1981, Clark1983, Lee1983, Lee1985]

$$\begin{aligned} \hat{w}_{k+1} &= \hat{w}_k - \tfrac{1}{2}\mu_B \bigtriangledown_B \left(\epsilon_k^T \epsilon_k\right) \\ &= \hat{w}_k + \mu_B \Re_k^T \epsilon_k \\ &= \hat{w}_k + \mu_B \textstyle\sum_{j=kN}^{kN+N-1} e_j \chi_j \\ &= \hat{w}_k + \mu_B \phi_k \end{aligned} \tag{3.17}$$

$\phi_k = \Re_k^T \epsilon_k$ correspond à une corrélation croisée dont les coefficients ϕ_{ik} sont donnés par :

$$\phi_{ik} = \sum_{m=0}^{N-1} e_{kN+m} x_{kN+m-i},\ 0 \leq i \leq L-1$$

Dans le cas $n = kN + m$, les ϕ_{ik} peuvent s'écrire sous forme d'une convolution de la manière suivante :

$$\phi_{ik} = \sum_{n=kN}^{(k+1)N-1} e_n x_{n-i} = e_i * x_{-i} \tag{3.18}$$

où * représente la convolution circulaire.

L'équation (3.18) montre que la mise à jour des coefficients du filtre est donnée par une convolution circulaire.

Les différents éléments du vecteur $\hat{w}_k$ de longueur L sont alors donnés par :

$$\hat{w}_{k+1}(i) = \hat{w}_k(i) + \mu_B (e_i * x_{-i}) \tag{3.19}$$

Les éléments du vecteur d'erreur $\epsilon_k = \tilde{d}_k - \tilde{y}_k$ de longueur N sont calculés par :

$$\begin{aligned} e_{kN+m} &= d_{kN+m} - y_{kN+m} \\ &= d_{kN+m} - \sum_{i=0}^{L-1} \hat{w}_k(i)\, x_{kN+m-i} \\ &= d_{kN+m} - \hat{w}_k(m) * x_k(m) \end{aligned} \tag{3.20}$$

Cette convolution rapide temporelle présente dans les équations (3.19) et (3.20) peut être calculée dans le domaine de la FFT avec une complexité de calcul réduite. Le filtrage adaptatif par l'algorithme $BLMS$ consiste alors à calculer la convolution circulaire entre les vecteurs $\hat{w}_k$ et $\tilde{x}_k$, de longueur respective L et $N+L-1$ et la convolution circulaire entre les vecteurs $\tilde{x}_{-k}$ et ϵ_k, $\tilde{x}_k$ étant la séquence du signal d'entrée définie par :

$$\tilde{x}_k = \begin{bmatrix} x_{kN-L+1} & x_{kN-L+2} & \dots & x_{kN+N-1} \end{bmatrix}^T \tag{3.21}$$

Notons $\tilde{Y}_k$ le produit de la convolution circulaire entre $\tilde{x}_k$ et $\hat{w}_k$:

$$\begin{aligned} \tilde{Y}_k &= \left(\begin{bmatrix} \hat{w}_k^T \mid & 0_{1\times(M-L)} \end{bmatrix} \right) * \left(\begin{bmatrix} \tilde{x}_k^T \mid & 0_{(1\times M-(N+L-1))} \end{bmatrix} \right)^T \\ &= FFT^{-1}\left(FFT\left(\left[\hat{w}_k^T \mid 0_{1\times(M-L)} \right] \right) \bullet FFT\left(\left[\tilde{x}_k^T \mid 0_{(1\times M-(N+L-1))} \right] \right) \right)^T \end{aligned} \tag{3.22}$$

où FFT et FFT^{-1} désignent respectivement les transformées directe et inverse de Fourier. $0_{(i\times j)}$ est une matrice nulle de taille $(i \times j)$.

Pour le calcul de la FFT, il faut choisir M égale à une puissance de 2. En général, on choisit $M = N+L-1$ si cette valeur est une puissance de 2 sinon il faut compléter les vecteurs par un nombre nécessaire de coefficients nuls afin de leur donner une taille égale à une puissance de 2. Pour simplifier, on suppose que $M \geq N+L-1$.

A chaque itération k, la mise à jour du filtre adaptatif est alors réalisée par le calcul de la convolution circulaire dans le domaine de la FFT selon l'expression suivante :

$$\begin{aligned} \hat{w}_{k+1} &= \hat{w}_k + \mu_B P_1 \left(\left(\begin{bmatrix} \epsilon_k^T \mid & 0_{1\times(M-N)} \end{bmatrix} \right) * \left(\begin{bmatrix} \tilde{x}_{-k}^T \mid & 0_{(1\times M-(N+L-1))} \end{bmatrix} \right) \right)^T \\ &= \hat{w}_k + \mu_B P_1 FFT^{-1} \left(FFT\left(\begin{bmatrix} \epsilon_k^T \mid & 0_{1\times(M-N)} \end{bmatrix} \right) \bullet FFT\left(\begin{bmatrix} \tilde{x}_{-k}^T \mid & 0_{(1\times M-(N+L-1))} \end{bmatrix} \right) \right)^T \end{aligned} \tag{3.23}$$

où

$$\tilde{x}_{-k} = \left[\begin{array}{cccc} x_{kN+N-1} & x_{kN+N-2} & \cdots & x_{kN-L+1} \end{array} \right]^T \tag{3.24}$$

P_1 est une matrice de dimension $(L \times M)$ qui permet d'écarter les $(M - L)$ premières composantes vectorielles introduites lors de la transformation inverse. Elle est définie par :

$$P_1 = \left[\begin{array}{c|c} 0_{L\times(M-L)} & I_L \end{array} \right] \tag{3.25}$$

I_L est une matrice identité de taille $(L \times L)$.

Grâce à l'utilisation de la FFT, le calcul de l'erreur et de la mise à jour des coefficients du filtre par blocs est de cette façon rendu moins coûteux que dans le cas du $BLMS$ temporel.

Pour une complexité de calcul réduite au maximum, nous proposons de remplacer la FFT par la mise en oeuvre de la transformée en nombres de Fermat (FNT) [Shu1988].

3.3 Algorithme du gradient stochastique Proportionné Normalisé par bloc (BPNLMS++)

Dans un système d'annulation d'écho acoustique, un algorithme adaptatif doit répondre à deux critères que sont la vitesse de convergence et la complexité de calcul. Notre étude consiste à proposer un algorithme performant qui répondrait simultanément à ces deux critères.

Nous avons vu, dans le premier chapitre, que l'algorithme $PNLMS{+}{+}$ présente une meilleure convergence par rapport aux autres algorithmes du filtrage $NLMS$ et $PNLMS$. Il est alors intéressant de traiter cet algorithme par bloc afin de réduire sa complexité pour pouvoir envisager un algorithme d'annulation d'écho acoustique qui répond aux deux critères précédents.

Nous allons rappeler rapidement les différentes équations du filtrage adaptatif par l'algorithme $PNLMS{+}{+}$. L'algorithme de mise à jour des coefficients $\hat{w}_k$ du filtrage adapté par $PNLMS++$ est le suivant :

$$\begin{cases} \hat{w}_{k+1} = \hat{w}_k + \mu_k \chi_k e_k = \hat{w}_k + \mu \frac{\chi_k e_k}{\chi_k^T \chi_k + \beta}, \text{ pour le bloc pair } (k \,\text{pair}) \\ \hat{w}_{k+1} = \hat{w}_k + \mu_k \chi_k e_k = \hat{w}_k + \mu \frac{G_k \chi_k e_k}{\chi_k^T G_k \chi_k + \beta}, \text{ pour le bloc impair } (k \,\text{impair}) \end{cases}$$

où, $G_k = diag \left[\begin{array}{ccc} g_k(0), & \ldots, & g_k(L-1) \end{array} \right]$ est une matrice diagonale $(L \times L)$ avec $g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\sum_{m=0}^{L-1}\gamma_k(m)}$, où $\gamma_k(n) = max\{\rho\nu_k, |\hat{w}_k(n)|\}$, $n \in \{0, \ldots, L-1\}$ et où $\nu_k = max\{\delta, |\hat{w}_k(0)|, \ldots, |\hat{w}_k(L-1)|\}$. Les termes ρ et δ sont respectivement choisis égaux à $\frac{5}{L}$ et à 10^{-2}.

Il correspond à l'algorithme $NLMS$ pour un indice d'itération pair et à l'algorithme $PNLMS$ pour un

indice d'itération impair. Avant de traiter cet algorithme par blocs, nous traitons tout d'abord, par blocs, les deux algorithmes qui le constituent, $NLMS$ et $PNLMS$.

3.3.1 Traitement par blocs de l'algorithme LMS normalisé (BNLMS)

Une variante de l'algorithme Bloc Least Mean Squares ($BLMS$) a été proposée. Cette variante consiste à normaliser le pas d'adaptation par rapport à la fonction d'autocorrélation $\mathbf{R}_{xx}(k) = \tilde{x}_k * \tilde{x}_{-k}$, pour un bloc d'indice k. L'algorithme $BNLMS$ présente par rapport à l'algorithme $BLMS$ l'intérêt d'être indépendant de la variance du signal d'entrée, ce qui donne une convergence plus uniforme durant chaque processus d'adaptation.

La normalisation du pas d'adaptation par rapport à la fonction d'autocorrélation est choisie pour mettre en application cette fonction dans le domaine de convolution rapide. La fonction d'autocorrélation $\mathbf{R}_{xx}(k)$ est la convolution rapide dans le domaine de Fourier du spectre d'énergie du signal d'entrée $\{x\}$. Elle est définie par :

$$\mathbf{R}_{xx}(k) = P_1 FFT^{-1}\left(FFT\left(\left[\begin{array}{c|c} \tilde{x}_k^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right) \bullet FFT\left(\left[\begin{array}{c|c} \tilde{x}_{-k}^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right)\right)^T \tag{3.26}$$

où $\tilde{x}_k$ et $\tilde{x}_{-k}$ sont respectivement donnés par les relations (3.21) et (3.24).

A partir de l'équation (3.23), une normalisation du pas d'adaptation μ_B par la fonction d'autocorrélation $R_{xx}(k)$, permet de définir la nouvelle variante de l'algorithme $BNLMS$.

La mise à jour du filtre adaptatif par cet algorithme est donnée par : [Kazuo1990, Baghious2004]

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{R_{xx}(k) + \beta}(\epsilon_k * \tilde{x}_{-k}) \tag{3.27}$$

$$= \hat{w}_k + \mu_B \frac{P_1 FFT^{-1}\left(FFT\left(\left[\begin{array}{c|c} \epsilon_k^T & 0_{1\times(M-N)} \end{array}\right]\right) \bullet FFT\left(\left[\begin{array}{c|c} \tilde{x}_{-k}^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right)\right)^T}{P_1 FFT^{-1}\left(FFT\left(\left[\begin{array}{c|c} \tilde{x}_k^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right) \bullet FFT\left(\left[\begin{array}{c|c} \tilde{x}_{-k}^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right)\right)^T + \beta}$$

où, $\epsilon_k = \left[\begin{array}{cccc} e_{kN} & e_{kN+1} & \ldots & e_{kN+N-1} \end{array}\right]^T$ est le vecteur erreur défini par :

$$\epsilon_k = \tilde{d}_k - P_2 FFT^{-1}\left(FFT\left(\left[\begin{array}{c|c} \hat{w}_k^T & 0_{1\times(M-L)} \end{array}\right]\right) \bullet FFT\left(\left[\begin{array}{c|c} \tilde{x}_k^T & 0_{(1\times M-(N+L-1))} \end{array}\right]\right)\right)^T \tag{3.28}$$

$$= \tilde{d}_k - P_2 \tilde{Y}_k$$

Le vecteur $\tilde{Y}_k$ est de longueur M, ses N dernières composantes correspondent à la convolution linéaire

de l'équation (3.8) c'est-à-dire aux composantes de $\tilde{y}_k$. Les $(M-N)$ premières composantes résultant de la convolution circulaire sont écartées par l'application de la matrice P_2 de dimension $(N \times M)$ définie par :

$$P_2 = \left[0_{N\times(M-N)} \mid I_{N\times N}\right] \tag{3.29}$$

3.3.2 Traitement par blocs de l'algorithme NLMS proportionné (BPNLMS)

Comme nous l'avons rappelé dans le premier chapitre, l'algorithme $PNLMS$ est développé dans le contexte de l'annulation d'écho acoustique pour adapter les coefficients du filtre adaptatif. Cette adaptation efficace réalise une convergence des coefficients du filtre adaptatif sensiblement plus rapide que celle obtenue par l'algorithme standard *NLMS*. Comme nous l'avons fait pour l'algorithme $NLMS$, nous proposons d'étudier l'algorithme $PNLMS$ en traitant les échantillons par blocs. L'algorithme ainsi obtenu, *BPNLMS*, permettra de diminuer la charge de calcul par la mise en oeuvre de la transformée en nombres entiers et en particulier par la transformée en nombres de Fermat (FNT) et en utilisant la même technique que celle utilisé pour l'algorithme $BNLMS$.

L'algorithme $BPNLMS$ propose une nouvelle normalisation du pas d'adaptation, μ_B, selon l'équation suivante :

$$\mu_B(k) = \frac{\mu_B G_k}{G_k R_{xx} + \beta} \tag{3.30}$$

G_k est une matrice diagonale définie par :

$$G_k = diag\left[\begin{array}{cccc} g_k(0) & g_k(1) & \ldots & g_k(L-1) \end{array}\right] \tag{3.31}$$

dont les éléments $g_k(n)$ sont donnés par :

$$g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\sum_{m=0}^{L-1}\Sigma_{m=0}^{L-1}\gamma_k(m)} \tag{3.32}$$

où $\gamma_k(n) = max\left\{\begin{array}{cc} \rho\nu_k, & |\hat{w}_k(n)| \end{array}\right\}$, $n \in \{0,\ldots,L-1\}$ et $\nu_k = max\left\{\delta, |\hat{w}_k(0)|, \ldots, |\hat{w}_k(L-1)|\right\}$. Les termes ρ et δ ont respectivement les valeurs $\frac{5}{L}$ et 10^{-2}.

La mise à jour des coefficients $\hat{w}_k$ du filtre adaptatif par l'algorithme $BPNLMS$ est donnée par : [Benesty2001, Baghious2004]

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B G_k}{G_k R_{xx} + \beta}(\epsilon_k * \tilde{x}_{-k}) \tag{3.33}$$

Dans cette équation, le calcul de la fonction d'autocorrélation $\mathbf{R}_{xx}$ et de la convolution est réalisé de la

même manière que dans le paragraphe précédent.

Si la réponse impulsionnelle du filtre adaptatif est dispersée [Gersho1983], dans ce cas, la convergence donnée par l'algorithme $BPNLMS$ sera plus lente que celle donnée par l'algorithme $BNLMS$. Comme nous l'avons rappelé dans le premier chapitre, l'algorithme $PNLMS++$, développé dans le contexte de l'annulation d'écho acoustique pour adapter les coefficients du filtre adaptatif, présente une convergence plus rapide que celle donnée par l'algorithme $PNLMS$. Il est alors intéressant de le traiter par blocs. Ce traitement qui fait appel au calcul d'une série de produits de convolution, permet, dans un premier temps, une implantation simple par la mise en oeuvre de la FFT, puis, dans un deuxième temps, une implantation de complexité réduite par la mise en oeuvre de la FNT.

3.3.3 Traitement par blocs de l'algorithme PNLMS++ (BPNLMS++)

Nous proposons de traiter l'algorithme $PNLMS++$ par bloc d'échantillons et non échantillon par échantillon. Cet algorithme intitulé $BPNLMS{+}{+}$, résulte alors d'une combinaison des algorithmes $BNLMS$ et $BPNLMS$.

La mise à jour des coefficients du filtre adaptatif par l'algorithme proposé $BPNLMS{+}{+}$ est alors donnée par [Benesty2001, Baghious2004] :

• si k est pair, nous utilisons l'adaptation du filtre, donnée par l'algorithme $BNLMS$.

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{R_{xx}(k) + \beta} (\epsilon_k * \tilde{x}_{-k}) \tag{3.34}$$

• si k est impair, nous utilisons l'adaptation du filtre, donnée par l'algorithme $BPNLMS$.

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{G_k R_{xx} + \beta} G_k (\epsilon_k * \tilde{x}_{-k})$$

Cette alternance peut améliorer la convergence du filtre adaptatif en utilisant des techniques basées sur la méthode rapide de la FFT et de la NTT.

Les différents éléments du vecteur d'erreur $\epsilon_k = \tilde{d}_k - \tilde{y}_k$ sont définis par :

$$e_{kN+m} = d_{kN+m} - \hat{y}_{kN+m} = d_{kN+m} - \sum_{i=0}^{L-1} \hat{w}_k(i)\, x_{kN+m-i} \tag{3.35}$$

avec $m \in \{0, 1, \ldots, N-1\}$. D'ailleurs, le dénominateur de l'équation de la mise à jour a été redéfini comme calcul de corrélation. Ainsi, les différents éléments du $\hat{w}_{k+1}$ sont donnés par :

$$\begin{cases} \hat{w}_{k+1}(n) = \hat{w}_k(n) + \mu_B \frac{\sum_{m=0}^{N-1} e_{kN+m} x_{kN+m-i}}{\beta + \sum_{m=0}^{N-1} x_{kN+m} x_{kN+m-i}}, \text{ pour le bloc pair } (k \text{ pair}) \\ \hat{w}_{k+1}(n) = \hat{w}_k(n) + \mu_B \frac{\sum_{m=0}^{N-1} e_{kN+m} g_k(n) x_{kN+m-i}}{\beta + \sum_{m=0}^{N-1} x_{kN+m} g_k(n) x_{kN+m-i}}, \text{ pour le bloc impair } (k \text{ impair}) \end{cases} \tag{3.36}$$

$n \in \{0, 1, \ldots, L-1\}$.

Les éléments $g_k(n)$ de la matrice diagonale G_k sont calculés à partir de l'équation (3.32).

En conclusion, le traitement par blocs des différents algorithmes du filtrage adaptatif fait appel aux calculs de produits de convolution qui peuvent être réalisés soit par la FFT soit par la FNT.

Une comparaison des performances de ces algorithmes, surtout en terme de degré de complexité de leur implantation, nous permettra de déterminer l'algorithme le plus performant qui sera utilisé par la suite dans le système d'annulation d'écho. Les résultats de cette comparaison sont donnés dans le paragraphe qui suit.

3.4 Comparaison des performances des différents algorithmes (BN-LMS, BPNLMS, BPNLMS++)

Plusieurs simulations numériques ont été réalisées pour évaluer et comparer les performances des différents algorithmes du filtrage adaptatif proposés (*BNLMS*, *BPNLMS*, *BPNLMS++*). Pour ces tests, l'algorithme $BPNLMS++$ est comparé aux autres algorithmes, $BNLMS$ et $BPNLMS$ à l'aide du logiciel de programmation *MatLab*, mettant en oeuvre la transformée de Fourier rapide (FFT).

Le problème qui se pose est celui du choix d'un algorithme d'optimisation. Ce choix va être déterminé par le nombre d'opérations nécessaires à chaque étape pour mettre à jour les coefficients du filtre adaptatif et par la vitesse de convergence de ce filtre. L'algorithme optimal est celui qui répondrait simultanément à ces deux critères.

Les différents algorithmes sont évalués avec un signal d'entrée et une réponse impulsionnelle de chemin d'écho présentés respectivement par les figures (3.2-a) et (3.2-b). Dans cette partie, nous nous intéressons au cas d'une simple parole (présence d'un signal lointain). L'atténuation de l'écho peut être mesurée à différents endroits, donnant ainsi des renseignements différents mais complémentaires à savoir, l'atténuation, notée ER, fournie par l'annulateur d'écho, la convergence N_m du filtre adapté par ces différents algorithmes, l'évolution des coefficients du filtre. La mesure, donnée en dB, est effectuée de manière classique en utilisant des blocs successifs de N échantillons, soit, pour un bloc d'indice k :

$$ER(k) = 10log_{10}\left(\sum_{n=(k-1)N+1}^{kN}(d_n - \hat{y}_n)^2\right) \tag{3.37}$$

$$N_m = 10log_{10}\left(\frac{\|w - \hat{w}\|^2}{\|w\|^2}\right) \tag{3.38}$$

où w et $\hat{w}$ désignent respectivement la réponse impulsionnelle et la réponse impulsionnelle estimée du chemin d'écho. d_n et $\hat{y}_n$ sont respectivement les échantillons du signal désiré et du signal d'écho estimé par

les différents processus d'adaptation.

Pour la simulation, les valeurs des différents paramètres utilisés dans les algorithmes sont données par : $\beta = 10^2$, $\delta = \frac{1}{L}$, $\rho = 10^{-2}$ et $\mu_B = 0.8$. Les dimensions du bloc et du filtre adaptatif sont respectivement données par : $N = 64$ et $L = 64$.

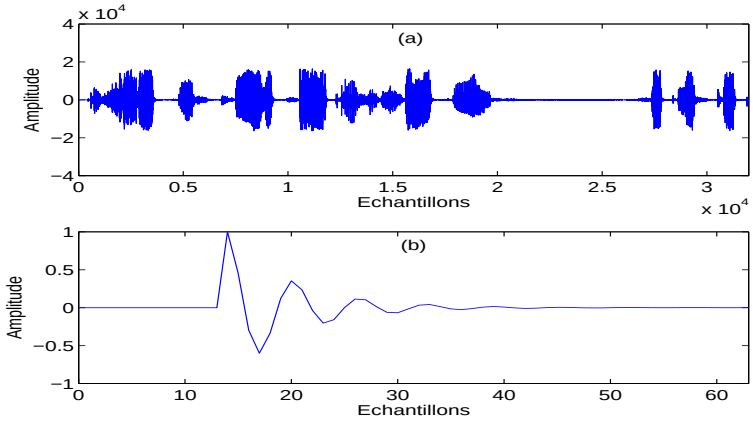

FIG. 3.2 – (a) : Signal d'entrée (b) : Réponse impulsionnelle du chemin d'écho

Les diverses simulations menées avec des signaux de parole ont permis de mettre en évidence certains comportements typiques des trois techniques de filtrage proposées. Les simulations ont été obtenues avec un rapport Signal à Bruit de l'ordre de 30 dB.

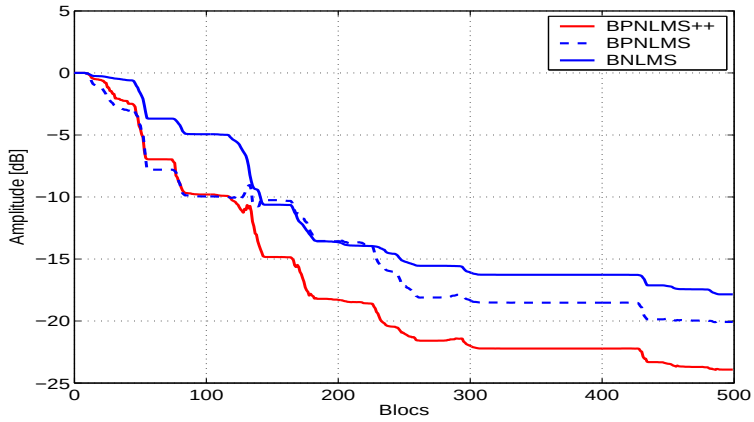

FIG. 3.3 – Convergence des coefficients du filtre par les différentes méthodes d'adaptation

Nous pouvons noter, à partir de la figure (3.3), que l'algorithme $BPNLMS++$ proposé, représenté par la courbe en rouge, présente une meilleure convergence par rapport aux autres algorithmes, $BNLMS$ et $BPNLMS$, représentés respectivement par les courbes en bleu gras et bleu pointillés.

La performance des trois algorithmes est mesurée cette fois par l'écho résiduel, $\epsilon_k = \tilde{d}_k - \tilde{y}_k$, présenté par la figure (3.4). La simulation de cette performance montre que l'algorithme $BPNLMS++$ (figure 3.4-a) fournit une atténuation d'écho plus grande que les deux autres algorithmes, $BNLMS$ (figure 3.4-c) et $BPNLMS$ (3.4-b).

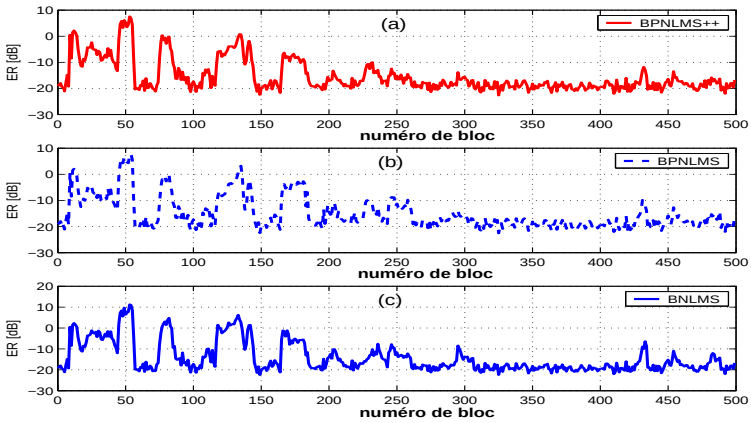

FIG. 3.4 – Echo résiduel mesuré pour les trois méthodes d'adaptation

L'autre critère de performance réside dans l'évolution des coefficients du filtre adaptatif. Cette évolution donnée par la figure (3.5) pour les trois algorithmes, $BNLMS$, $BPNLMS$, et $BPNLMS++$, montre que le filtre adapté par l'algorithme $BPNLMS++$ présente une meilleure évolution de ces coefficients par rapport au filtre adapté par les deux autres algorithmes.

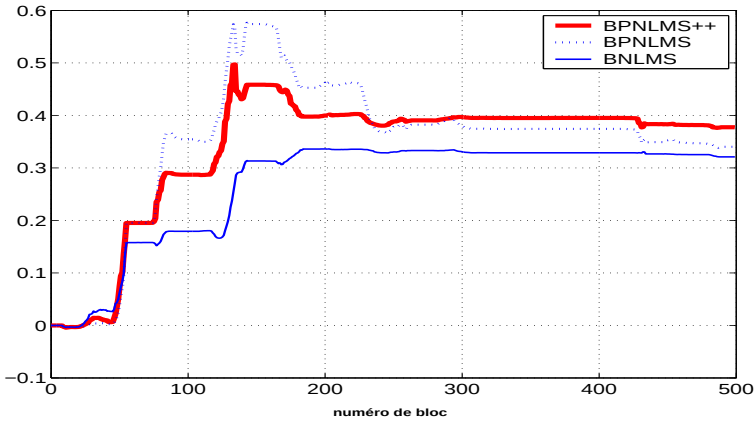

FIG. 3.5 – Evolution d'un coefficient, $(\hat{w}_k(21))$, du filtre par les trois méthodes d'adaptation

A partir de ces différents critères de performances, mesurés par l'atténuation ER fournie par l'annulateur d'écho, la convergence N_m du filtre adaptatif et l'évolution des coefficients du filtre adaptatif, nous constatons que l'algorithme $BPNLMS++$ fournit un meilleur résultat au sens de rendre l'écho résiduel moins audible en sortie du système d'annulation d'écho par rapport aux autres algorithmes, $BNLMS$, et $BPNLMS$.

Il est alors intéressant de proposer une implantation de l'algorithme $BPNLMS++$ sur un processeur de traitement du signal, (DSP), à virgule fixe, mettant en oeuvre la transformée en nombres de Fermat (FNT). Précisions que le choix d'un DSP à virgule fixe provient du fait que son coût de calcul est beaucoup plus réduit par rapport à un DSP à virgule flottante, mettant en oeuvre la FFT.

3.5 Implantation par la FNT de l'algorithme BPNLMS++ proposé

L'algorithme $BPNLMS++$ que nous avons proposé précédemment a été développé pour permettre son implantation par la mise en oeuvre de transformée en nombres de Fermat sur un processeur à $b = 16$ bits. Une implémentation de cet algorithme, basée sur la propriété de convolution cyclique, entraînera alors l'utilisation de transformée en nombres de Fermat de longueur $M = 128$ échantillons où le modulo F_t est pris égal au cinquième nombre de Fermat $F_4 = 2^b + 1 = 2^{16} + 1$. Toutes les opérations implantées dans le $BPNLMS++$ sont définies dans le corps de Galois modulo F_4, $GF(F_4)$, avec $\alpha = \sqrt[4]{2} = 4938$.

3.5.1 Procédures d'implantation de la convolution circulaire par la FNT

Le filtrage adaptatif par l'algorithme proposé $BPNLMS++$ requiert trois convolutions à calculer. Ces différentes convolutions pourront facilement être réalisées à l'aide de transformées en nombres entiers et plus particulièrement de transformée en nombres de Fermat (FNT), par l'application de la propriété de convolution circulaire.

Les différentes séquences à convoluer rencontrées dans le système d'annulation d'écho acoustique seront toutes de dimension égale à $M = 128$. En effet, le calcul de convolution de deux séquences de M échantillons, en utilisant la convolution circulaire, exige l'application de trois transformées en nombres entiers de longueur M, les séquences étant étendues par des zéros pour atteindre la longueur M, M étant une puissance de 2. Toutes les opérations implantées dans l'algorithme proposé sont définies dans le corps de Galois $GF(F_4)$.

Les différentes étapes de la procédure d'implantation de l'algorithme $BPNLMS++$ par la FNT, au moyen de la convolution circulaire, sont décrites ci-dessous et résumées par la figure (3.6) :

1. Fenêtrage du signal d'entrée $\{x\}$ (fenêtre de $N = 64$ échantillons).
2. Construction de $\tilde{X}_k$ par récupération des $(L-1) = 63$ échantillons du vecteur précédent, $\tilde{X}_{k-1}$.
3. Calcul du vecteur $\tilde{\mathbf{X}}_k$ par application de la FNT au vecteur d'entrée $\tilde{X}_k$.
4. Construction du vecteur $\tilde{W}_k$ par ajout de $(M-L)$ échantillons nuls à la fin du vecteur $\hat{w}_k$.
5. Calcul du vecteur $\tilde{\mathbf{W}}_k$ par application de la FNT au vecteur $\tilde{W}_k$.
6. Calcul du produit terme à terme $\tilde{\mathbf{W}}_k \bullet \tilde{\mathbf{X}}_k$ des vecteurs $\tilde{\mathbf{W}}_k$ et $\tilde{\mathbf{X}}_k$.
7. Calcul du vecteur $\tilde{Y}_k$ par application de la transformée inverse de la FNT $\left(FNT^{-1}\right)$ au vecteur $\tilde{\mathbf{W}}_k \bullet \tilde{\mathbf{X}}_k$: $\tilde{Y}_k = FNT^{-1}\left(\tilde{\mathbf{W}}_k \bullet \tilde{\mathbf{X}}_k\right)$.
8. Récupération des N derniers échantillons $(N = 64)$ du vecteur $\tilde{Y}_k$.
9. Calcul du vecteur d'erreur $\epsilon_k = \tilde{d}_k - \tilde{y}_k$.
10. Construction du vecteur $\tilde{E}_k$ par ajout de $(M-N)$ échantillons nuls à la fin du vecteur ϵ_k.
11. Calcul du vecteur $\tilde{\mathbf{E}}_k$ par application de la FNT au vecteur d'erreur $\tilde{E}_k$.
12. Calcul de $\tilde{X}_{-k}$ par application de la matrice $\overline{I}_{M\times M}$.

13. Calcul du vecteur $\tilde{\mathbf{X}}_{-k}$, transformée FNT du vecteur $\tilde{X}_{-k}$.
14. Calcul du produit terme à terme $\tilde{\mathbf{X}}_{-k} \bullet \tilde{\mathbf{E}}_k$.
15. Calcul de la transformée inverse de la FNT $\left(FNT^{-1}\right)$ du vecteur $\tilde{\mathbf{X}}_{-k} \bullet \tilde{\mathbf{E}}_k$.
16. Récupération des $L = 64$ derniers échantillons du vecteur obtenu à l'issue de l'étape précédente.
17. Calcul du produit terme à terme $\left(\tilde{\mathbf{X}}_k \bullet \tilde{\mathbf{X}}_{-k}\right)$.
18. Calcul de la transformée inverse de la FNT $\left(FNT^{-1}\right)$ du vecteur $\tilde{\mathbf{X}}_k \bullet \tilde{\mathbf{X}}_{-k}$.
19. Récupération des $L = 64$ derniers échantillons du vecteur obtenu à l'issue de l'étape précédente.
20. Mise à jour du calcul des coefficients $\hat{w}_k$ du filtre de longueur $L = 64$ par l'algorithme $BPNLMS++$.

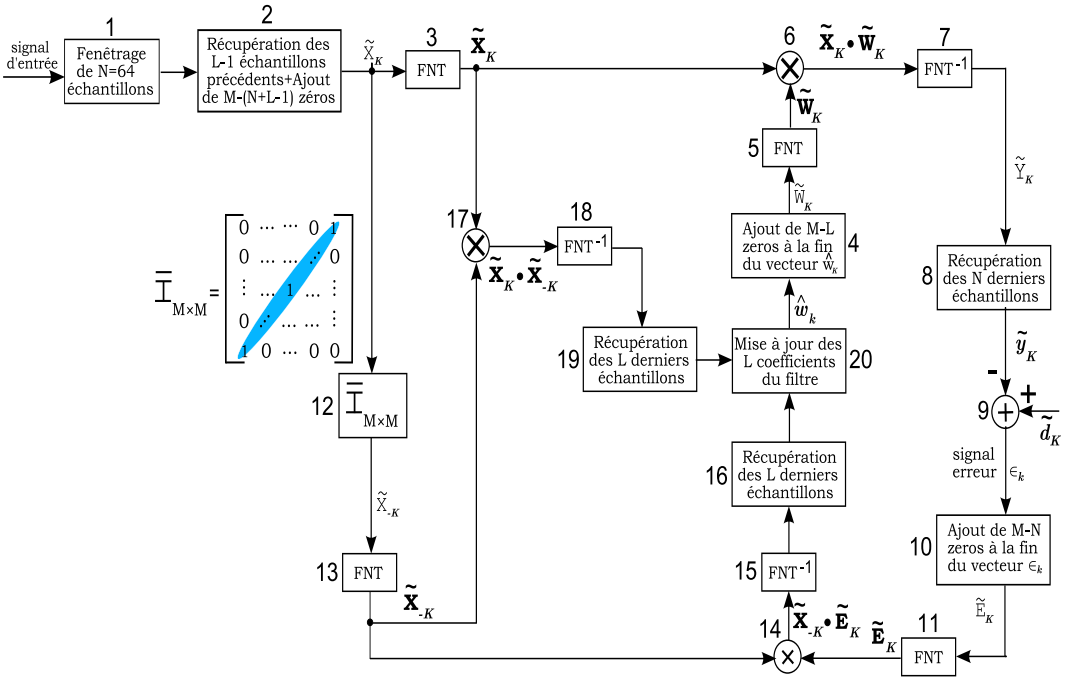

FIG. 3.6 – Procédure d'implantation de l'algorithme BPNLMS++ par la FNT

3.5.2 Résultats des simulations de l'implantation par la FNT de l'algorithme proposé

Afin de comparer les deux implantations, en FFT et en FNT, de l'algorithme $BPNLMS++$ nous avons effectué, sous le logiciel de programmation *MatLab,* un certain nombre de simulations.

Le signal d'entrée et la réponse impulsionnelle du chemin d'écho utilisés lors des simulations sont précédemment représentés par les figures (3.2-a) et (3.2-b).

Une transformée en nombres de Fermat présente bien des avantages par rapport à la transformée de Fourier. Citons ici deux points importants pour une implantation en virgule fixe, sur un processeur DSP, d'une FNT.

- Premièrement, l'implantation d'une FFT nécessite généralement l'enregistrement dans une table de

toutes les puissances des parties réelle et imaginaire du terme générateur $e^{\frac{-j2\pi}{M}}$. Cette mémorisation n'est pas négligeable au niveau des performances des processeurs. Ce stockage n'est cependant pas nécessaire pour une FNT où le terme générateur est égal à une puissance de deux. Les multiplications étant remplacées par des décalages de bits réalisés par une fonction logique, un préenregistrement dans une table n'est nécessaire que dans le cas d'une très grande valeur de M.

• Deuxièmement, une implémentation en virgule fixe sur un processeur d'une transformée de Fourier introduit irrémédiablement une erreur d'arrondi sur la séquence de sortie, ce qui n'est pas le cas d'une transformée en nombres de Fermat. La transformée de Fourier d'une séquence réelle codée sur b bits est calculée avec les deux parties réelle et imaginaire d'un mot complexe arrondi sur $\frac{b}{2}$ bits, alors que la précision sur b bits est maintenue par la FNT. Dans le cas d'une entrée réelle (signaux de parole), une FNT de b bits sera alors à comparer à une FFT de $\frac{b}{2}$ bits lors d'une implantation machine [Agarwal1975]. Ainsi, une opération de filtrage, réalisé à l'aide de la transformée en nombres de Fermat, ne présente aucune erreur d'arrondi.

Les différentes implantations de l'algorithme proposé, $BPNLMS++$, réalisées par la mise en oeuvre de la FNT utilisant un mot codé sur b bits et de la FFT utilisant un mot codé sur $\frac{b}{2}$ bits (fixed-point), sont comparées ($b = 16$ bits).

Toutes les multiplications de la transformée en nombres de Fermat peuvent être réalisées à l'aide d'additions et de décalages de bits là où des multiplications complexes, bien plus lourdes à mettre en oeuvre et moins rapides à exécuter, sont nécessaires pour une implantation en transformée de Fourier rapide.

• Pour calculer une FNT, les multiplications par des puissances de deux au nombre d'environ $\frac{M}{2}log_2M$, sont réalisées par des décalages de bits et des soustractions.

• Pour déterminer une FFT conventionnelle, une partie importante du temps de calcul est affectée aux $\frac{M}{2}log_2M$ multiplications complexes rencontrées dans cette transformée directe ou inverse.

Dans les différentes simulations, les valeurs des paramètres utilisés dans l'algorithme $BPNLMS++$, sont identiques pour les deux implantations. Ces différentes simulations sont évaluées, par des critères objectifs comme, l'évolution des coefficients du filtre adaptatif et l'atténuation d'écho fournie par l'annulateur d'écho.

Le premier critère déterminé par l'évolution des coefficients du filtre adaptatif est donné par la figure (3.7).

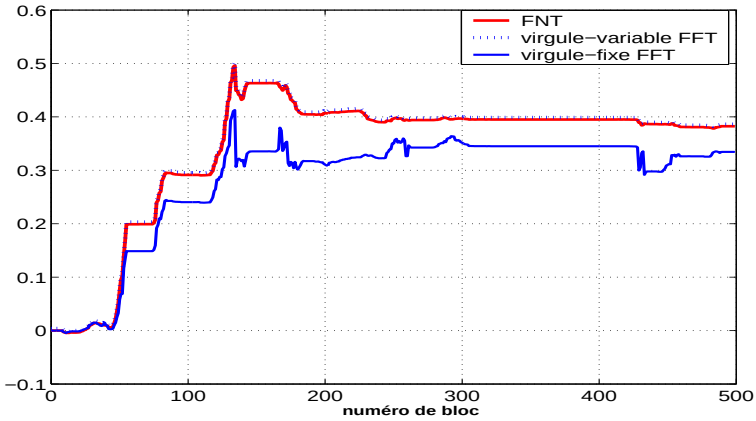

FIG. 3.7 – Evolution d'un coefficient, $(\hat{w}_k\,(21))$, du filtre

Cette figure montre que l'implantation de l'algorithme $BPNLMS++$ en virgule fixe par la mise en oeuvre de la FFT dégrade l'évolution des coefficients du filtre par rapport à la FNT.

Le deuxième critère, concerne l'évaluation de l'atténuation d'écho $ERLEC$ fournie par l'annulateur d'écho et définie, en dB, pour un bloc de longueur N et d'indice k, par :

$$ERLEC\,(k) = 10log_{10}\left(\frac{\sum_{n=(k-1)N+1}^{kN}(d_n)^2}{\sum_{n=(k-1)N+1}^{kN}(d_n-\hat{y}_n)^2}\right) \tag{3.39}$$

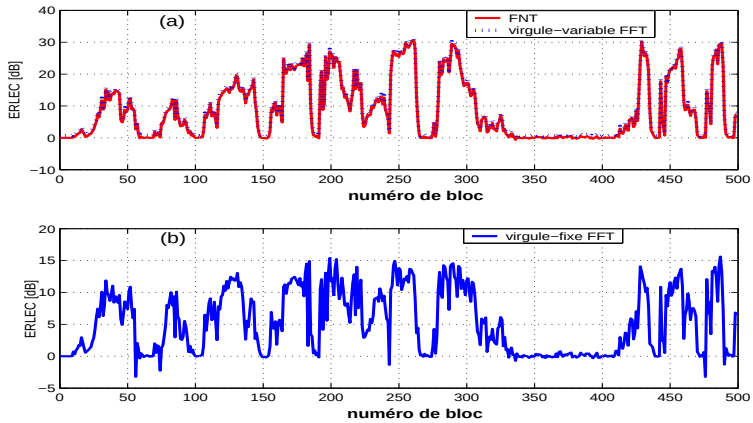

FIG. 3.8 – Atténuation fournie par l'annulateur d'écho

Cette figure montre que l'implantation de l'algorithme $BPNLMS++$ en virgule fixe, mettant en oeuvre la FFT, laisse un écho résiduel toujours audible en sortie du système d'annulation d'écho acoustique.

Comme le montrent les résultats des simulations donnés par les figures (3.7), et (3.8-b), l'exécution du filtre adapté par l'algorithme $BPNLMS++$ en utilisant la FFT avec un mot codé sur 8 bits (fixed-point) devient non pertinente du fait que l'annulateur d'écho fournit une atténuation d'écho insuffisante (moins de 15 dB).

Bien que les processeurs DSP deviennent de plus en plus puissant, les multiplications restent plus com-

plexes à réaliser par rapport aux opérations d'additions ou de décalages de bits. C'est pourquoi, nous considérerons plus particulièrement les multiplications pour l'évaluation du coût de calculs des différentes procédures.

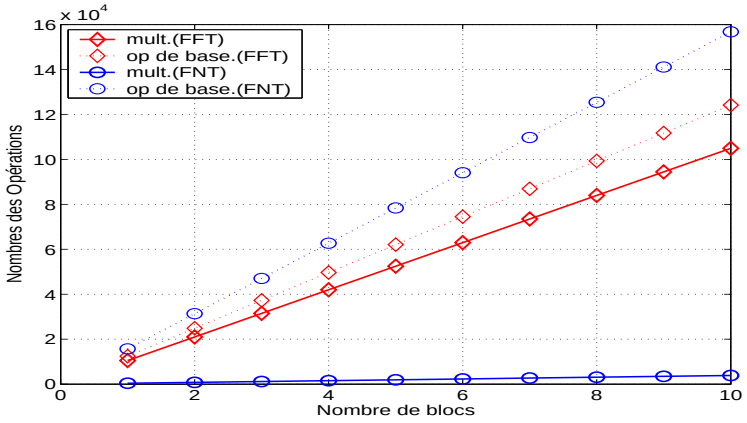

FIG. 3.9 – Opérations requises pour l'algorithme BPNLMS++ pour 10 blocs

L'ordre de grandeur entre le nombre d'opérations simples (additions réelles, soustractions réelles, décalages de bits) et de multiplications réelles pour deux implantations mettant en oeuvre la FFT et la FNT, en fonction du nombre de blocs du signal d'entrée, est donné par la figure (3.9). Le nombre d'opérations nécessaires à l'implantation de l'algorithme $BPNLMS++$ est représenté par le symbole ◯ pour la FNT et par le symbole ♢ pour la FFT. La courbe en pointillés donne les résultats des opérations simples et celle en ligne continue correspond au nombre de multiplications. Nous remarquons que, pour chaque bloc d'entrée, les nombres d'opérations sont proportionnels. La figure (3.10) donne le nombre d'opérations requises par les deux méthodes d'implantation pour un ensemble de 10 blocs.

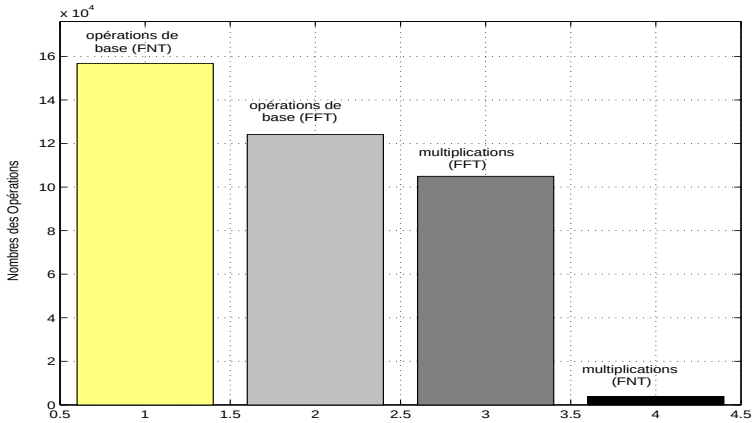

FIG. 3.10 – Opérations requises pour l'algorithme BPNLMS++ pour 10 blocs

Cette figure montre tout l'intérêt de l'utilisation de la transformée en nombres de Fermat pour la réduction du nombre de multiplications. Remarquons une diminution de plus de 95% du nombre de multiplications nécessaires à la réalisation de ce filtre par la FNT par rapport à celle mettant en oeuvre la FFT.

3.6 Conclusion

Ce chapitre nous a permis dans un premier temps de rappeler la structure en bloc des algorithmes du filtrage adaptatif dans le contexte d'annulation d'écho acoustique et dans le cas de simple parole. Nous avons ensuite proposé et étudié une variante de ces algorithmes consistant à adapter les coefficients du filtre à chaque bloc d'échantillons.

Les différents algorithmes proposés ($BNLMS$, $BPNLMS$ et $BPNLMS++$) sont comparés au niveau des différents critères, mettant en oeuvre la FFT, afin de déterminer l'algorithme qui minimise l'écho résiduel. Après cette comparaison, nous avons conclu que l'algorithme $BPNLMS++$ présente un meilleur résultat pour ce qui est de minimiser l'écho résiduel par rapport aux autres algorithmes proposés, $BNLMS$, $BPNLMS$.

Cette conclusion nous a conduit à proposer un développement de l'algorithme $BPNLMS++$ pour permettre son implantation au moyen de la FNT. Cette implantation par la FNT a permis une réduction significative du nombre de multiplications par rapport à une implantation par la FFT.

Chapitre 4

Traitement du problème d'annulation d'écho acoustique par la méthode du Sliding Généralisé

4.1 Introduction

Dans le chapitre précédent, nous avons traité l'algorithme de filtrage adaptatif $BPNLMS++$ par la mise en oeuvre de la transformée en nombres de Fermat (FNT). L'implantation de cet algorithme par la FNT, comparée à une implantation par la FFT, a permis de réduire au maximum le nombre de multiplications réelles utilisées. En revanche, elle fait augmenter le nombre d'opérations telles que les additions, les soustractions et les décalages de bits. Dans ce chapitre, notre étude consiste à réduire ce nombre d'opérations pour aboutir à un algorithme d'annulation d'écho acoustique de complexité de calcul réduite, afin de pouvoir envisager une implantation moins coûteuse sur un microprocesseur de traitement du signal.

A ce titre, nous proposons d'utiliser un algorithme développé initialement pour calculer la transformée de Fourier rapide d'une succession de séquences qui diffèrent de N échantillons l'une de l'autre, où N est une puissance de 2, et qui a pour objet de réduire la complexité de calcul par rapport à la structure de la FFT. Cette méthode est connue sous le nom de la transformée rapide de Fourier "Sliding Généralisé" ($GSFFT$: *Generalized Sliding Fast Fourier Transform*).

Pour réduire encore plus cette complexité de calcul, nous proposons d'étudier, par analogie avec la $GSFFT$, une nouvelle technique de calcul de la transformée en nombres entiers. Cette nouvelle technique, que nous intitulons Sliding Généralisé FNT ($GSFNT$), est développée pour traiter de manière efficace l'algorithme $BPNLMS++$. Cette nouvelle transformée permet une réduction maximale de la complexité de

calcul au niveau du nombre d'opérations de certaines fonctions rencontrées dans le calcul du $BPNLMS++$, telles que les produits de convolution.

Nous présenterons dans ce chapitre des mesures objectives pour comparer les performances de l'algorithme $BPNLMS++$ implanté, sous *MatLab*, par les deux méthodes, la $GSFNT$ et la FNT.

4.2 Principe général de la technique du Sliding Généralisé appliquée à la FFT

4.2.1 Introduction

Le but de cette technique de transformation est de réduire la complexité de calcul dans un cas particulier où il existe un chevauchement entre deux séquences successives à traiter. La procédure d'implantation par la $GSFFT$ est basée sur une structure de type Butterfly où la longueur de la séquence est une puissance de 2.

4.2.2 Principe d'implantation du butterfly

Les structures de la FFT de type butterfly, peuvent être adoptées pour l'exécution en temps réel d'une transformée de Fourier rapide.

Une transformée de Fourier rapide de dimension M peut être calculée, en utilisant le principe basé sur la décomposition [Cooley1965] en une somme de deux transformées de Fourier discrètes chacune de longueur $\frac{M}{2}$. En effet, en supposant que le nombre M est pair, on peut séparer la contribution des termes pairs et celle des termes impairs par :

$$X_k = \sum_{n=0}^{M-1} x_n exp\left(\frac{-j2\pi nk}{M}\right)$$

$$\begin{aligned}
&= \textstyle\sum_{n=0}^{\frac{M}{2}-1} exp\left(\frac{-j2\pi(2n)k}{M}\right) x_{2n} + \sum_{n=0}^{\frac{M}{2}-1} exp\left(\frac{-j2\pi(2n+1)k}{M}\right) x_{2n+1} \\
&= \textstyle\sum_{n=0}^{\frac{M}{2}-1} exp\left(\frac{-j2\pi(2n)k}{M}\right) x_{2n} + exp\left(\frac{-j2\pi k}{M}\right) \sum_{n=0}^{\frac{M}{2}-1} exp\left(\frac{-j2\pi(2n)k}{M}\right) x_{2n+1} \\
&= G_k + exp\left(\frac{-j2\pi k}{M}\right) H_k
\end{aligned} \tag{4.1}$$

Avec $k = 0, 1, \ldots, M-1$. Les vecteurs $\{G_k\}$ et $\{H_k\}$ sont respectivement les transformées de Fourier des séquences paires et impaires $\{x_{2n}\}_{n=0}^{\frac{M}{2}-1}$ et $\{x_{2n+1}\}_{n=0}^{\frac{M}{2}-1}$. A noter que ces deux transformées sont périodiques en k avec une période $\frac{M}{2}$.

Sachant que $exp\left(\frac{-j2\pi\left(k+\frac{M}{2}\right)}{M}\right) = -exp\left(\frac{-j2\pi k}{M}\right)$, la transformée de Fourier $\{X_k\}$ complète, de longueur M, peut alors être exprimée par :

$$\begin{cases} X_k = G_k + exp\left(\frac{-j2\pi k}{M}\right) H_k \\ X_{k+\frac{M}{2}} = G_k - exp\left(\frac{-j2\pi k}{M}\right) H_k \end{cases} \qquad k = 0, 1, \ldots, \frac{M}{2} - 1$$

Ce découpage de la séquence en deux parties peut être répété pour décomposer la transformée de longueur $M = 2^b$ en $\left\{4, 8, \ldots, \frac{M}{2}\right\}$ parties. Toute FFT d'une séquence de longueur M peut alors être calculée au moyen de $log_2 M$ décompositions et de $\frac{M}{2} log_2 M$ Butterflies.

La figure (4.1) ci-dessous illustre l'implantation symétrique, appelé Butterfly ou en papillon.

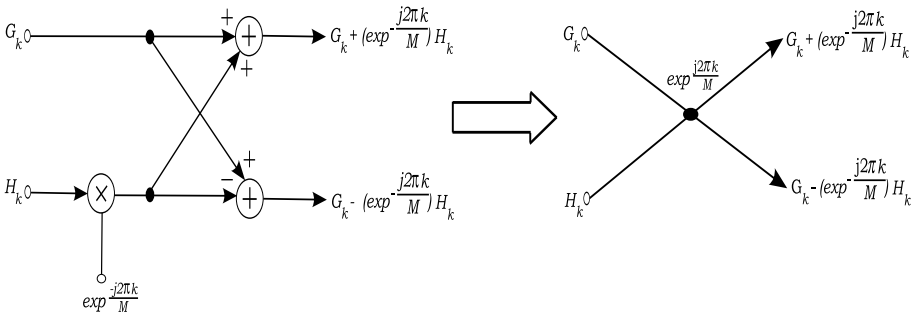

FIG. 4.1 – Décomposition Butterfly

De cette structure, qui requiert deux additions et une multiplication, est tiré le nombre d'opérations requises pour une transformée de Fourier. Comme indiqué précédemment, le nombre d'additions et de multiplications complexes mis en oeuvre pour une transformée de Fourier est respectivement de l'ordre de $M log_2 M$ et $\frac{M}{2} log_2 M$.

La figure (4.2) illustre une architecture de type Butterfly permettant le calcul de la FFT d'une séquence de longueur $M = 16$.

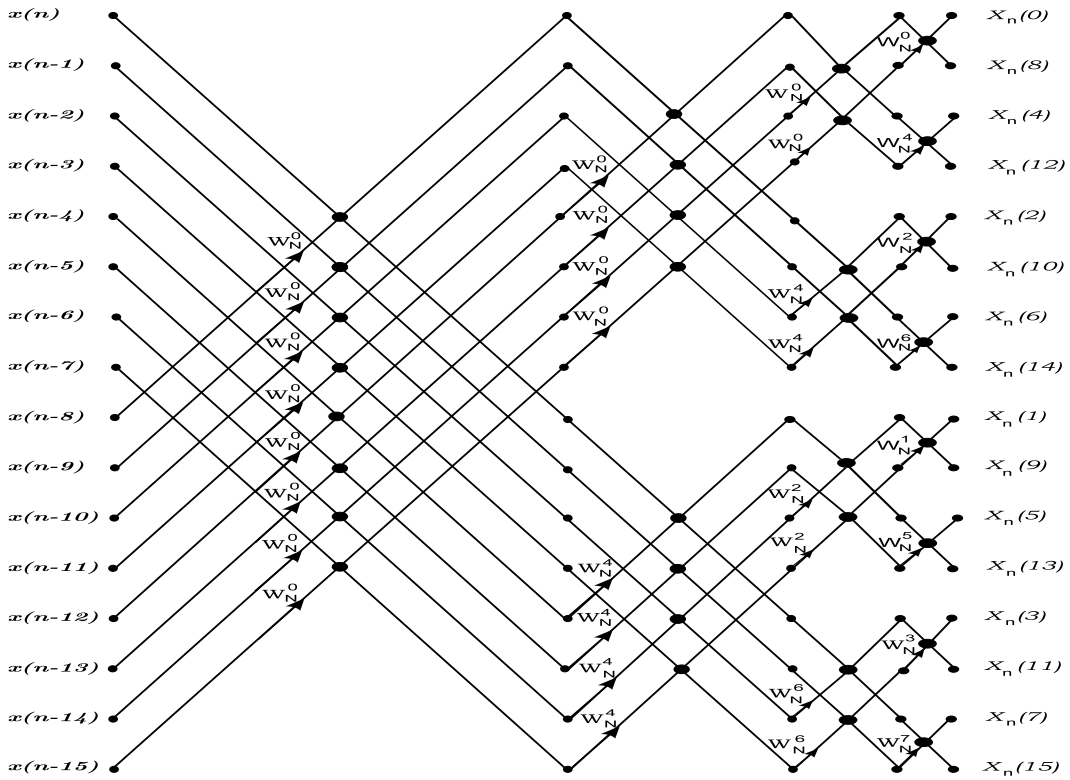

FIG. 4.2 – Architecture de la FFT pour une séquence de longueur M=16

A noter qu'à chaque décomposition, la séquence obtenue est formée de deux sous-séquences, l'une de numéro impair et l'autre de numéro pair (figure 4.3).

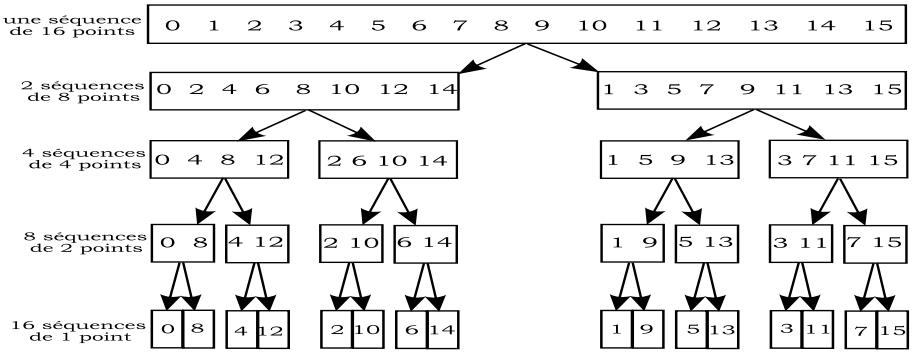

FIG. 4.3 – Décomposition par alternance d'une séquence de longueur M=16

4.2.3 Principe du Sliding Généralisé

Le principe du Sliding Généralisé a pour objet de réduire le nombre de Butterfly à calculer dans le cas de la transformée de Fourier rapide d'une séquence présentant un chevauchement avec la séquence qui la précède [Regalia1989, Stasinski1990, Farhang1992, Farhang1994].

Un exemple de chevauchement est illustré par la figure (4.4). Il montre une succession de séquences de

longueur $M = 16 = 2^4$, qui diffèrent de $N = 2^0 = 1$ échantillon l'une de l'autre. Par conséquent, d'un instant k à instant $k+1$, nous aurons à faire le calcul pour un échantillon, les autres sont déjà connus et mis en mémoire à l'instant précédent k.

De manière générale, la technique du Sliding Géneralisé, basée sur la structure de Butterfly, traite, à un instant k, uniquement les N nouveaux échantillons, les autres échantillons étant calculés et mémorisés à l'instant k.

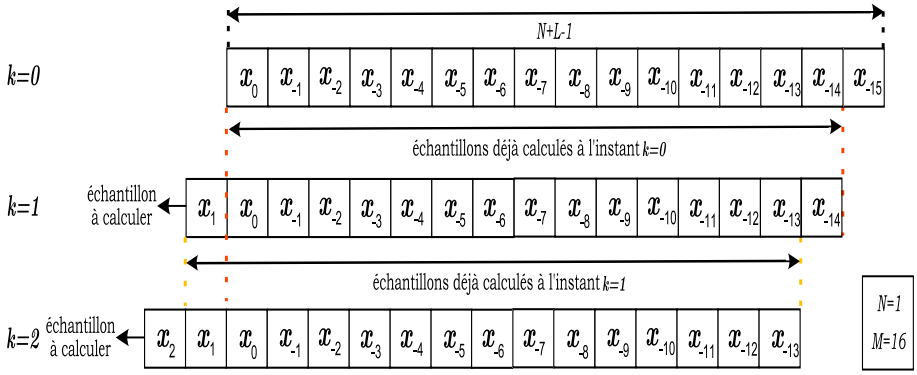

FIG. 4.4 – Principe de chevauchement pour M=16 ; N=1

La figure (4.5) montre le principe d'implantation du Butterfly par la technique du Sliding Généralisé appliquée à la FFT [Farhang1994].

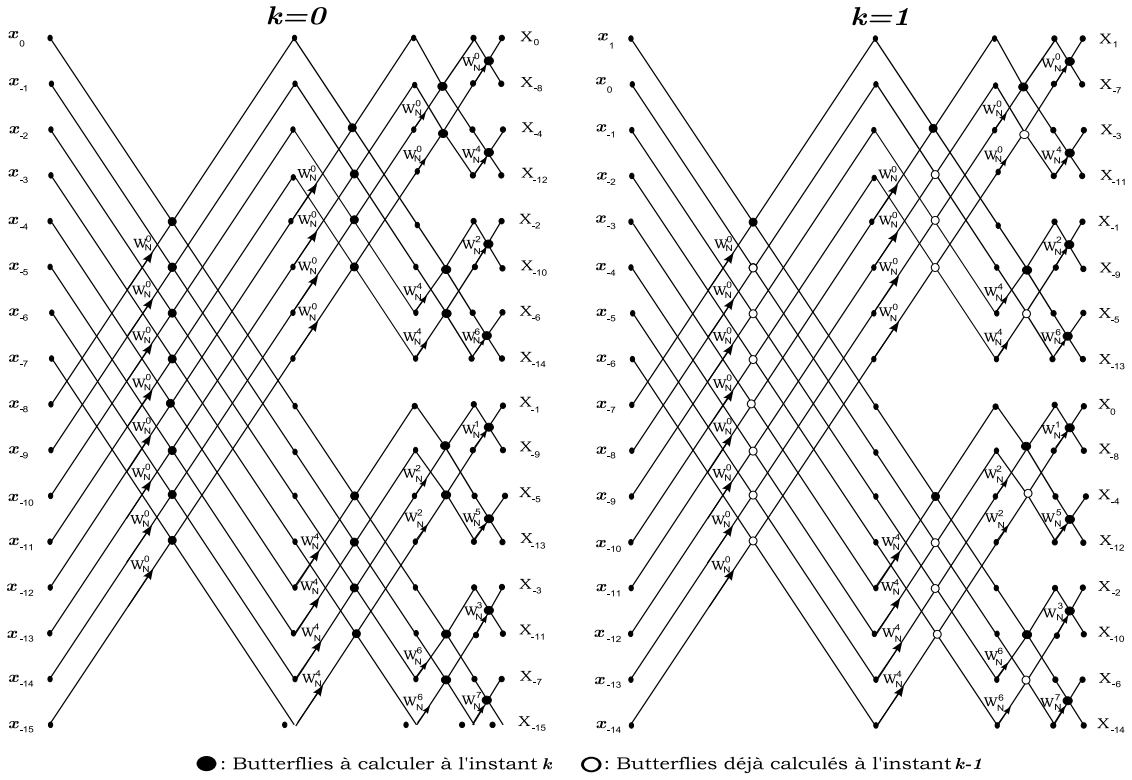

FIG. 4.5 – Principe de la SFFT pour M=16

Le symbole • représente les Butterflies à calculer à un instant donné et le symbole ∘ représente les Butterflies qui sont déjà calculées à l'instant précédent.

Cet exemple où $N = 1$ montre tout l'intérêt d'utiliser la technique du Sliding pour calculer une FFT d'une séquence de longueur $M = 16$. Cette structure de calcul de la FFT par la technique du Sliding montre que le nombre n_b de Butterfly à calculer, en passant de $k = 0$ à $k = 1$, est de 16, alors que par la FFT, ce nombre est de 32. Nous pouvons donc conclure que par rapport à la FFT, l'application de la $SFFT$ en temps discret réduit le nombre de Butterfly et par conséquence la complexité de calcul.

Ce cas particulier où $N = 1$ peut être généralisé à deux séquences consécutives qui diffèrent de N échantillons, avec $N > 1$. Ce cas général, illustré par la figure (4.6) et dont la structure de principe est représentée par la figure (4.7), permet de calculer la $GSFFT$ d'une séquence en ne traitant à chaque itération que les N nouveaux échantillons qui la distingue de la séquence précédente, les autres échantillons étant traités à l'instant précédent. Cette structure permet de calculer le nombre n_b de Butterflies nécessaire au calcul de la $GSFFT$ d'une séquence de longueur M. Ce nombre est donné par : [Gazor1992]

$$n_b\,(GSFFT) = \frac{M}{2}\,(log_2 N + 2) - N \tag{4.2}$$

De cette relation, nous en déduisons que :

• Si $N = 1$, cela correspond à la $SFFT$ et le nombre de Butterflies est réduit au maximum [Farhang1992].

• Si $N \geq \frac{M}{2}$, le nombre de Butterflies est identique à celui demandé par le calcul d'une FFT.

Pour réduire le coût de calcul en utilisant la méthode du Sliding Généralisé, il faut donc que deux séquences consécutives diffèrent d'un nombre N d'échantillons tels que : $1 \leq N < \frac{M}{2}$.

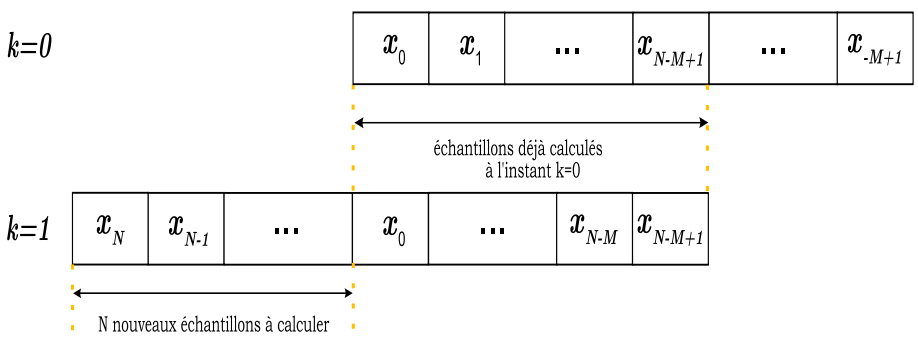

FIG. 4.6 – Principe du Sliding Généralisé pour M=N+L-1 et pour un chevauchement de L-1 échantillons

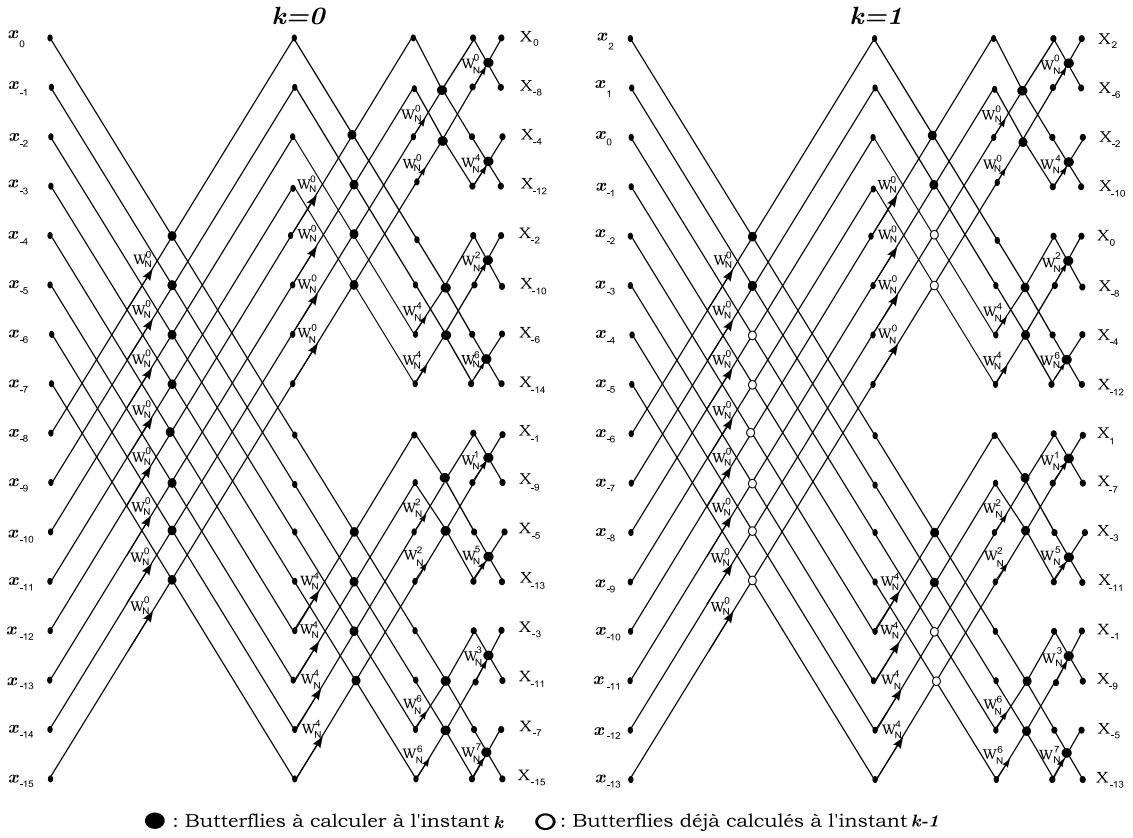

FIG. 4.7 – Principe de la GSFFT pour M=16 et N=2

4.2.4 Complexité de calcul

Nous avons vu dans le paragraphe précédent que l'utilisation de la technique du Sliding généralisé appliquée à la transformée de Fourier rapide $(GSFFT)$, comparée à la FFT, permet de réduire le nombre de Butterfly à réaliser pour une séquence à traiter, et par conséquent, le nombre des opérations mises en oeuvre dans cette réalisation. Cette réduction est constatée dans le cas où deux séquences successives diffèrent de N échantillons, avec $1 \leq N < \frac{M}{2}$. En effet, le nombre des Butterflies pour la $GSFFT$ d'une séquence de longueur M est de l'ordre de $\frac{M}{2}(log_2 N + 2) - N$ alors que, pour la FFT conventionnelle de cette même séquence, le nombre des Butterflies est de l'ordre de $\frac{M}{2} log_2 M$. Comme nous avons vu précédemment que chaque Butterfly requiert deux additions complexes et une multiplication complexe, nous pouvons alors conclure que :

• Chaque transformée $GSFFT$ exige $\frac{M}{2}(log_2 N + 2) - N$ multiplications complexes et $M(log_2 N + 2) - 2N$ additions complexes.

• Pour calculer la FFT de la même séquence, le coût de calcul est de l'ordre de $\frac{M}{2} log_2 M$ multiplications et $M log_2 M$ additions.

Les niveaux de complexité de calcul des deux transformées $GSFFT$ et FFT sont déterminés par le nombre de Butterflies nécessaires à leur réalisation. Leur évaluation est donnée par les relations suivantes :

$$n_b\left(GSFFT\right) = \frac{M}{2}\left(log_2 N + 2\right) - N \tag{4.3}$$

$$n_b\left(FFT\right) = \frac{M}{2}\left(log_2 M\right) \tag{4.4}$$

Nous pouvons en déduire le rendement de la complexité de la $GSFFT$, comparée à celle de la FFT. Il est donné par : [Gazor1992]

$$\begin{aligned}\eta_{GSFFT} = \frac{n_b(FFT)}{n_b(GSFFT)} &= \frac{\frac{M}{2}(log_2 M)}{\frac{M}{2}(log_2 N+2)-N} \\ &= \frac{\frac{M}{2}(log_2 M)}{\frac{M}{2}\left(log_2 N+2-\frac{2N}{M}\right)} \\ &= \frac{log_2 M}{log_2 N+2-\frac{2N}{M}}\end{aligned}$$

Si nous considérons le cas où $M >> N$, le rapport $\left(\frac{2N}{M}\right) \longrightarrow 0$ et nous obtenons :

$$\eta_{GSFFT} \approx \frac{log_2 M}{log_2 N + 2} = \frac{log_2 M}{log_2 N + log_2 4} = \frac{log_2 M}{log_2 4N} = log_{4N} M > 1 \tag{4.5}$$

Par conséquent : $n_b\left(GSFFT\right) < n_b\left(FFT\right)$. La transformée $GSFFT$ présente donc un avantage au niveau de la complexité de calcul, puisqu'elle nécessite moins de Butterflies et donc moins d'opérations (multiplications, additions, soustractions, etc...) que la FFT conventionnelle.

La figure (4.8) fournit une comparaison en terme de nombre de Butterflies nécessaires à la réalisation des deux types de transformées en fonction de N. Cette comparaison est réalisée à partir d'une séquence de longueur $M = 1024$.

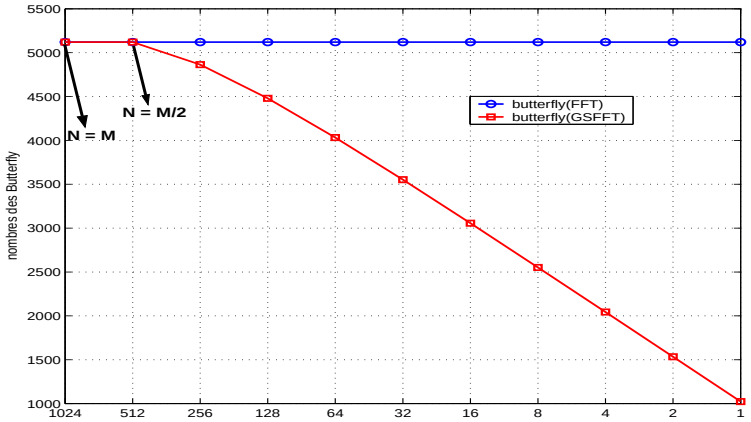

FIG. 4.8 – Nombre de Butterfly en fonction du N et pour M=1024

Cette figure (4.8) montre que, quand N diminue, le nombre de Butterflies diminue pour la structure $GSFFT$ alors qu'il reste constant pour la FFT. A partir de cette figure, nous présentons sur la figure (4.9), la complexité de calcul, déterminée par le nombre de multiplications et d'additions en fonction de N,

nécessaires dans le calcul des deux transformées, FFT et $GSFFT$.

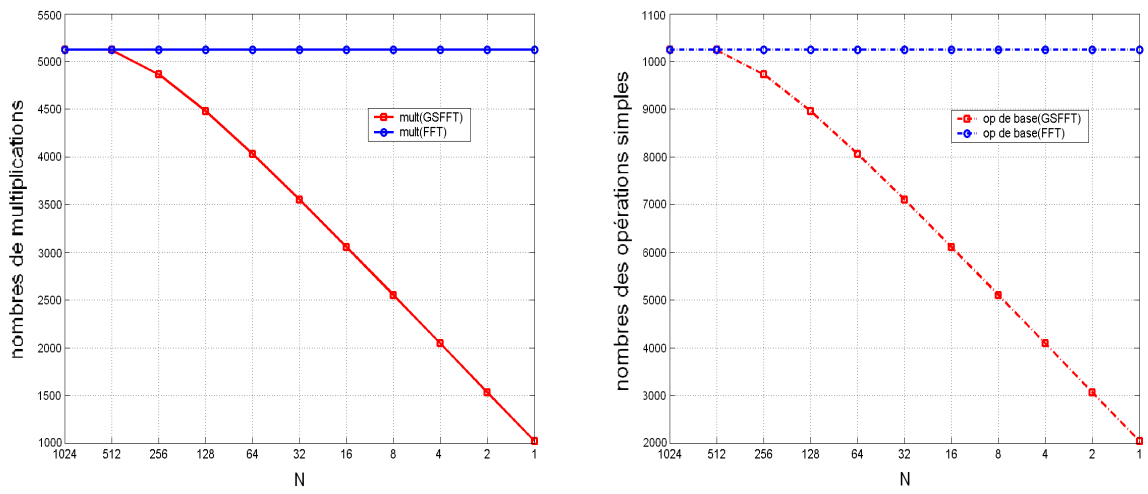

FIG. 4.9 – Complexité de calcul en fonction de N et pour M=1024

Sur cette figure, les courbes en pointillés représentent le nombre d'opérations simples telles que les additions et les soustractions et les courbes en ligne pleine représentent le nombre de multiplications. Les courbes montrent que, plus N diminue, plus le nombre de calculs diminue pour la $GSFFT$, et plus l'avantage de l'implantation par la $GSFFT$ sera marqué.

Les figures, (4.8) et (4.9), nous montrent la nécessité de choisir N tel que $1 \leq N < \frac{M}{2}$ afin de réduire la complexité de calcul.

Pour différentes valeurs de M et de N, le tableau (4.1) donne une idée sur l'économie de calcul $100.\left(1-\eta_{GSFFT}\right)$, en %, réalisée durant le traitement d'une séquence lorsque l'on remplaçe la FFT par la $GSFFT$.

$N \diagdown M$	16	32	64	128	256	512	1024
1	53.12	61.25	67.19	71.65	75.10	77.82	80.02
2	31.25	42.50	51.04	57.59	62.70	66.75	70.04
4	12.50	25.00	35.42	43.75	50.39	55.73	60.08
8	0.00	10.00	20.83	30.36	38.28	44.79	50.16
16	0.00	0.00	8.33	17.86	26.56	34.03	40.31
32	0.00	0.00	0.00	7.14	15.62	23.61	30.62
64	0.00	0.00	0.00	0.00	6.25	13.89	21.25
128	0.00	0.00	0.00	0.00	0.00	5.56	12.50
256	0.00	0.00	0.00	0.00	0.00	0.00	5.00
512	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TAB. 4.1 – Economie de calcul en % réalisée lorsque l'on remplace une FFT par une GSFFT

Les résultats de ce tableau montrent que pour $N << M$, l'économie de calcul varie de 50% à 80%. Mais quand N augmente, c'est à dire lorsque la longueur de chevauchement diminue, l'économie de calcul diminue jusqu'à s'annuler lorsque $N \geq \frac{M}{2}$.

De la même manière que l'on a défini la transformée de Fourier rapide inverse, nous définissons la transformée Sliding Généralisé inverse ($IGSFFT$) d'une séquence de longueur M. Cette transformée inverse requiert le même nombre de Butterflies que celui demandé par la $GSFFT$.

Une fois cette technique du Sliding Généralisé définie, nous allons maintenant présenter l'algorithme correspondant afin de pouvoir l'implanter au moyen de la FFT et de la NTT.

4.3 Algorithme "Sliding Généralisé"

Dans ce qui précède, nous avons défini le principe du Sliding Généralisé et son intérêt dans le traitement des séquences numériques. Il est donc nécessaire de définir un algorithme pour cette technique. Pour bien expliquer cette technique, nous allons présenter sa mise en oeuvre dans le calcul de la FFT. L'algorithme ainsi obtenu, connu sous le nom du Generalized Sliding Fast Fourier Transform ($GSFFT$) sera développé.

4.3.1 Algorithme SFFT

Avant de présenter l'algorithme $GSFFT$ dans le cas général où $N > 1$, nous présentons tout d'abord l'algorithme dans le cas particulier où $N = 1$. L'algorithme ainsi obtenu, intitulé $SFFT$ (*Sliding Fast Fourier Transform*), consiste à traiter, à un instant k, une séquence $X_k = \begin{bmatrix} x_k & x_{k-1} & \dots & x_{k-M+1} \end{bmatrix}^T$ de longueur $M = 2^{q'}$, qui diffère de la précédente $X_{k-1} = \begin{bmatrix} x_{k-1} & x_{k-2} & \dots & x_{k-M} \end{bmatrix}^T$, traitée à l'instant $k-1$, d'un échantillon [Gazor1992]. Pour cela, nous définissons un vecteur U_k comprenant q' éléments qui nous servira par la suite pour déterminer la $SFFT$ de la séquence X_k. Ce vecteur est défini par :

$$U_k = \begin{bmatrix} U_{0,k}^T & U_{1,k}^T & \dots & U_{q',k}^T \end{bmatrix}^T \tag{4.6}$$

Les différents éléments $U_{i,k}$ de ce vecteur sont de longueur 2^i, où $i = 0, 1, \dots, q'$ et ils sont définis par une relation de récurrence donnée par :

$$\begin{cases} U_{0,k} = x_k \\ U_{i+1,k} = H_i \begin{bmatrix} I_{2^i} & I_{2^i} \\ I_{2^i} & -I_{2^i} \end{bmatrix} \begin{bmatrix} U_{i,k} \\ V_i U_{i,k-2^{q'-i-1}} \end{bmatrix} \text{ pour } 0 \leq i < q' \end{cases} \tag{4.7}$$

H_i étant une matrice de permutation $\left(2^{i+1} \times 2^{i+1}\right)$ donnée par :

$$H_i = \begin{bmatrix} E_1 & E_{1+2^i} & E_2 & E_{2+2^i} & \dots & E_{2^{i+1}} \end{bmatrix} \tag{4.8}$$

Les différents éléments E_j de cette matrice représentent la $j^{ème}$ colonne de la matrice identité $I_{2^{i+1}}$ de

taille $\left(2^{i+1} \times 2^{i+1}\right)$.

V_i est une matrice diagonale définie par :

$$V_i = diag\left(\begin{array}{cccc} 1, & W^{\sigma(1)}, & \ldots, & W^{\sigma\left(2^i-1\right)} \end{array} \right) \tag{4.9}$$

où $W = exp\left(-j2\pi/M\right)$ et $\sigma\left(r\right)$ est le bit reverse de r, $r = 0, 1, \ldots, 2^i - 1$.

Pour $i = q' - 1$, nous avons à calculer $U_{i+1,k} = U_{q',k} = FFT\left(X_k\right) = SFFT\left(X_k\right)$. Ce vecteur, de longueur $2^{q'}$, est calculé en utilisant l'équation (4.7) selon la relation suivante :

$$U_{q',k} = H_{q'-1} \left[\begin{array}{cc} I_{2^{q'-1}} & I_{2^{q'-1}} \\ I_{2^{q'-1}} & -I_{2^{q'-1}} \end{array} \right] \left[\begin{array}{c} U_{q'-1,k} \\ V_{q'-1} U_{q'-1,k-1} \end{array} \right] = FFT\left(X_k\right) \tag{4.10}$$

Le calcul du vecteur $U_{i+1,k} = U_{q',k}$ faisant appel au calcul du vecteur $U_{i,k-2^{q'-i-1}}$, déjà calculé et mémorisé à l'instant $k - 2^{q'-i-1}$, le traitement par la $SFFT$ permet de réduire la charge de calcul par rapport à un traitement par la FFT.

Exemple

Pour illustrer l'utilisation de l'algorithme $SFFT$, un exemple est traité pour une séquence de longueur $M = 2^{q'} = 4$.

Si à l'instant $k = 1$, $X_1 = \left[\begin{array}{cccc} x_1 & x_0 & x_{-1} & x_{-2} \end{array} \right]^T = \left[\begin{array}{cccc} 1 & 2 & 0 & 1 \end{array} \right]^T$, la FFT de la séquence X_1 est égale à :

$$FFT\left(X_1\right) = FFT\left(\left[\begin{array}{cccc} 1 & 2 & 0 & 1 \end{array} \right]^T \right) = \left[\begin{array}{c} 4 \\ 1-j \\ -2 \\ 1+j \end{array} \right]$$

où j est le nombre complexe imaginaire pur.

D'après l'algorithme $SFFT$, défini avant, le vecteur U_1 s'écrit alors :

$$U_1 = \left[\begin{array}{ccc} U_{0,1}^T & U_{1,1}^T & U_{2,1}^T \end{array} \right]^T$$

$$U_{0,1} = 1$$

Pour $i = 0$:

$$U_{1,1} = H_0 \begin{bmatrix} I_{2^0} & I_{2^0} \\ I_{2^0} & -I_{2^0} \end{bmatrix} \begin{bmatrix} U_{0,1} \\ V_0 U_{0,-1} \end{bmatrix}$$

La matrice de permutation $\left(2^1 \times 2^1\right)$ est donnée par :

$$H_0 = \begin{bmatrix} E_1 & E_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Le vecteur $U_{0,-1} = 0$, parce que la séquence X_{-1} à l'instant $k = -1$ est définie par :

$$X_{-1} = \begin{bmatrix} 0 & 1 & x_{-3} & x_{-4} \end{bmatrix}^T$$

La matrice V_0 obtenue se réduit à :

$$V_0 = 1$$

Le vecteur $U_{1,1}$ s'écrit donc en remplaçant V_0, $U_{0,-1}$, et $U_{0,1}$ par leurs valeurs :

$$U_{1,1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Pour $i = q' - 1 = 1$:

$$U_{2,1} = H_1 \begin{bmatrix} I_{2^1} & I_{2^1} \\ I_{2^1} & -I_{2^1} \end{bmatrix} \begin{bmatrix} U_{1,1} \\ V_1 U_{1,0} \end{bmatrix} \tag{4.11}$$

$U_{1,1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $V_1 = diag\begin{pmatrix} 1, & W^{\sigma(1)} \end{pmatrix}$, avec $\sigma(1) = 1$, $W^{\sigma(1)} = exp(-j2\pi/M) = -j$, donc

$V_1 = diag\left(\begin{bmatrix} 1 & -j \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 0 & -j \end{bmatrix}$, et la matrice de permutation H_1 est définie par :

$$H_1 = \begin{bmatrix} E_1 & E_3 & E_2 & E_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Il reste à calculer le vecteur $U_{1,0}$ qui est défini à l'instant $k = 0$ et pour $i = 1$. A l'instant $k = 0$, la séquence d'entrée est définie par :

$$X_0 = \begin{bmatrix} 2 & 0 & 1 & x_{-3} \end{bmatrix}^T$$

Pour $i = 0$, le vecteur $U_{1,0}$ qu'on cherche est calculé selon la relation :

$$U_{1,0} = H_0 \begin{bmatrix} I_{2^0} & I_{2^0} \\ I_{2^0} & -I_{2^0} \end{bmatrix} \begin{bmatrix} U_{0,0} \\ V_0 U_{0,-2} \end{bmatrix}$$

$U_{0,0} = 2$, $V_0 = diag\,(1) = 1$ et $U_{0,-2} = 1$.

En remplaçant V_0 et $U_{0,-2}$ par leur valeur, le vecteur $U_{1,0}$ devient :

$$U_{1,0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Si on remplace le vecteur $U_{1,0}$ par sa valeur dans la relation (4.11), on obtient :

$$U_{2,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \begin{bmatrix} 1 & 0 \\ 0 & -j \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \end{bmatrix}$$

Soit au final, $U_{2,1} = \begin{bmatrix} 4 \\ 1-j \\ -2 \\ 1+j \end{bmatrix}$.

Nous avons donc : $SFFT\,(X_1) = \begin{bmatrix} 4 \\ 1-j \\ -2 \\ 1+j \end{bmatrix}$

Ce résultat est identique à celui obtenu par le calcul de la FFT de X_1.

Après cette présentation de l'algorithme $SFFT$ qui traite le cas particulier, où $N = 1$, nous allons développer l'algorithme $GSFFT$ qui traite le cas général où $N > 1$.

4.3.2 Algorithme GSFFT

Pour calculer la $GSFFT$ d'une séquence de longueur M, nous considérons le cas où nous avons, à chaque itération, N nouveaux échantillons ($N > 1$) [Gazor1992]. La séquence d'entrée X, de longueur M à l'instant k est définie par :

$$X_k = \begin{bmatrix} x_k & x_{k-1} & \dots & x_{k-M+1} \end{bmatrix}^T \tag{4.12}$$

$$= \begin{bmatrix} Y_k^T & Y_{k-1}^T & \dots & Y_{k-N'+1}^T \end{bmatrix}^T$$

où $Y_k = \begin{bmatrix} x_k & x_{k-1} & \dots & x_{k-N+1} \end{bmatrix}^T$ est un vecteur de longueur N qui représente l'ensemble des nouveaux échantillons à l'instant k, $k \in \{0, 1, \dots, N'-1\}$.

Si $N = 2^n$ représente le nombre de nouveaux échantillons et $N' = 2^m$ le nombre de vecteurs Y_k, la longueur de la séquence X est alors définie par $M = N'.N = 2^{n+m} = 2^{q'}$.

La technique de la $GSFFT$ consiste à traiter uniquement les échantillons représentés par le nouveau vecteur Y_k.

Comme pour la $SFFT$, nous définissons un vecteur $U_k = \begin{bmatrix} U_{0,k}^T & U_{1,k}^T & \dots & U_{(n+m),k}^T \end{bmatrix}^T$ de q' éléments, calculés par une relation de récurrence définie par l'équation (4.13) et qui permet de calculer la $GSFFT$ d'une séquence X_k :

$$\begin{cases} U_{0,k} = Y_k \\ U_{i+1,k} = (H_{2^{i+1}} \otimes I_{2^n}) \cdot \left(\left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes I_{2^{n+i}} \right) \begin{bmatrix} U_{i,k} \\ V_{i,n} U_{i,k-2^{m-i-1}} \end{bmatrix} \right) \text{ pour } 0 \le i \le m-1 \end{cases} \tag{4.13}$$

$$= \mathcal{H}_{2^{n+i+1}} . \mathcal{R}_{2^{n+i+1}} . \begin{bmatrix} U_{i,k} \\ V_{i,n} U_{i,k-2^{m-i-1}} \end{bmatrix}$$

$\mathcal{H}_{2^{n+i+1}}$ et $\mathcal{R}_{2^{n+i+1}}$ sont des matrices qui sont données par :

$$\mathcal{H}_{2^{n+i+1}} = (H_{2^{i+1}} \otimes I_{2^n}) \tag{4.14}$$

$$\mathcal{R}_{2^{n+i+1}} = \left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes I_{2^{n+i}} \right) \tag{4.15}$$

$H_{2^{i+1}}$ est une matrice de permutation de taille $\left(2^{i+1} \times 2^{i+1}\right)$ donnée par l'équation (4.8). Le symbole $\otimes$ désigne le produit de Kronecker. $V_{i,n}$ est une matrice carrée d'ordre 2^{n+i} définie par :

$$V_{i,n} = V_i \otimes I_{2^n} \tag{4.16}$$

Le vecteur V_i est donné par l'équation (4.9).

La matrice $\mathcal{H}_{2^{n+i+1}}$ est donnée par :

$$\mathcal{H}_{2^{n+i+1}} = [H_{2^{i+1}} \otimes I_{2^n}] = \begin{bmatrix} e_{1,1}I_{2^n} & e_{1,1+2^i}I_{2^n} & \dots & \dots & e_{1,2^{i+1}}I_{2^n} \\ e_{1+2^i,1}I_{2^n} & \ddots & \dots & \dots & e_{1+2^i,2^{i+1}}I_{2^n} \\ \vdots & \dots & e_{p,s}I_{2^n} & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ e_{2^{i+1},1}I_{2^n} & e_{2^{i+1},1+2^i}I_{2^n} & \dots & \dots & e_{2^{i+1},2^{i+1}}I_{2^n} \end{bmatrix} \tag{4.17}$$

où, $e_{p,s}$ désignent les éléments de la matrice $H_{2^{i+1}}$ calculés par :

$$\left\{ \begin{array}{lll} e_{p,s} = 1 & si & p = s \\ e_{p,s} = 0 & si & p \neq s \end{array} \right\} \tag{4.18}$$

De la même façon, la matrice $\mathcal{R}_{2^{n+i+1}}$ peut s'écrire sous la forme :

$$\mathcal{R}_{2^{n+i+1}} = \left[\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes I_{2^{n+i}} \right] = [S \otimes I_{2^{n+i}}]$$

$$= \begin{bmatrix} e_{1,1}S & e_{1,2}S & \dots & \dots & e_{1,2^{n+i}}S \\ e_{2,1}S & \ddots & \dots & \dots & e_{2,2^{n+i}}S \\ \vdots & \dots & e_{p,s}S & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ e_{2^{n+i},1}S & e_{2^{n+i},2}S & \dots & \dots & e_{2^{n+i},2^{n+i}}S \end{bmatrix} \tag{4.19}$$

Par définition des éléments $e_{p,s}$, la matrice $\mathcal{R}_{2^{n+i+1}}$ peut s'écrire :

$$\mathcal{R}_{2^{n+i+1}} = \begin{bmatrix} S & 0 & \dots & \dots & 0 \\ 0 & \ddots & \dots & \dots & 0 \\ \vdots & \dots & S & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & 0 \\ 0 & 0 & \dots & 0 & S \end{bmatrix} \tag{4.20}$$

Dans le cas où $m \leq i < n + m$, le vecteur $U_{i,k}$ est donné par :

$$U_{i,k} = \begin{bmatrix} U_{i,k}(0) & U_{i,k}(1) & \dots & U_{i,k}(M-1) \end{bmatrix}^T \tag{4.21}$$

Les différents éléments $U_{i,k}(c)$ sont calculés selon la relation :

$$\begin{bmatrix} U_{i+1,k}(c) \\ U_{i+1,k}\left(c+2^{n+m-i-1}\right) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} U_{i,k}(c) \\ V_{i,c}U_{i,k}\left(c+2^{n+m-i-1}\right) \end{bmatrix} = \begin{bmatrix} 1 & V_{i,c} \\ 1 & -V_{i,c} \end{bmatrix} \begin{bmatrix} U_{i,c}(c) \\ U_{i,k}\left(c+2^{n+m-i-1}\right) \end{bmatrix} \tag{4.22}$$

avec $V_{i,c} = W^{\sigma_i(c)}$, ceci quel que soit c tel que $0 \leq \langle c \rangle_{M/2^i} < M/2^{i+1}$ et $0 \leq c < M$.

$\sigma_i(c)$ est le bit reverse de $\langle (c-1) \rangle_{M/2^i}$.

La $GSFFT$ de la séquence X_k est alors donnée par le vecteur $U_{i+1,k}$ pour $i = n+m-1$, c'est-à-dire $U_{n+m,k}$. Comme pour la FFT (figure 4.2), les couples d'échantillons de la séquence $U_{n+m,k}$ doivent être choisis selon un ordre particulier. Cette incrémentation particulière est appelée "retenue inverse" (reverse carry). La structure de l'algorithme $GSFFT$ ne change pas, simplement ce sont les index d'échantillons du vecteur $U_{n+m,k}$ qui changent. Les échantillons de ce vecteur sont ordonnés en "reverse carry". A partir de cette incrémentation, nous avons donc :

$$GSFFT(X_k) = FFT(X_k) = U_{n+m,k}$$

Exemple

Pour illustrer l'utilisation de l'algorithme $GSFFT$, nous allons prendre un exemple simple pour mettre en évidence la $GSFFT$ et la comparer à la FFT. Pour cela, nous prenons l'exemple d'une séquence $X_1 = \begin{bmatrix} x_1 & x_0 & x_{-1} & x_{-2} \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 0 & 2 \end{bmatrix}^T$ de longueur $M = 2^n.2^m = N.N' = 4$ qui diffère de la séquence précédente $X_0 = \begin{bmatrix} x_{-1} & x_{-2} & x_{-3} & x_{-4} \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & x_{-3} & x_{-4} \end{bmatrix}^T$ de $N = 2^n = 2$ échantillons.

A partir de cette séquence X_1, le vecteur Y_1 composé de nouveaux échantillons est défini par :

$$Y_1 = \begin{bmatrix} x_1 & x_0 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$$

A partir de l'équation (4.13) définit l'algorithme $GSFFT$ nous pourrons écrire :

$$\text{pour } i = 0 \begin{cases} U_{0,1} = Y_1 \\ U_{1,1} = \mathcal{H}_{2^2}.\mathcal{R}_{2^2}.\begin{bmatrix} U_{0,1} \\ V_{0,1}U_{0,0} \end{bmatrix} \end{cases}$$

$U_{0,0} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ et $V_{0,1} = V_0 \otimes I_{2^1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Si on remplace le vecteur $U_{0,0}$ et la matrice $V_{0,1}$ par leur valeur on obtient :

$$U_{1,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 0 \end{bmatrix}$$

pour $i = 1$, les différents éléments du vecteur $U_{2,1}$, sont calculés par :

$$\begin{bmatrix} U_{2,1}(c) \\ U_{2,1}(c+1) \end{bmatrix} = \begin{bmatrix} 1 & V_{1,c} \\ 1 & -V_{1,c} \end{bmatrix} \begin{bmatrix} U_{1,1}(c) \\ U_{1,1}(c+1) \end{bmatrix} \tag{4.23}$$

$0 \leq \langle c \rangle_2 < 1$ *et* $0 \leq c < 4$, dans ce cas, $c = 0, 1, 2, 3$; et comme $0 \leq \langle c \rangle_2 < 1$, alors l'indice c prend la valeur 0 ou 2.

Dans le premier cas $(c = 0)$, $\sigma_i(0) = 0$, $V_{1,0} = 1$, et l'équation (4.23) s'écrit :

$$\begin{bmatrix} U_{2,1}(0) \\ U_{2,1}(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ -3 \end{bmatrix}$$

Dans le second cas $(c = 2)$, $\sigma_i(2) = 1$, $V_{1,1} = -j$ et l'équation (4.23) s'écrit :

$$\begin{bmatrix} U_{2,1}(2) \\ U_{2,1}(3) \end{bmatrix} = \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Le vecteur $U_{n+m,k} = U_{2,1}$ s'écrit donc :

$$U_{2,1} = \begin{bmatrix} U_{2,1}(0) & U_{2,1}(1) & U_{2,1}(2) & U_{2,1}(3) \end{bmatrix}$$

En appliquant la "retenue inverse" au vecteur $U_{2,1}$ nous obtenons la FFT de X_1 qui est aussi la $GSFFT$ de X_1.

$$U_{2,1} = \begin{bmatrix} U_{2,1}(0) & U_{2,1}(2) & U_{2,1}(1) & U_{2,1}(3) \end{bmatrix} = \begin{bmatrix} 5 & 1 & -3 & 1 \end{bmatrix} = FFT(X_1) = GSFFT(X_1)$$

La différence entre les deux transformées se situe au niveau de la complexité de calcul qui est plus réduite dans le cas de la $GSFFT$.

4.4 Procédure de calcul de la convolution circulaire par la GSFFT

De nombreuses fonctions comme le produit de convolution sont utilisées dans les algorithmes d'annulation d'écho acoustique. Pour cela, nous présenterons une procédure de calcul du produit de convolution au moyen de la technique du Sliding Généralisé appliquée à la FFT $(GSFFT)$.

Le calcul de la convolution $h_k = x_k * y_k$ de deux séquences x_k et y_k de M échantillons, où M est une puissance de deux, nécessite, en utilisant la Propriété de Convolution Circulaire, le calcul de trois transformées, dont deux directes $(GSFFT)$ et une inverse $(IGSFFT)$. En effet, si T_{GSFFT} désigne la $GSFFT$ d'une séquence et T_{IGSFFT} désigne la $GSFFT$ inverse, nous pouvons écrire :

$$h_k = x_k * y_k \tag{4.24}$$

$$T_{GSFFT}(x_k) \bullet T_{GSFFT}(y_k) = H_k \tag{4.25}$$

$$X_k \bullet Y_k = H_k \tag{4.26}$$

où X_k, Y_k et H_k désignent respectivement la $GSFFT$ des séquences x_k, y_k et h_k et $\bullet$ désigne la multiplication terme à terme. La $GSFFT$ inverse de la séquence H_k est donnée par :

$$h_k = T_{IGSFFT}(H_k) \tag{4.27}$$

Le principe de cette procédure de calcul de convolution circulaire est résumé par la figure (4.10).

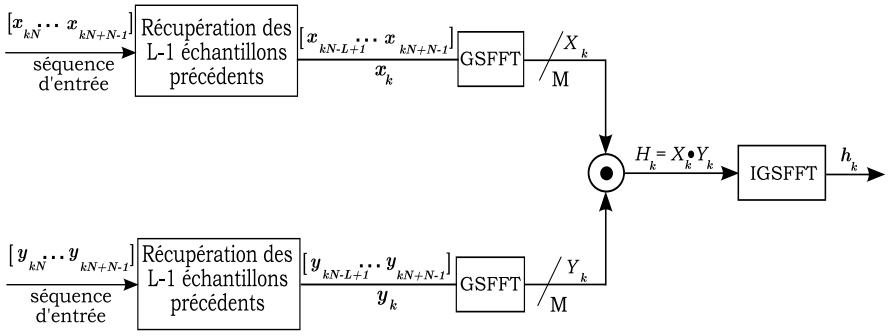

FIG. 4.10 – Description d'un calcul de convolution par la GSFFT

Le passage du domaine fréquentiel au domaine temporel nécessite l'application de la transformée inverse $IGSFFT$ de la séquence H_k. Cette transformée inverse requiert le même nombre des Butterflies que la transformée inverse de la FFT, parce que la séquence à transformer, H_k, ne contient pas de chevauchement. Par conséquent le nombre d'opérations nécessaires au calcul de cette transformée inverse $IGSFFT$ est le

même que celui demandé par la FFT inverse.

Pour illustrer la procédure de convolution circulaire par la $GSFFT$, nous donnons un exemple de convolution de deux séquences $\{x_k\}$ et $\{y_k\}$ de longueur $M = 256$. A l'instant k, chacune de ces séquences diffère de la précédente de $N = 32$ échantillons.

Une comparaison, en termes de nombre d'opérations de base (additions réelles, soustractions réelles) et de nombre de multiplications réelles a été réalisée entre deux implantations, l'une en $GSFFT$ et l'autre en FFT.

La figure (4.11) représente les résultats de cette comparaison réalisée en traitant dix blocs de chacune des séquences x_k et y_k.

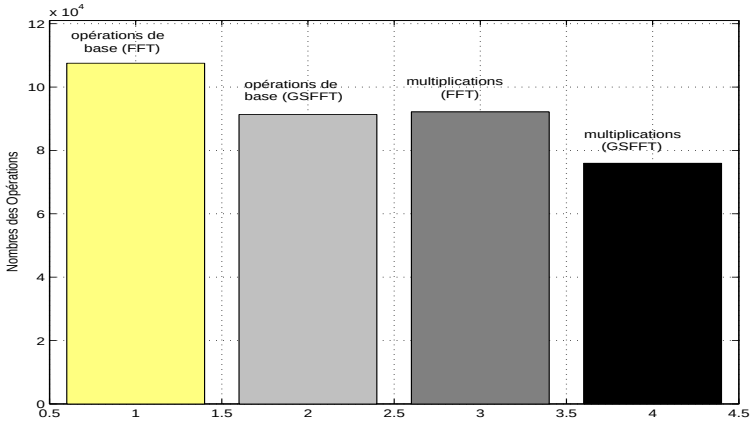

FIG. 4.11 – Opérations requises pour 10 blocs des convolutions pour M=256 et N=32

Ce résultat montre tout l'intérêt que représente l'utilisation de la $GSFFT$ en terme de réduction du nombre d'opérations (multiplications, additions, soustractions, etc...). Nous notons que, par rapport à la FFT, la $GSFFT$ nécessite, pour sa réalisation, moins d'opérations de multiplications et d'additions.

Ce résultat intéressant obtenu avec l'utilisation de la $GSFFT$, associé aux résultats précédents obtenus au moyen de la NTT, nous a permis de penser que l'application de la technique du Sliding Généralisé à la NTT apportera une amélioration à la réduction de complexité des calculs.

4.4.1 Technique du Sliding Généralisé appliquée à la transformée en nombres entiers (GSNTT)

Cette nouvelle technique de transformation ($GSNTT$) est basée sur le même principe que celui de la $GSFFT$. Cependant son domaine de définition n'est pas l'ensemble des complexes $\mathbb{C}$ mais un corps de Galois $GF(q)$ d'ordre q. Le calcul des transformées $GSNTT$ utilise essentiellement des opérations dans le corps de $GF(q)$ d'ordre q premier ou pseudo premier. La racine primitive d'une $GSFFT$ d'ordre M dans $\mathbb{C}$, définie par le terme $e^{-j\frac{2\pi}{M}}$, est alors remplacée par la racine $M^{ème}$ de l'unité dans le corps de Galois $GF(q)$ représentée

par le terme générateur α tel que :

$$\left\langle \alpha^M = 1 \right\rangle_q \tag{4.28}$$

Comme pour la $GSFFT$, la $GSNTT$ consiste à définir un vecteur $U_k = \begin{bmatrix} U_{0,k}^T & U_{1,k}^T & \dots & U_{(n+m),k}^T \end{bmatrix}^T$ de q' éléments calculés par une relation de récurrence définie par :

$$\begin{cases} U_{0,k} = Y_k \\ U_{i+1,k} = \left\langle \left(H_{2^{i+1}} \otimes I_{2^n}\right) . \left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes I_{2^{n+i}} \right) \begin{bmatrix} U_{i,k} \\ V_{i,n} U_{i,k-2^{m-i-1}} \end{bmatrix} \right\rangle_q \text{ pour } 0 \leq i \leq m-1 \end{cases} \tag{4.29}$$

où H_i est une matrice définie par l'équation (4.8). Les indices n et m sont définis dans le paragraphe 4.3.2. $V_{i,n}$ est une matrice carrée d'ordre 2^{n+i} définie par :

$$\begin{aligned} V_{i,n} &= V_i \otimes I_{2^n} \\ &= diag \begin{pmatrix} 1, & \alpha^{\sigma(1)}, & \dots, & \alpha^{\sigma\left(2^i-1\right)} \end{pmatrix} \otimes I_{2^n} \end{aligned} \tag{4.30}$$

où $\sigma(r)$ est le bit reverse de r, où $r = 0, 1, \dots, 2^i - 1$.

Pour $m \leq i < n + m$, le vecteur $U_{i,k}$ est donné par :

$$U_{i,k} = \begin{bmatrix} U_{i,k}(0) & U_{i,k}(1) & \dots & U_{i,k}(M-1) \end{bmatrix}^T$$

Les différents éléments sont calculés par la relation :

$$\begin{bmatrix} U_{i+1,k}(c) \\ U_{i+1,k}\left(c + 2^{n+m-i-1}\right) \end{bmatrix} = \left\langle \begin{bmatrix} 1 & V_{i,c} \\ 1 & -V_{i,c} \end{bmatrix} \begin{bmatrix} U_{i,k}(c) \\ U_{i,k}\left(c + 2^{n+m-i-1}\right) \end{bmatrix} \right\rangle_q \tag{4.31}$$

où le calcul de l'indice c est donné par le paragraphe 4.3.2. $V_{i,c} = \alpha^{\sigma_i(c)}$, avec $\sigma_i(c)$ est le bit reverse de $\langle (c-1) \rangle_{M/2^i}$.

De la même façon que la $GSFFT$, la $GSNTT$ d'une séquence X_k composée de M éléments définie dans le corps de Galois $GF(q)$ d'ordre q est donnée par le vecteur $U_{i+1,k}$ pour $i = n + m - 1$, c'est-à-dire $U_{n+m,k}$ dont les coefficients sont ordonnés en "retenue inverse". Ce vecteur $U_{n+m,k}$ représente aussi la NTT de la séquence X_k :

$$GSNTT(X_k) = NTT(X_k) = U_{n+m,k} \tag{4.32}$$

Nous pouvons conclure à partir de l'équation (4.32) que les deux transformées, NTT et $GSNTT$ donnent le même résultat. La différence entre les deux se situe au niveau de la complexité du calcul qui est plus réduite dans le cas de la $GSNTT$.

4.4.2 Cas particulier de l'application de la technique du Sliding Généralisé à la Transformée en Nombre de Fermat (GSFNT)

La $GSFNT$ est un cas particulier de la $GSNTT$ définie dans un corps de Galois d'ordre q égal à un nombre de Fermat $F_t = 2^b + 1$, où b est une puissance de 2 [Agarwal1974].

Le choix du terme générateur α est essentiel pour réduire la complexité de calcul d'une $GSFNT$. Pour la suite, nous choisissons $\alpha = \sqrt{2} = \left\langle 2^{2^{(t-2)}}.\left(2^{2^{(t-1)}} - 1\right)\right\rangle_{F_t}$ avec $t \in \mathbb{N}$, (voir tableau 2.1, chapitre 3). Cette décomposition en puissances de 2 montre que les multiplications peuvent être réalisées par des décalages de bits. La longueur M de la transformée est donnée par $M = 2^{t+2}$.

La figure (4.12), montre un exemple de Butterfly d'une structure $GSFNT$ pour une séquence de longueur $M = 16$ et pour $N = 2$. Dans ces conditions, le nombre de Butterflies nécessaires au calcul de la $GSFNT$ est égal à $\frac{M}{2}(log_2 N + 2) - N$ =22, alors que le nombre de Butterflies nécessaires au calcul de la FNT est égal à $\frac{M}{2}(log_2 M) = 32$. Par cette nouvelle technique proposée, nous pourrons réaliser une structure $GSFNT$ avec moins de Butterflies que dans une structure FNT, et par conséquent de nombre d'opérations (additions/soustractions, décalages des bits, etc...).

Pour illustrer ce résultat intéressant de la $GSFNT$, nous présentons un exemple pour une séquence $X_1 = \begin{bmatrix} x_1 & x_0 & x_{-1} & x_{-2} \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 0 & 0 \end{bmatrix}^T$ de longueur $M = 4$ et dans le cas où $N = 2$. En se référant au tableau (2.1), nous avons pour $t = 1$ un nombre de Fermat $F_1 = 5$ et un terme générateur $\alpha = 2$ qui vérifie l'égalité (4.28) c'est à dire $\left\langle 2^4 = 1\right\rangle_5$. La $GSFNT$ de la séquence X_1 est calculée par les équations (4.29) et (4.31). De ces équations nous avons :

$$U_{0,1} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$$

$$U_{1,1} = \left[\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right) \left(\begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} \right) \right] \ mod\ 5$$

$$= \left[\left(\left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right] \right) \left(\left[\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{array} \right] \right) \left(\left[\begin{array}{c} 1 \\ 2 \\ 0 \\ 0 \end{array} \right] \right) \right] \ mod\ 5 = \left[\begin{array}{c} 1 \\ 2 \\ 1 \\ 2 \end{array} \right]$$

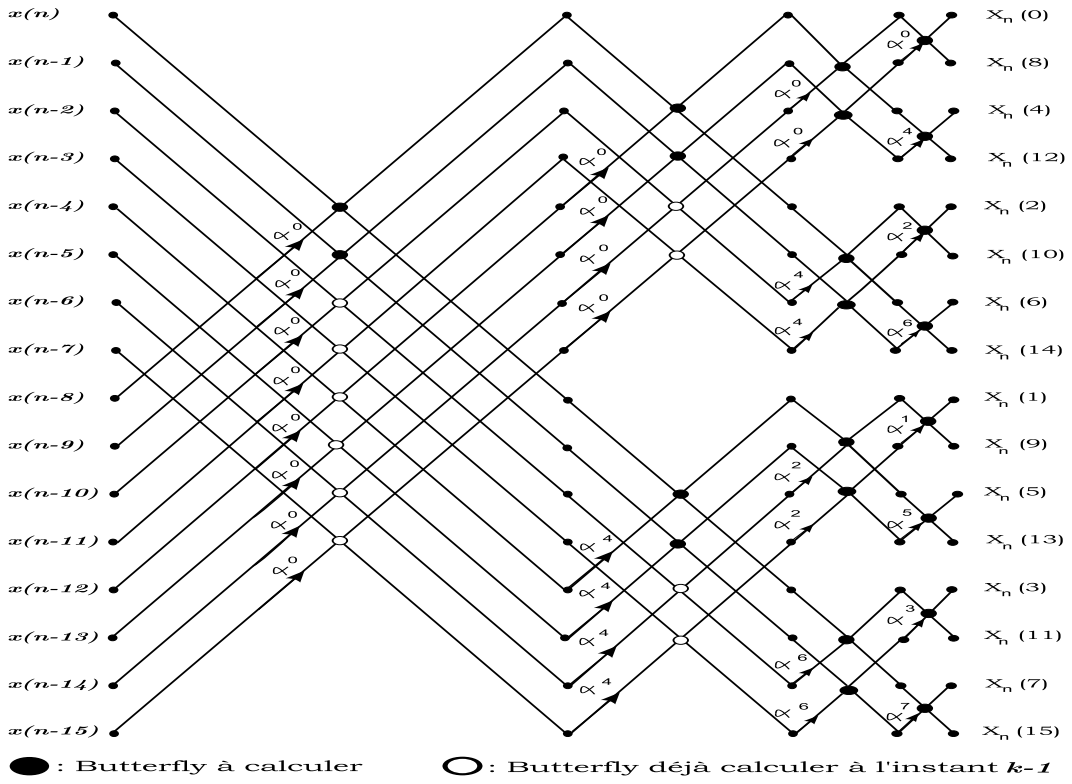

FIG. 4.12 – Principe de la GSFNT pour M=16, N=2

Pour $i = 1$, le terme c prend les valeurs 0 et 2 et l'équation (4.31) s'écrit :

$$\left[\begin{array}{c} U_{2,1}(0) \\ U_{2,1}(1) \end{array} \right] = \left[\begin{array}{cc} 1 & 1 \\ 1 & -1 \end{array} \right] \left[\begin{array}{c} 1 \\ 2 \end{array} \right] \ mod\ 5 = \left[\begin{array}{c} 3 \\ -1 \end{array} \right] \ mod\ 5 = \left[\begin{array}{c} 3 \\ 4 \end{array} \right], \ \text{pour } c = 0$$

$$\left[\begin{array}{c} U_{2,1}(2) \\ U_{2,1}(3) \end{array} \right] = \left[\begin{array}{cc} 1 & 1 \\ 1 & -1 \end{array} \right] \left[\begin{array}{c} 1 \\ 4 \end{array} \right] \ mod\ 5 = \left[\begin{array}{c} 3 \\ -1 \end{array} \right] \ mod\ 5 = \left[\begin{array}{c} 0 \\ 2 \end{array} \right], \ \text{pour } c = 2$$

Le vecteur $U_{2,1}$ est alors donné par :

$$U_{2,1} = \left[\begin{array}{cccc} U_{2,1}(0) & U_{2,1}(2) & U_{2,1}(1) & U_{2,1}(3) \end{array} \right] = \left[\begin{array}{cccc} 3 & 0 & 4 & 2 \end{array} \right] = FNT(X_1) = GSFNT(X_1)$$

Cet exemple illustre bien que les approximations telles que des arrondis des nombres n'existent pas dans une arithmétique en nombres entiers. Il est alors intéressant de traiter les produits de convolution présents dans le système d'annulation d'écho acoustique par la $GSFNT$ afin de réduire au maximum la complexité de calcul déjà réduite par la FNT.

4.4.3 Principe de calcul de la convolution circulaire par la GSFNT

Nous allons présenter dans cette partie le principe de calcul de la convolution circulaire par la mise en oeuvre de la $GSFNT$. Le calcul de convolution circulaire entre deux séquences $\{x\}$ et $\{y\}$, de longueur M chacune, exige celui de trois transformées dont deux transformées directes ($GSFNT$) et une transformée inverse ($IGSFNT$) comme le montre la figure (4.13).

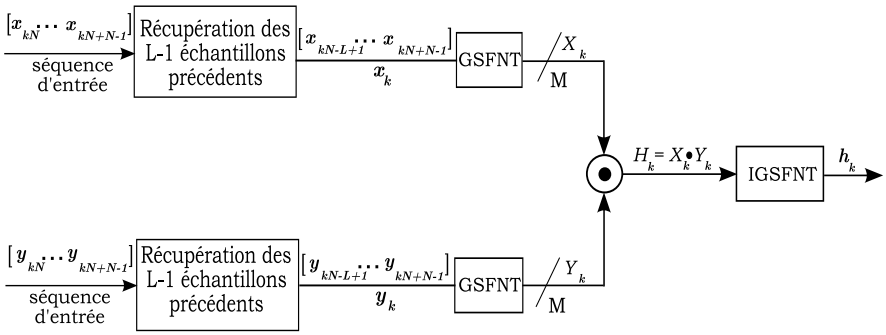

FIG. 4.13 – Description d'un calcul de convolution par la GSFNT

La procédure illustrée par la figure (4.13) permet d'obtenir la séquence $h = x * y$, résultant du produit de convolution entre $\{x\}$ et $\{y\}$.

Un exemple du produit de convolution est illustré dans cette partie pour montrer l'intérêt que présente la $GSFNT$ par rapport à la FNT au niveau de la complexité de calcul.

Cette procédure de convolution, mise en oeuvre en utilisant des séquences de longueur $M = 256$ avec $N = 32$, est réalisée en virgule fixe sur $b = 16$ bits avec un terme générateur $\alpha = \sqrt[8]{2} = 5574$.

Les calculs de la $GSFNT$ seront alors réduits par un modulo égal au nombre de Fermat $F_4 = 2^b + 1 = 65537$.

L'ordre de grandeur au niveau de la complexité de calcul donné par la FNT et par la $GSFNT$ est présenté par la figure (4.14).

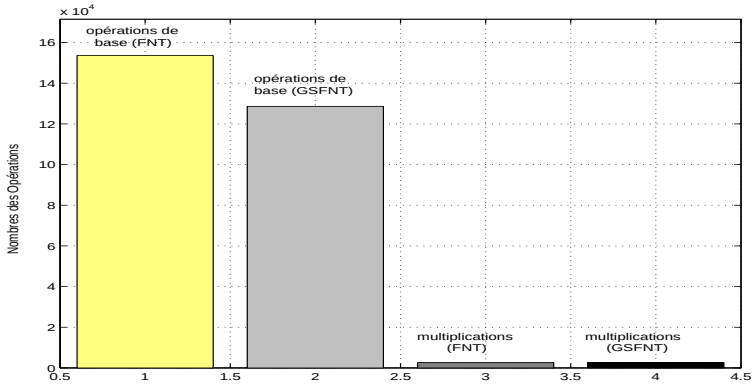

FIG. 4.14 – Opérations requises pour 10 blocs des convolutions pour M=256, N=32

D'après cette figure, le calcul d'une convolution au moyen de la transformation $GSFNT$ nécessite le même nombre de multiplications que si ce même calcul est effectué au moyen de la FNT. Cependant, ce calcul par la $GSFNT$ présente l'avantage d'exiger moins d'additions réelles et de soustractions réelles que s'il est effectué par la FNT. Ainsi une réduction de plus de 20 % du nombre d'opérations (additions, soustractions, décalages de bits) est obtenue en mettant en oeuvre la transformation $GSFNT$ à la place de la FNT.

Les algorithmes du filtrage adaptatif utilisés dans le traitement d'annulation d'écho acoustique nécessitent des calculs de convolutions. Il est alors intéressant de traiter ces algorithmes par la transformation $GSFNT$ pour réduire la complexité de calcul par rapport à la FNT et envisager ainsi une implémentation moins coûteuse d'un système d'annulation d'écho acoustique sur un processeur de traitement du signal.

4.5 Traitement du problème d'annulation d'écho par la technique de la GSFNT

Nous avons proposé la technique du Sliding Généralisé appliquée à la FNT ($GSFNT$) pour la mettre en application dans le domaine du traitement du signal et en particulier dans le domaine de l'annulation d'écho acoustique qui nous intéresse. Cette nouvelle transformation ($GSFNT$) présente, nous l'avons vu, l'avantage de nécessiter moins d'opérations (additions, soustractions, décalages de bits) par rapport à la FNT.

A ce titre, nous proposons une implantation efficace de l'algorithme $BPNLMS++$ par la $GSFNT$ afin de réduire au maximum le nombre d'opérations nécessaires à cette implantation.

4.5.1 Procédure d'implantation de l'algorithme BPNLMS++ par la GSFNT

L'implantation du système d'annulation d'écho acoustique par la mise en oeuvre de la $GSFNT$ sur un processeur de $b = 16$ bits est basée sur les propriétés de la convolution circulaire. Toutes les opérations arithmétiques seront effectuées modulo le nombre de Fermat $F_4 = 2^b + 1 = 65537$. Les différentes séquences à

convoluer dans l'algorithme $BPNLMS++$ seront choisies de longueur $M = N + L - 1 = 256$, avec $N = 64$, quantifiées sur $b = 16$ bits. Le terme générateur $\alpha = \sqrt[8]{2} = 5574$.

Les convolutions rencontrées dans l'algorithme $BPNLMS++$ sont alors effectuées à l'aide de la $GSFNT$ et le calcul d'une convolution exige celui de trois transformées $GSFNT$ de longueur $M = 256$ chacune.

Avant de décrire la procédure d'implantation de l'algorithme $BPNLMS++$ par la $GSFNT$, rappelons que la mise à jour des coefficients $\hat{w}_k$ du filtrage adapté par cet algorithme est donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{G_k\left(\tilde{x}_k * \tilde{x}_{-k}\right)} G_k\left(\epsilon_k * \tilde{x}_{-k}\right)$$

avec $\tilde{x}_k = \begin{bmatrix} x_{kN-L+1} & x_{kN-L+2} & \dots & x_{kN+N-1} \end{bmatrix}^T$, $\tilde{x}_{-k} = \begin{bmatrix} x_{kN+N-1} & x_{kN+N-2} & \dots & x_{kN-L+1} \end{bmatrix}^T$ et $\hat{w}_k = \begin{bmatrix} \hat{w}_k(0) & \hat{w}_k(1) & \dots & \hat{w}_k(L-1) \end{bmatrix}^T$. ϵ_k désigne le signal d'erreur donné par :

$$\epsilon_k = \tilde{d}_k - \tilde{y}_k = \begin{bmatrix} e_{kN} & e_{kN+1} & \dots & e_{kN+N-1} \end{bmatrix}^T$$

Pour les étapes paires, G_k est une matrice identité I de taille $(L \times L)$, et pour les étapes impaires $G_k = diag\begin{bmatrix} g_k(0), & \dots, & g_k(L-1) \end{bmatrix}$ est une matrice diagonale de taille $(L \times L)$ avec $g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\sum_{m=0}^{L-1}\gamma_k(m)}$, où $\gamma_k(n) = max\left\{\rho\nu_k, |\hat{w}_k(n)|\right\}$, $n \in \{0, \dots, L-1\}$ et $\nu_k = max\left\{\delta, |\hat{w}_k(0)|, \dots, |\hat{w}_k(L-1)|\right\}$. Les termes ρ et δ sont respectivement choisis égaux à $\frac{5}{L}$ et à 10^{-2}.

Les différentes étapes de la procédure d'implantation de l'algorithme $BPNLMS++$ par la $GSFNT$, au moyen de la convolution circulaire, sont décrites ci-dessous et résumées par la figure (4.15) :

1. Fenêtrage du signal d'entrée $\{x\}$ (fenêtre de $N = 64$ échantillons).

2. Construction de $\tilde{X}_k$ par récupération des $L - 1 = 192$ échantillons du vecteur précédent, $\tilde{X}_{k-1}$.

3. Calcul de $\tilde{\mathbf{X}}_k$, $GSFNT$ de la séquence $\tilde{X}_k$.

4. Construction du vecteur $\tilde{W}_k$ par ajout de $M - L = 63$ échantillons nuls à la fin du vecteur $\hat{w}_k$.

5. Calcul de $\tilde{\mathbf{W}}_k$, $GSFNT$ de la séquence $\tilde{W}_k$.

6. Calcul du produit terme à terme $\tilde{\mathbf{W}}_k \bullet \tilde{\mathbf{X}}_k$ des vecteurs $\tilde{\mathbf{W}}_k$ et $\tilde{\mathbf{X}}_k$.

7. Passage dans le domaine temporel par le calcul de la transformée inverse de la FNT $\left(FNT^{-1}\right)$ ou par le calcul de la transformée inverse de la $GSFNT$ ($IGSFNT$) du produit terme à terme $\tilde{\mathbf{W}}_k \bullet \tilde{\mathbf{X}}_k$.

8. Récupération des N derniers échantillons $(N = 64)$.

9. Calcul du vecteur d'erreur ϵ_k.

10. Construction du vecteur $\tilde{E}_k$ par ajout de $M - N = 192$ échantillons nuls à la fin du vecteur ϵ_k.

11. Calcul du vecteur $\tilde{\mathbf{E}}_k$ par application de la FNT ou de la $GSFNT$ au vecteur d'erreur $\tilde{E}_k$.

12. Calcul de $\tilde{X}_{-k}$ par application de la matrice $\overline{I}_{M\times M}$.

13. Calcul du vecteur $\tilde{\mathbf{X}}_{-k}$, transformée $GSFNT$ du vecteur $\tilde{X}_{-k}$.

14. Calcul du produit terme à terme $\tilde{\mathbf{X}}_{-k} \bullet \tilde{\mathbf{E}}_k$.

15. Passage dans le domaine temporel par le calcul de la transformée inverse FNT^{-1} ou $IGSFNT$ du vecteur $\tilde{\mathbf{X}}_{-k} \bullet \tilde{\mathbf{E}}_k$.

16. Récupération des $L = 193$ derniers échantillons du vecteur obtenu à l'issue de l'étape précédente.

17. Calcul du produit terme à terme $\left(\tilde{\mathbf{X}}_k \bullet \tilde{\mathbf{X}}_{-k}\right)$.

18. Passage dans le domaine temporel par le calcul de la transformée inverse FNT^{-1} ou $IGSFNT$ du vecteur $\tilde{\mathbf{X}}_k \bullet \tilde{\mathbf{X}}_{-k}$.

19. Récupération des $L = 193$ derniers échantillons du vecteur obtenu à l'issue de l'étape précédente.

20. Mise à jour du calcul des coefficients $\hat{w}_k$ du filtre de longueur $L = 193$ par l'algorithme $BPNLMS{+}{+}$.

21. Répétition des étapes 1 à 20 de la procédure pour l'indice $k + 1$.

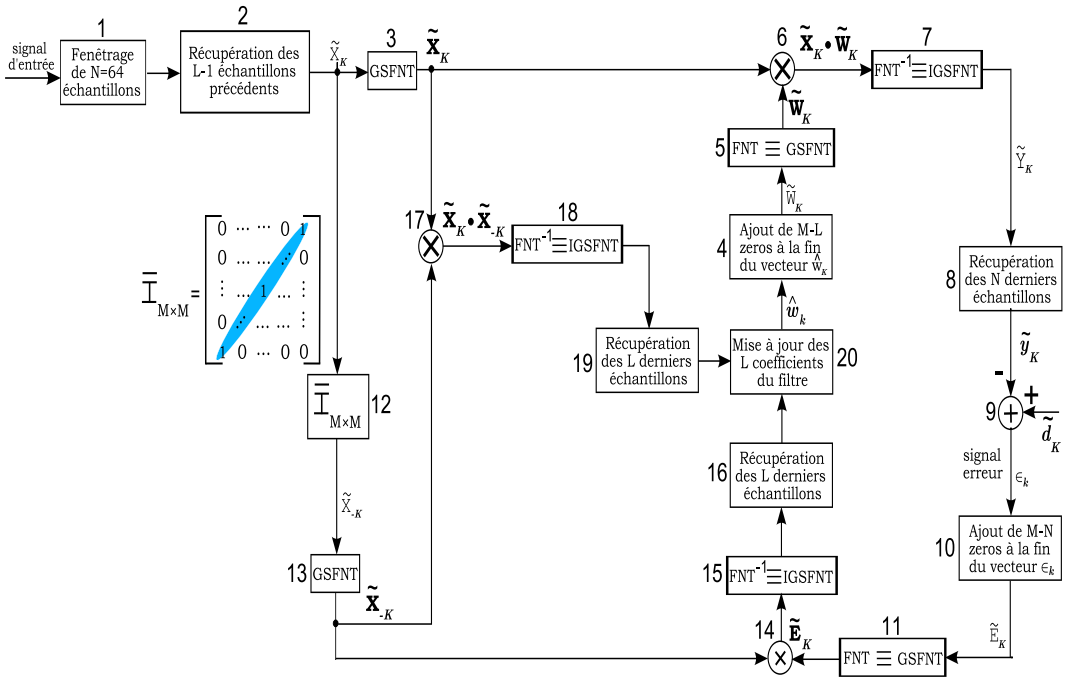

FIG. 4.15 – Procédure du filtrage adaptatif par la GSFNT

Dans cette procédure, lorsqu'on traite un vecteur ne présentant pas de chevauchement avec le vecteur précédent, nous pouvons utiliser aussi bien la $GSFNT$ que la FNT dans la mesure où celles-ci nécessitent des calculs de même complexité.

4.5.2 Résultats des simulations

Plusieurs simulations numériques ont été réalisées pour évaluer les performances de l'algorithme $BPNLMS{+}{+}$ traité par la $GSFNT$ dans le cadre de l'annulation d'écho acoustique.

Les simulations sont réalisées au moyen d'un signal d'entrée (signal de parole lointaine) et une réponse impulsionnelle de chemin d'écho représentés respectivement par les figures (4.16-a) et (4.16-b). Les valeurs des

différents paramètres utilisés par l'algorithme $BPNLMS++$ sont données par : $\beta = 10^2$, $\rho = \frac{1}{L}$, $\delta = 10^{-2}$, $\mu_B = 0.8$, $N = 64$.

Les différentes performances sont évaluées, en calculant des mesures objectives comme, l'évolution des coefficients du filtre adaptatif, la puissance de l'écho résiduel, l'atténuation $ERLEC$, fournie par l'annulateur d'écho et la convergence N_m des coefficients du filtre adaptatif.

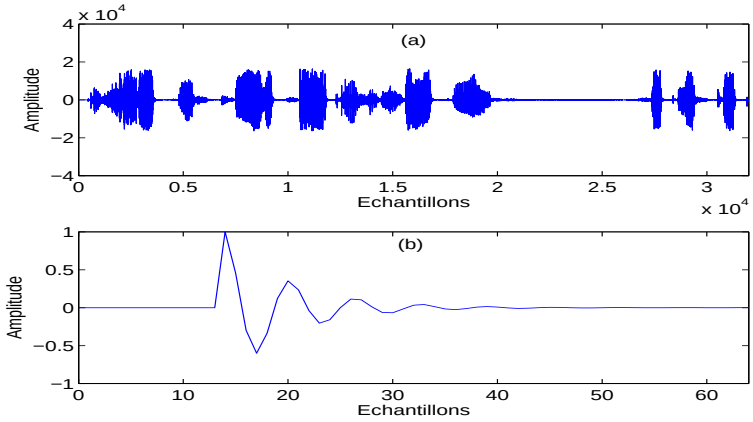

FIG. 4.16 – (a) : Signal d'entrée, (b) : Réponse impulsionnelle du chemin d'écho

Un des critères des performances est déterminé par l'atténuation de l'écho acoustique mesurée par la convergence $N_m = 10log_{10}\left(\frac{\|w-\hat{w}\|^2}{\|w\|^2}\right)$ des coefficients du filtre adaptatif donnée par la figure (4.17-a).

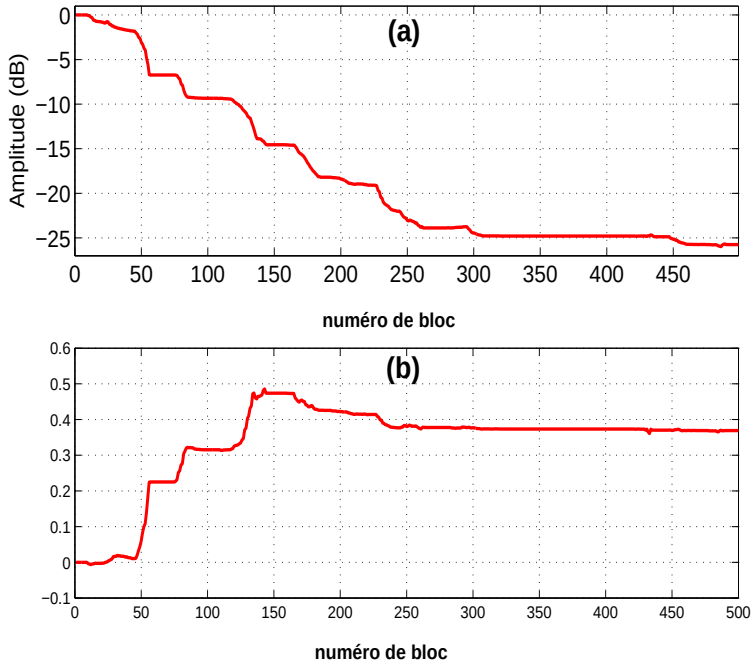

FIG. 4.17 – (a) : Convergence des coefficients du filtre adaptatif, (b) : Evolution d'un coefficient, $(\hat{w}_k\,(21))$, du filtre

L'autre critère est mesuré par l'évolution du coefficient $\hat{w}_k\,(21)$ du filtre, représentée par la figure (4.17-b).

Ces deux mesures montrent que l'algorithme $BPNLMS++$ implanté par la $GSFNT$ présente une convergence comparable à celle obtenue au moyen de l'implantation en FNT.

L'atténuation d'écho fournie par l'annulateur d'écho peut aussi être mesurée par le rapport :

$$ERLEC\,(k) = 10log_{10}\left(\frac{\sum_{n=(k-1)N+1}^{kN}(d_n)^2}{\sum_{n=(k-1)N+1}^{kN}(d_n-\hat{y}_n)^2}\right) \tag{4.33}$$

Cette atténuation, calculée au moyen de la FNT et de la $GSFNT$, est représentée par la figure (4.18-a) et (4.18-b).

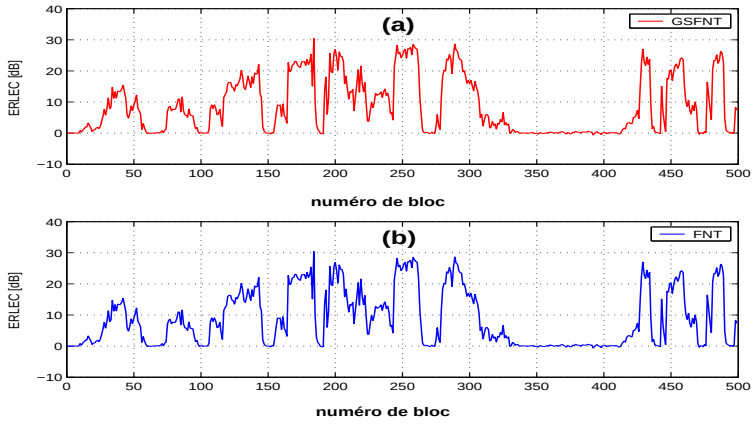

FIG. 4.18 – Atténuation d'écho, $ERLEC$, fournie par l'algorithme BPNLMS++ en utilisant la FNT et la GSFNT

Ces deux figures montrent que les deux méthodes d'implantation de l'algorithme $BPNLMS++$ fournissent le même résultat en terme d'atténuation d'écho.

Les performances de l'algorithme $BPNLMS++$ peuvent aussi être déterminées par la mesure de l'atténuation de la puissance de l'écho résiduel représentée par la figure (4.19).

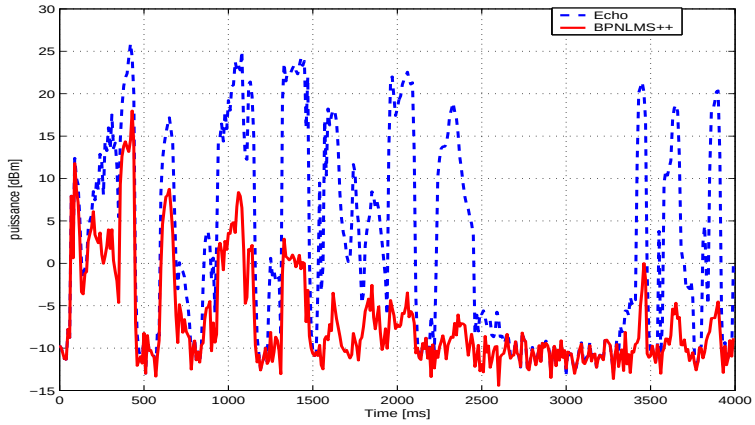

FIG. 4.19 – Puissance de l'écho et de l'écho résiduel

Cette illustration montre que l'algorithme $BPNLMS++$ a rendu la puissance de l'écho acoustique

inaudible en sortie du système d'annulation d'écho.

La différence entre les deux méthodes d'implantation réside plutôt au niveau de la complexité de calcul. Pour illustrer cette différence, nous avons représenté sur la figure (4.20) le nombre d'opérations requises pour l'implantation en *FNT* et en *GSFNT* de l'algorithme *BPNLMS++*.

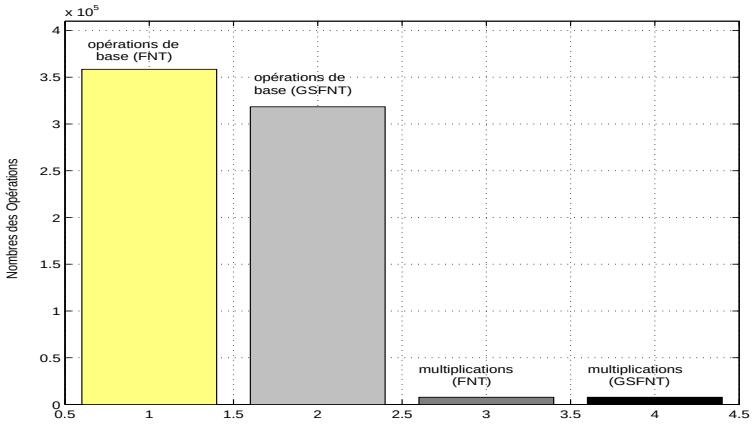

FIG. 4.20 – Opérations requises par l'algorithme BPNLMS++ pour 10 blocs

Cette illustration montre que l'implantation de l'algorithme *BPNLMS++* par la *GSFNT* nécessite moins d'opérations classiques (additions, soustractions, décalages de bits) que l'implantation de ce même algorithme en *FNT*. Cependant, ce calcul par la *GSFNT* nécessite le même nombre de multiplications que s'il est effectué au moyen de la *FNT*.

4.6 Conclusion

Ce chapitre nous a permis, dans un premier temps, de rappeler la technique du Sliding Généralisé appliquée à la FFT ($GSFFT$) . Cette technique, qui consiste à calculer la transformée de Fourier rapide d'une succession de séquences dans laquelle deux séquences successives diffèrent de N échantillons, a pour objectif de réduire la complexité de calcul d'une FFT.

Nous avons ensuite proposé une technique analogue à celle de la *GSFFT* que nous avons étudiée dans le corps de Galois en faisant appel à la *FNT*. Cette technique intitulée Generalized Sliding *FNT* ($GSFNT$) présente l'avantage de nécessiter moins d'opérations (additions, soustractions, décalages de bits) par rapport à la *FNT*.

Cet avantage nous a conduit à proposer une implantation de faible complexité, au moyen de la $GSFNT$, de l'algorithme *BPNLMS++*.

Il serait donc intéressant d'implémenter cet algorithme en utilisant la *GSFNT* sur une carte *DSP*. Cela permettrait d'observer les temps de calcul et les nombres d'opérations de manière plus représentative.

Chapitre 5

Traitement du problème d'annulation d'écho par des algorithmes robustes associés au système combiné

5.1 Introduction

Nous avons vu dans les deux chapitres précédents que les différents algorithmes adaptatifs peuvent fournir de bons résultats au niveau de l'annulation d'écho acoustique lorsqu'une seule voix est émise. Cependant si deux personnes émettent simultanément un signal, l'algorithme d'adaptation ne va plus tendre vers une modélisation du chemin d'écho mais il aura tendance à diverger. Nous observons ainsi une dégradation du signal reçu, ce qui met en cause les performances de l'annulateur d'écho précédemment étudié. La manière la plus simple et la plus couramment utilisée en pratique pour éviter ce problème de divergence consiste à combiner l'annulateur d'écho acoustique à un système de variation de gain pour rendre l'écho résiduel inaudible. En ce sens, l'association d'un annulateur d'écho et de ce système de variation de gain constitue ce que l'on appelle par la suite, un système combiné. Dans notre contexte d'étude, si nous devions donner une définition du système combiné, ce pourrait être "la combinaison d'un annulateur d'écho à un système de variation de gain" qui permet d'éliminer complètement l'écho sans engendrer de dégradation sur le signal de parole locale.

Le but de ce chapitre est d'introduire un nouveau concept appliqué au système combiné, tel que présenté pour la première fois par Rainer Martin [Martin1995], qui de notre point de vue, constitue la première proposition d'un système combiné au sens où nous l'entendons et servira de point de départ pour ce chapitre.

Les traitements mis en oeuvre par ce nouveau système combiné représentent une charge importante

de calcul. Il est alors nécessaire de réduire au maximum les coûts des opérations à traiter pour pouvoir envisager une implantation de ce système combiné dans un microprocesseur de traitement de signal. Pour une complexité de calculs réduite au maximum, nous proposerons d'implanter ce système combiné par la mise en oeuvre de la $GSFNT$.

Des mesures objectives pour évaluer les performances du système proposé seront alors présentées.

5.2 Etude du système combiné "Rainer Martin"

Un annulateur d'écho utilisé seul dans un contexte radio mobile est en pratique généralement insuffisant pour enlever complètement l'écho. Pour remédier à cette insuffisance, certains auteurs proposent l'utilisation d'un post-filtre appelé "écho shaping filter" placé après l'annulateur d'écho classique [Martin1995]. La combinaison d'un annulateur d'écho et d'un post-filtre constitue le système combiné dit "Rainer Martin". Décrit en termes généraux, le post-filtre, placé après l'annulateur d'écho, permet d'enlever la partie audible de l'écho résiduel pour laquelle l'écho résiduel est plus énergétique que le signal de parole locale et sans engendrer de dégradations perceptibles sur ce signal. Par conséquent, le post-filtre doit être vu comme un moyen permettant d'apporter une réduction supplémentaire d'écho afin de rendre cet écho inaudible.

Le système combiné, dont le principe est décrit par la figure (5.1), est constitué de deux filtres adaptatifs RIF transverses, $\hat{w}$ et $\hat{w}_{opt}$, et d'un détecteur de double parole DTD (DTD : Double Talk Detector). Le filtre $\hat{w}$ est un annulateur d'écho classique pouvant être adapté typiquement selon les algorithmes adaptatifs ($NLMS$, $PNLMS$, $PNLMS++$). La particularité de $\hat{w}$ est d'avoir une taille L réduite devant celle de la réponse impulsionnelle de couplage et qui ne lui permet d'identifier que très partiellement la réponse. Néanmoins, un postulat de base pour cette approche est que l'annulateur d'écho délivre une atténuation minimale de l'ordre de 10 à 15 dB [Turbin1997a].

Le filtre adaptatif $\hat{w}_{opt}$ placé juste après l'annulateur de l'écho, filtre le signal d'erreur $\{e\}$, mélange d'un signal de parole locale $\{s\}$, d'un bruit ambiant $\{b\}$ et de l'écho résiduel $\{y\} - \{\hat{y}\}$. La distorsion générée sur le signal de parole locale doit être maintenue à un niveau acceptable du point de vue subjectif, ce qui signifie qu'en moyenne l'énergie de l'écho résiduel $\{y\} - \{\hat{y}\}$ doit être inférieure à celle du signal de parole locale $\{s\}$. Pour cette raison, dans ce système, le post-filtre $\hat{w}_{opt}$ doit être combiné à un annulateur d'écho dont le rôle est de garantir une atténuation acceptable de l'écho.

Une fonction de détection d'une double activité de parole, formée de deux détecteurs d'activité de parole (VAD : Voice Activity Detector), est placée à l'entrée et à la sortie du système combiné. Ce détecteur DTD est donc nécessaire pour stopper l'adaptation des coefficients du filtre d'annulation d'écho dans le cas de double parole [Cho1999, Benesty2000, Gänsler2000].

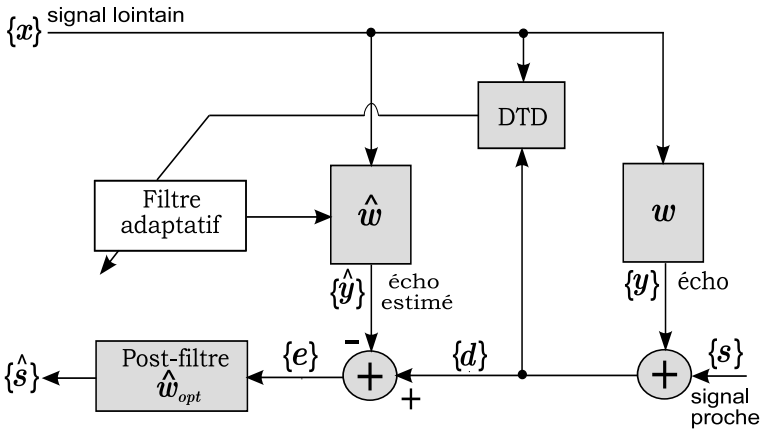

FIG. 5.1 – Principe de réalisation du système combiné

5.3 Principe du système combiné proposé

Il est possible que le détecteur de double parole donne une erreur de détection. Dans ce cas, le système d'annulation d'écho acoustique va diverger, ce qui entraîne une dégradation sur le signal de parole locale.

Pour éviter ce problème de divergence du filtre adaptatif dans le cas d'une erreur de détection, nous proposons d'adapter les coefficients du filtre adaptatif par un algorithme plus robuste. Cet algorithme, que nous nommerons "Robuste $PNLMS++$" ($RPNLMS++$), permet d'atténuer l'écho acoustique dans le cas de simple parole et d'éviter le problème de divergence dans le cas d'une erreur de détection de la double parole.

Le système combiné proposé est représenté par la figure (5.2) [Alaeddine2006]. L'adaptation du filtre $\hat{w}$ est réalisée au moyen de l'algorithme $RPNLMS++$ que nous proposons de traiter par bloc tout en mettant en oeuvre la technique de la transformation $GSFNT$. Cette proposition permet de faire l'étude d'un système complet d'annulation d'écho acoustique présentant une faible complexité.

Le système combiné est basé sur un détecteur de double parole (DTD). Ce détecteur est employé dans le système proposé afin de traiter les situations de simple ou double parole. Les différents cas à traiter dans le système proposé sont les suivants :

1. Si le signal $\{x\}$ est présent et le signal $\{s\}$ est absent, présence de simple parole, le filtre adaptatif est mis à jour à partir de l'algorithme proposé, $RBPNLMS++$, pour éliminer une partie de l'écho acoustique. Le post-filtre placé après l'annulateur d'écho permet de supprimer la partie audible de l'écho résiduel.

2. Si le signal $\{x\}$ est absent et le signal $\{s\}$ est présent, l'écho n'existe pas et les deux filtres adaptatifs sont arrêtés.

3. Si les deux signaux $\{x\}$ et $\{s\}$ sont présents, l'adaptation du filtre $\hat{w}$ est stoppée et l'écho acoustique va être traité par le post-filtre sans engendrer de dégradation notable sur le signal de parole locale.

Les différents éléments qui constituent le système combiné proposé sont décrits dans les paragraphes

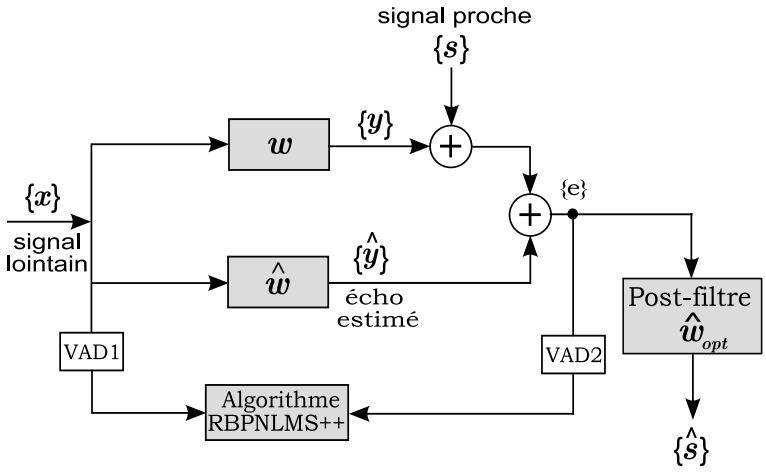

FIG. 5.2 – Schéma de principe du système combiné proposé

suivants.

5.3.1 Algorithme de détection d'activité de la voix (VAD)

Dans cette partie, nous allons décrire l'algorithme de détection d'activité vocale de l'annexe B de la recommandation $G.729$ de l'Union Internationale des Télécommunications [IUTG729]. Dans l'algorithme du VAD, la détection d'activité vocale s'effectue toutes les $10ms$ pour coïncider avec la taille de la trame d'annulateur d'écho acoustique. On extrait un ensemble de paramètres de différence pour rendre une décision initiale. Les paramètres sont l'énergie de la pleine bande de fréquences E_f, l'énergie dans la bande de fréquences basses E_l, le nombre de passages par zéro ZC, et une mesure spectrale. Pendant les segments vocaux non actifs, les moyennes à long terme de ces paramètres, notées $\overline{E}_f$, $\overline{E}_l$ et $\overline{ZC}$, suivent le caractère évolutif du bruit de fond. Un ensemble de paramètres différentiels est obtenu à chaque trame, mesurant la différence entre chaque paramètre et la moyenne à long terme correspondant à ce paramètre. La décision initiale de détection d'activité est obtenue en utilisant une frontière de décision linéaire par morceaux entre chaque couple de paramètres différentiels. Une décision finale de détection d'activité vocale est obtenue en lissant la décision initiale.

La sortie du module VAD est 1 ou 0, indiquant respectivement la présence ou l'absence d'activité vocale.

Un diagramme fonctionnel de l'algorithme VAD est donné par la figure (5.3). L'algorithme VAD fonctionne sur des trames de parole numérisée.

A la première étape, quatre options paramétriques sont extraites du signal d'entrée. Les paramètres sont les énergies dans la pleine bande de fréquences et dans la bande de fréquences basses, l'ensemble des fréquences LSF_i de raies spectrales ("LSF_i", *line spectral frequencies*) et le nombre de passage par zéro.

Si le numéro de trame est inférieur à une constante $N_i = 32$, une étape d'initialisation des moyennes à long terme intervient et la décision de détection d'activité vocale est forcée à 1 si l'énergie de trame obtenue

à partir de l'analyse LPC (LPC, *linear prediction coding*) est supérieure à $15dB$, sinon, la décision de détection d'activité vocale est forcée à 0. La procédure d'initialisation des moyennes concerne les paramètres spectraux du bruit de fond indiqués par $\overline{LSF}_i$, la moyenne des passages par zéro $\overline{ZC}$ et la moyenne de l'énergie de trame E_f sur les premières trames N_i notée par $\overline{E}_n$. Si le numéro de trame est égale à N_i une étape d'initialisation pour les énergies caractéristiques du bruit de fond intervient.

Lors de l'étape suivante, un ensemble de paramètres de différences est calculé. Cet ensemble est produit par une mesure de différence entre les paramètres de la trame courante et les moyennes glissantes des caractéristiques de bruit du fond. Quatre mesures de différences sont calculées :

- Une distorsion spectrale
- Une différence d'énergie
- Une différence d'énergie dans la bande de fréquences basses
- Une différence des nombres de passage par zéro

La prise de décision initiale sur l'activité vocale est prise lors de l'étape suivante, en utilisant des régions de décision à frontières multiples dans l'espace des quatre mesures de différence. La prise de décision sur l'activité vocale s'effectue à partir de la réunion des régions de décision et la décision de non-activité vocale. La prise en compte de l'énergie, de même que les décisions antérieures sur les trames voisines, sont utilisées pour le lissage de décision.

Les moyennes glissantes doivent être mises à jour uniquement en présence de bruit de fond et non en présence de parole. Un seuil adaptatif est essayé, et la mise à jour a lieu uniquement si le critère de seuil est atteint.

Des simulations numériques ont été obtenues pour évaluer le processus de détection d'activité de parole à partir de deux signaux d'entrée, prononcés indifféremment par des locuteurs de sexe féminin et masculin.

Les résultats de ces simulations, réalisées en l'absence de bruit, sont donnés par les figures (5.4) et (5.5). Ils montrent que pour chaque trame du signal d'entrée, le VAD prend de bonnes décisions en fournissant en sortie une valeur 1 si le signal est présent ou une valeur 0 si le signal est absent.

Notons que ce sont ces résultats de décision qui nous serviront par la suite, dans notre système combiné, pour détecter la présence de simple ou double parole.

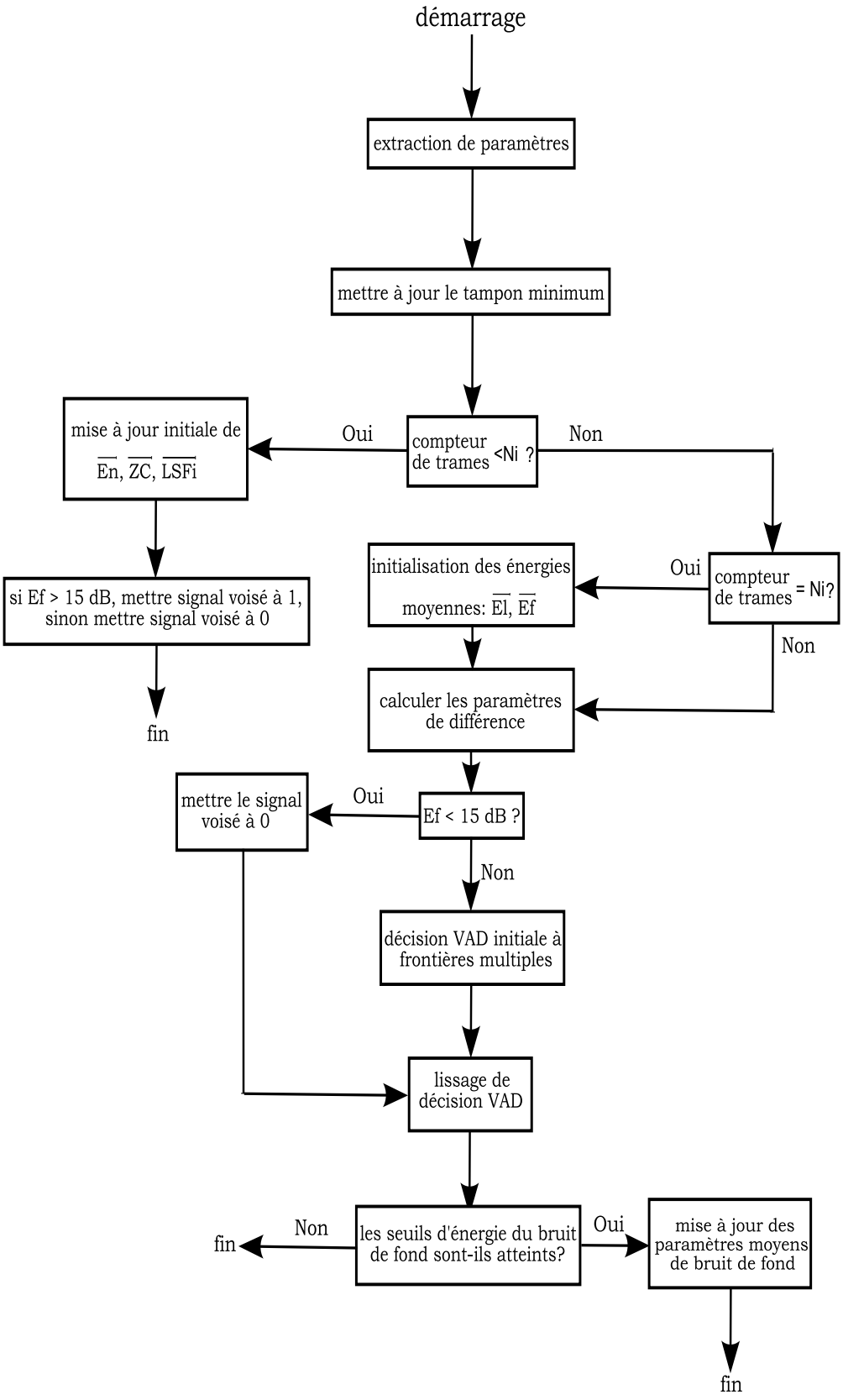

FIG. 5.3 – Organigramme du détecteur d'activité de voix VAD

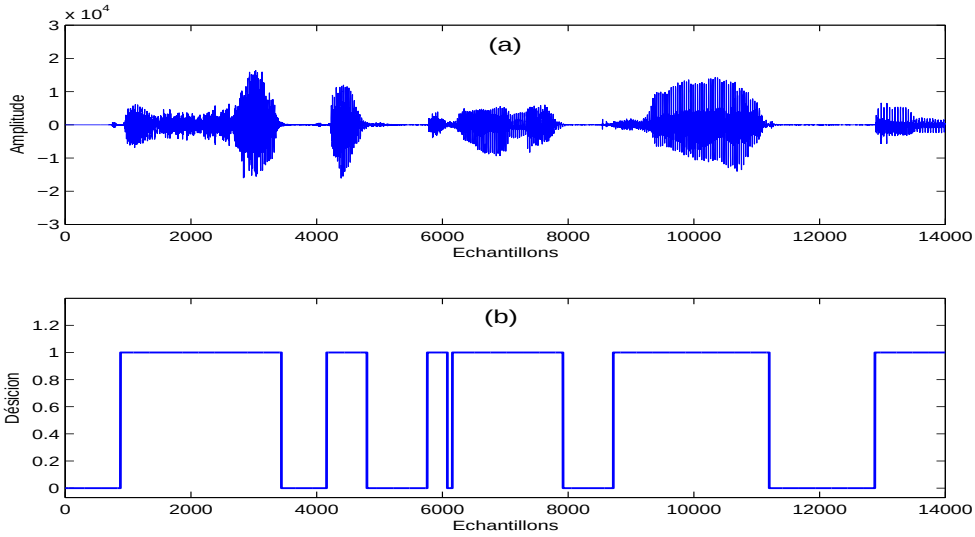

FIG. 5.4 – (a) Voix de femme (b) Trames de Silence/Voix

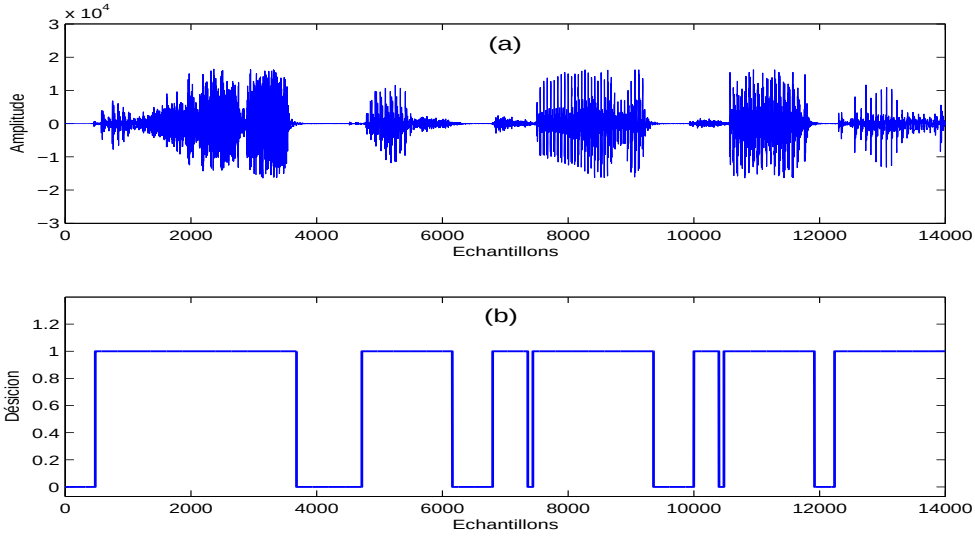

FIG. 5.5 – (a) Voix d'homme (b) Trames de Silence/Voix

5.3.2 Réalisation du post-filtre "echo shaping" : Règle de suppression de bruit d'Ephraim et Malah

5.3.2.1 Introduction

Le problème que nous cherchons à résoudre est la réduction de l'écho résiduel par la mise en oeuvre d'un post-filtre. L'écho résiduel peut être assimilé dans notre cas, et par analogie avec le débruitage, à un bruit dont il est nécessaire de diminuer la contribution dans le signal $\{s\}$ transmis au locuteur distant pour assurer une bonne qualité de communication.

Les techniques classiques de réduction de bruit sont basées sur une règle d'atténuation spectrale à court terme appliquée au signal bruité telle que la soustraction spectrale en puissance [Lim1979], la soustraction spectrale en amplitude [Boll1979], ou la mise en oeuvre directe du filtre de Wiener [Lim1979, Compernolle1992]. Un problème largement discuté dans la plupart des publications est que l'application directe d'une de ces techniques génère un bruit résiduel très peu naturel et souvent très gênant à l'écoute appelé bruit musical [Boll1979, Moorer1986, Vaseghi1992]. Ce phénomène s'explique par le fait que l'atténuation spectrale à court terme appliquée à une composante fréquentielle donnée ne s'exprime qu'en fonction du niveau relatif local mesuré dans la trame à court terme. L'utilisation d'une règle d'estimation du bruit est une technique efficace pour éliminer le bruit musical [Berouti1979, Lockwood1991].

Nous allons nous intéresser plus en détails à une autre méthode rendue très attractive par le fait qu'elle permet de réduire le phénomène très gênant du bruit musical sans introduire de distorsion sur le signal de parole locale [Ephraim1983, Ephraim1984]. Elle se distingue des règles précédentes par le fait qu'elle exploite des connaissances statistiques a priori sur les signaux traités. Comme nous allons le voir, la particularité de cette règle d'atténuation spectrale est qu'elle dépend essentiellement des valeurs du spectre à court terme mesurées dans les trames précédant la trame courante.

5.3.2.2 Description de règle de suppression

La présentation de cette règle de suppression, présentée par [Ephraim1983, Ephraim1984], a donc pour objet la restauration d'un signal utile $\{s\}$ à partir d'un signal bruité $\{d\} = \{s\} + \{b\}$. L'algorithme d'Ephraim et Malah est fondé sur l'estimation de l'amplitude du signal utile de parole suivant le critère de l'erreur quadratique moyenne (EQM), en ajoutant aux hypothèses classiques de stationnarité et de non corrélation du bruit les hypothèses suivantes :

1. Le bruit est supposé gaussien.

2. Sur le signal utile, l'hypothèse faite est que la valeur du spectre à court terme $D(k, f)$ (k est l'indice de la trame courante et f désigne la fréquence d'échantillonnage du signal utile $\{d\}$) est une variable aléatoire gaussienne complexe centrée, dont la partie réelle et la partie imaginaire sont indépendantes.

3. Les coefficients de Fourier de chaque processus (parole, bruit) sont statistiquement indépendants : $D(k_1, f)$ et $D(k_2, f)$ sont statistiquement indépendants dès que $k_1 \neq k_2$ (indépendance entre trames successives).

L'hypothèse (1) est classique et peut se justifier par le théorème centrale limite puisque le bruit peut être considéré comme résultat des contributions d'un grand nombre de sources ponctuelles. L'hypothèse (2) , qui concerne la densité de probabilité a priori de la grandeur à estimer, ne correspond pas forcément au cas des signaux de parole [Porter1984], mais peut être considérée comme une première approximation. En pratique, l'hypothèse (3) n'est que partiellement vérifiée compte tenu du recouvrement entre les trames successives et de la largeur de bande des filtres d'analyse [Cappé1993].

Avec ces hypothèses, l'estimation du spectre d'amplitude à court terme du signal non bruité $\left|\hat{S}(k, f)\right|$ est obtenue sous la forme :

$$\left|\hat{S}(k, f)\right| = \hat{W}_{opt}(k, f) \left|D(k, f)\right| \tag{5.1}$$

où l'atténuation apportée $\hat{W}_{opt}(k, f)$ peut être exprimée sous la forme [Ephraim1984] :

$$\hat{W}_{opt}(k, f) = \frac{\sqrt{\pi}}{2} \left(\sqrt{\left(\frac{1}{1+SNR_{post}(k,f)}\right)\left(\frac{SNR_{prio}(k,f)}{1+SNR_{prio}(k,f)}\right)} \right) \times \Theta\left[(1 + SNR_{post}(k, f))\left(\frac{SNR_{prio}(k,f)}{1+SNR_{prio}(k,f)}\right)\right] \tag{5.2}$$

où la fonction $\Theta(\cdot)$ est définie par :

$$\Theta(u) = exp\left(\frac{-u}{2}\right) \times \left[(1+u)\,\mathbf{I_0}\left(\frac{u}{2}\right) + u\mathbf{I_1}\left(\frac{u}{2}\right)\right] \tag{5.3}$$

où $\mathbf{I_0}(\cdot)$ et $\mathbf{I_1}(\cdot)$ désignent respectivement les fonctions de Bessel de première espèce modifiées d'ordre zéro et du premier ordre.

L'atténuation spectrale à court terme dépend de deux paramètres $SNR_{post}(k, f)$ et $SNR_{prio}(k, f)$, qui sont évalués à chaque trame court terme et pour chaque indice k, et qui sont donnés par :

• SNR_{post} "a posteriori Signal-to-Noise Ratio", désignant le rapport Signal à Bruit a posteriori est défini par :

$$SNR_{post}(k, f) = \frac{\left|D(k, f)\right|^2}{\hat{\gamma}_b(f)} - 1 \tag{5.4}$$

où $\hat{\gamma}_b(f)$ désigne l'estimée de la densité spectrale de puissance du bruit additif $\{b\}$. Cette équation indique que le rapport signal à bruit a posteriori est calculé à partir des données de la trame courante.

• $SNR_{prio}(k, f)$, "a priori Signal-to-Noise Ratio", est exprimé en fonction du module du spectre à court terme du signal de parole utile estimé à partir des trames précédentes et du SNR_{post}, comme suit : [Turbin1997a, Turbin1997b]

$$SNR_{prio}(k,f) = \tilde{\beta}\frac{\left|\hat{W}_{opt}(k-1,f)\,D(k-1,f)\right|^2}{\hat{\gamma}_b(f)} + \left(1-\tilde{\beta}\right) SNR_{post}(k,f) \tag{5.5}$$

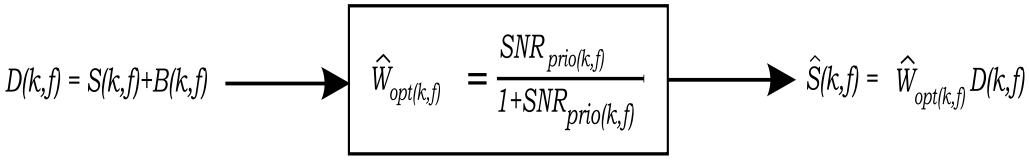

FIG. 5.6 – Schéma bloc du post-filtre

où $\hat{W}_{opt}(k-1,f)\,D(k-1,f)$ est par définition l'estimation du spectre $\hat{S}(k-1,f)$ du signal débruité obtenu au cours de la trame précédente.

Le terme $\frac{\left|\hat{W}_{opt}(k-1,f)D(k-1,f)\right|^2}{\hat{\gamma}_b(f)}$ correspond à une estimation du rapport signal à bruit de la trame précédente. SNR_{prio} est donc une estimée du rapport signal à bruit qui affecte d'un poids $\left(1-\tilde{\beta}\right)$ les données de la trame courante et d'un poids $\tilde{\beta}$ le résultat du traitement des trames précédentes. Les auteurs utilisent une valeur du paramètre $\tilde{\beta}$ d'environ 0.98, ce qui revient à privilégier très fortement la contribution des trames précédentes.

5.3.2.3 Transposition de règle de suppression de bruit au cas de la réduction de l'écho

En s'intéressant de nouveau au principe de réalisation du post-filtre, il est possible de formuler le problème posé sous la forme d'un problème de réduction d'un bruit, où le terme bruit doit être pris au sens large et compris comme un signal perturbateur additif au signal désiré $\{d\}$.

Dans l'approche choisie, le signal présenté en entrée du post-filtre est le signal d'erreur $\{e\}$ obtenu juste après l'annulation de l'écho dans le cas où le signal proche $\{s\}$ est absent.

Dans le cas contraire, le signal présenté en entrée du post-filtre, c'est à dire le signal bruité, est le signal microphone $\{d\}$ constitué d'un mélange du signal utile $\{s\}$ et de l'écho $\{y\}$. On en déduit immédiatement que, dans notre contexte, le signal bruit est le signal d'écho direct.

En supposant que l'écho direct est décorrélé du signal d'erreur $\{e\}$, l'atténuation spectrale à court terme du post-filtre optimal appliquée à la trame k et pour la fréquence f est donc donnée par : [Turbin1997a]

$$\hat{W}_{opt}(k,f) = \frac{SER_{prio}(k,f)}{1+SER_{prio}(k,f)} \tag{5.6}$$

où $SER_{prio}(k,f)$ désigne "a priori Signal-to-Echo Ratio" et s'exprime en fonction du module du spectre à court terme du signal de parole utile estimé à partir des trames précédentes et du SER_{post} comme suit :

$$SER_{prio}(k,f) = \tilde{\beta}\frac{\left|\hat{W}_{opt}(k-1,f)\,D(k-1,f)\right|^2}{\hat{\gamma}_y(f)} + \left(1-\tilde{\beta}\right)SER_{post}(k,f) \tag{5.7}$$

où SER_{post} désigne "a posteriori Signal-to-Echo Ratio" est donné par :

$$SER_{post}(k,f) = \frac{|D(k,f)|^2}{\hat{\gamma}_y(f)} - 1 \tag{5.8}$$

avec $\hat{\gamma}_y(f)$ désigne l'estimée de la densité spectrale de puissance du signal d'écho $\{y\}$.

La transposition de la règle de suppression de bruit d'Ephraim et Malah au cas de la réduction d'écho n'est pas triviale et peut paraître impromptue puisque le bruit traité ici, l'écho, en l'occurrence un signal de parole, ne satisfait pas les hypothèses de départ du bruit gaussien stationnaire. Malgré cela, et les résultats expérimentaux du système proposé le montreront, cette règle d'atténuation spectrale semble adaptée à la situation. On notera aussi que l'atténuation du filtre représentée par l'équation (5.6) est beaucoup plus facile à calculer que l'estimation d'origine représentée par l'équation (5.2).

Par la suite, l'implantation du post-filtre calculé par les équations, (5.6), (5.7) et (5.8), a été optimisée en terme de complexité de calculs à l'aide de Transformées en Nombres de Fermat (FNT).

Dans le paragraphe suivant, nous allons présenter le principe des algorithmes robustes. Nous proposerons ensuite de traiter, par blocs, ces différents algorithmes robustes pour pouvoir envisager une implantation en virgule fixe sur un processeur de traitement de signal par la mise en oeuvre de la $GSFNT$.

5.3.3 Algorithmes adaptatifs robustes

Le traitement du système d'annulation d'écho acoustique par des algorithmes robustes limite le problème de divergence du filtre adaptatif dans le cas où deux personnes parlent simultanément (double parole).

A ce titre, nous allons présenter la méthode de filtrage adaptatif par des algorithmes robustes. Par la suite, nous allons décrire ces différents algorithmes du gradient stochastique par bloc afin d'avoir un système combiné d'annulation d'écho acoustique complet que nous traiterons par la $GSFNT$ afin de réduire au maximum sa complexité de calcul.

5.3.3.1 Algorithmes du gradient stochastique robuste LMS (RLMS)

Nous rappelons que l'algorithme du gradient stochastique LMS consiste à minimiser l'erreur quadratique moyenne entre la sortie désirée $\{d_k\}$ et l'écho estimé $\{\hat{y}_k\}$, à partir de la fonction coût définie par :

$$\xi_k = E\left(e_k^2\right) \tag{5.9}$$

où $e_k = d_k - \hat{y}_k$.

La technique de l'algorithme du gradient stochastique robuste consiste à calculer la fonction coût par : [Sondhi1967]

$$\xi_k = E\left(\zeta\left(\frac{|e_k|}{s}\right)\right) \tag{5.10}$$

où la fonction $\zeta(\cdot)$ est une fonction symétrique quelconque dont la dérivée est une fonction monotone décroissante et s est un facteur d'échelle variable qui diminue l'effet de double parole et donc de divergence du filtre adaptatif. Ce facteur d'échelle est calculé par une relation de récurrence, soit pour l'indice k, par :

$$s_k = \lambda s_{k-1} + \frac{1-\lambda}{\lambda'} s_{k-1}\psi\left(\frac{|e_k|}{s_{k-1}}\right) \tag{5.11}$$

$\psi(\cdot)$ est une fonction limiteur donnée par :

$$\psi\left(\frac{|e_k|}{s_{k-1}}\right) = min\left\{\frac{|e_k|}{s_{k-1}}, \kappa_0\right\} \tag{5.12}$$

où $\kappa_0 = 1.1$. Le détail des calculs nécessaires pour parvenir à la formulation du facteur d'échelle s est donné en l'annexe C, $\lambda \in [0.99,\ 1]$, et $\lambda' \in [0.60,\ 0.74]$. La valeur initiale du facteur d'échelle pour $k = 0$ est choisie par : [Gänsler1998, Gänsler2000]

$$s_0 = \sigma_x \tag{5.13}$$

où σ_x est la puissance du signal de parole.

Les coefficients $\hat{w}_k$ du filtre adaptatif sont mis à jour selon le principe du gradient stochastique :

$$\hat{w}_{k+1} = \hat{w}_k - \mu \bigtriangledown \xi_k \tag{5.14}$$

Cet algorithme peut être rendu robuste si l'on choisit la fonction ζ, telle que $lim_{|e_k|\to\infty}\left|\nabla\zeta\left(\frac{|e_k|}{s_k}\right)\right| < \infty$. D'après cette hypothèse, le gradient $\nabla\xi_k$ peut s'écrire : [Huber1981]

$$\nabla\xi_k = E\left\{-\chi_k sign\,(e_k)\,\psi\left(\frac{|e_k|}{s_k}\right)\frac{1}{s_k}\right\} \tag{5.15}$$

avec

$$sign\,(u) = \left\{\begin{array}{c} 1 \text{ si } u > 0 \\ -1 \text{ si } u < 0 \end{array}\right\} \tag{5.16}$$

$\chi_k = \left[\begin{array}{cccc} x_k & x_{k-1} & x_{k-2} & \ldots & x_{(k-(L-1))} \end{array}\right]^T$ étant la séquence d'entrée.

L'algorithme ainsi obtenu est dit Robuste-LMS ($RLMS$) [Huber1981, Gänsler2000].

La mise à jour du filtre adaptatif de cet algorithme robuste est définie, à chaque indice k d'itération, par l'équation :

$$\hat{w}_{k+1} = \hat{w}_k + \mu\chi_k\psi\left(\frac{|e_k|}{s_k}\right) sign\left\{e_k\right\} s_k \tag{5.17}$$

Pour des signaux non stationnaires, et à partir de l'adaptation des coefficients du filtre adaptatif par l'algorithme $RLMS$, il est difficile de suivre les variations du signal d'entrée, ce qui donne une vitesse de convergence lente. Pour limiter ce problème de convergence, nous apportons une modification à l'algorithme $RLMS$ en normalisant le pas d'adaptation μ par rapport à l'énergie du signal d'entrée. L'algorithme ainsi obtenu, appelé $RNLMS$ est décrit dans le paragraphe suivant.

5.3.3.2 Algorithme Robuste LMS Normalisé (RNLMS)

Cet algorithme consiste à normaliser le pas d'adaptation par rapport à l'énergie du signal d'entrée, ceci permet de réduire au minimum l'effet de la variation de la puissance du signal d'entrée et de rendre plus au moins la convergence uniforme pendant le processus d'adaptation. Ceci permet d'aboutir à l'algorithme du gradient stochastique robuste LMS normalisé ($RNLMS$).

La mise à jour du filtre adaptatif par l'algorithme Robuste-$NLMS$ ($RNLMS$) est donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu}{\chi_k^T\chi_k + \beta}\chi_k\psi\left(\frac{|e_k|}{s_k}\right) sign\left\{e_k\right\} s_k \tag{5.18}$$

où β est un facteur permettant de suivre plus ou moins rapidement les variations d'énergie dans le signal d'entrée $\{x\}$.

5.3.3.3 Algorithme Robuste LMS Normalisé Proportionné (RPNLMS)

Cet algorithme exploite la faible densité des réponses impulsionnelles pour réaliser une adaptation sensiblement plus rapide que l'algorithme $RNLMS$.

L'algorithme $RPNLMS$ résulte directement de l'algorithme $RNLMS$ étudié dans le paragraphe précédent. La mise à jour du filtre adaptatif par l'algorithme $RPNLMS$ est donnée par : [Benesty2001, Gänsler2000]

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu}{\chi_k^T G_k\chi_k + \beta}G_k\chi_k\psi\left(\frac{|e_k|}{s_k}\right) sign\left\{e_k\right\} s_k \tag{5.19}$$

où $G_k = diag\left[\begin{array}{cccc} g_k(0), & \ldots, & g_k(L-1) \end{array}\right]$ est la matrice diagonale $(L \times L)$ avec $g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\Sigma_{m=0}^{L-1}\gamma_k(m)}$, $\gamma_k(n) = max\left\{\rho\nu_k, |\hat{w}_k(n)|\right\}$, $n \in \{0, \ldots, L-1\}$ et $\nu_k = max\left\{\delta, |\hat{w}_k(0)|, \ldots, |\hat{w}_k(L-1)|\right\}$. Les termes ρ et

δ sont respectivement choisis égaux à $\frac{5}{L}$ et à 10^{-2}.

Si la réponse impulsionnelle du filtre adaptatif est dispersive, le taux de convergence peut être réellement plus lent que pour l'algorithme $RNLMS$. Pour limiter ce problème, nous proposons d'utiliser l'algorithme Robuste-$PNLMS++$ ($RPNLMS++$), combinaison des deux algorithmes $RNLMS$ et $RPNLMS$.

5.3.3.4 Algorithme Robuste PNLMS++ (RPNLMS++)

L'algorithme $Robuste - PNLMS++$ est obtenu à partir des deux algorithmes précédents $RNLMS$ et $RPNLMS$ [Gay1998, Gänsler2000].

Pour les étapes impaires (indice k impair), l'adaptation est réalisée de la même manière que dans le cas de l'algorithme $RPNLMS$ et pour les étapes paires (indice k pair), l'adaptation est réalisée de la même manière que dans le cas de l'algorithme $RNLMS$.

L'alternance entre les deux algorithmes $RNLMS$ et $RPNLMS$ permet de rendre l'algorithme $RPNLMS++$ moins sensible aux variations de la réponse impulsionnelle de l'écho.

5.3.3.5 Comparaison des performances des différents algorithmes (RNLMS, RPNLMS, RPNLMS++)

L'intérêt d'utiliser des algorithmes robustes est de limiter la divergence du filtre adaptatif dans le cas de double parole.

Pour montrer l'avantage des algorithmes robustes ($RNLMS$, $RPNLMS$, $RPNLMS++$) par rapport aux algorithmes non robustes, ($NLMS$, $PNLMS$, $PNLMS++$) dans le cas de double parole, des simulations numériques ont été réalisées dans le cadre de l'annulation d'écho acoustique au moyen de la programmation *MatLab*.

Dans les simulations que nous allons présenter, les performances des algorithmes robustes sont comparées à celles des algorithmes non robustes.

Les différents algorithmes ont ainsi été exécutés pour deux signaux de parole, un signal $\{x\}$ pour la parole lointaine et un signal $\{s\}$ pour la parole locale (proche). Les deux signaux $\{x\}$ et $\{s\}$ sont respectivement représentés par les figures (5.7-a) et (5.7-b).

L'objectif étant de comparer les performances des algorithmes non robustes avec celles des algorithmes robustes, nous avons utilisé des mesures objectives comme la convergence du filtre adaptatif mesurée par N_m, définie par la relation (3.38) du chapitre 3 :

$$N_m = 10log_{10}\left(\frac{\|w - \hat{w}\|^2}{\|w\|^2}\right) \tag{5.20}$$

w et $\hat{w}$ désignent respectivement la réponse impulsionnelle et la réponse impulsionnelle estimée du chemin d'écho.

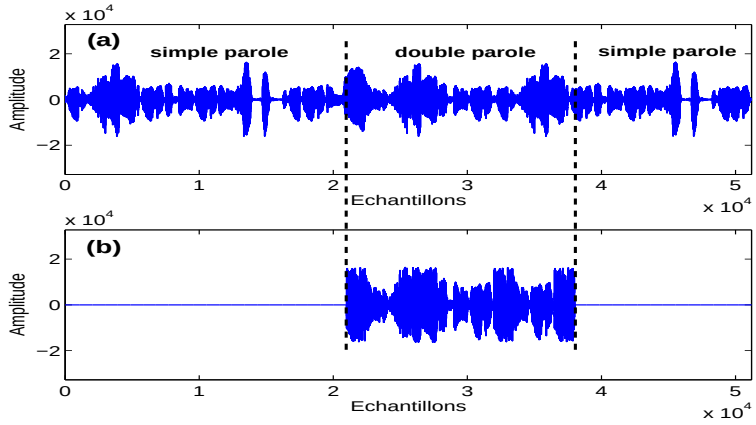

FIG. 5.7 – (a) Signal de parole lointain (b) Signal de parole proche

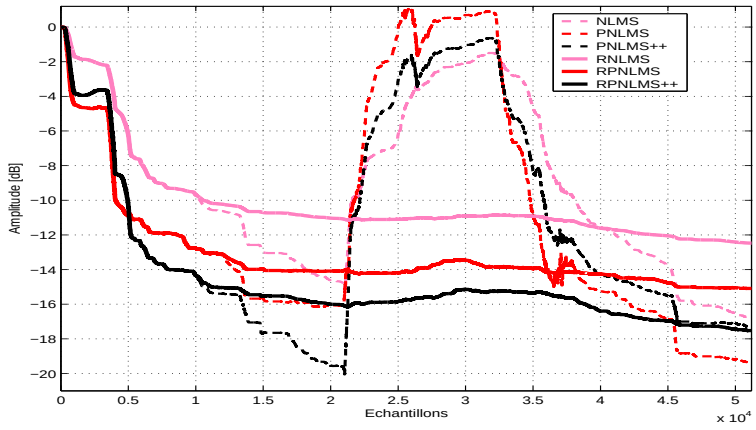

FIG. 5.8 – Convergence des coefficients du filtre par les méthodes d'adaptation robuste et non-robuste

Les résultats des simulations de cette comparaison sont représentés sur la figure (5.8). Ils sont obtenus en choisissant N et L, longueurs respectives de la séquence d'entrée et du filtre adaptatif égales à 64 et le pas d'adaptation $\mu = 0.8$. Les autres paramètres sont donnés par : $\beta = 100$, $\delta = 10^{-2}$, $\rho = \frac{5}{L}$, $\lambda' = 0.6$, $(\lambda, \kappa_0) = (0.995, 1.1)$ et $s_0 = 1000$.

Sur la figure (5.8), les courbes en pointillés représentent la convergence mesurée par N_m obtenue à partir des algorithmes non robustes et les courbes en trait plein donnent la mesure de N_m issue des algorithmes robustes, $RNLMS$, $RPNLMS$, et $RPNLMS++$.

La figure (5.8) montre que les algorithmes non robustes divergent pendant la phase de double parole, ce qui introduit un écho dans le signal de retour. En revanche, les algorithmes robustes suppriment, pendant cette phase de double parole, le problème de divergence du filtre adaptatif.

En conclusion, les algorithmes robustes et en particulier l'algorithme $RPNLMS++$ sont, malgré un ralentissement du taux de convergence qu'ils provoquent, plus performants que les algorithmes non robustes.

L'intérêt que présente l'algorithme $RPNLMS++$ nous amène à poursuivre son étude en le traitant par

blocs. Ce traitement par bloc nous permettra une implantation moins complexe d'un filtre adaptatif FIR par la mise en oeuvre de la transformée de Fourier. Cette implantation qui fait intervenir une série de produits de convolution peut aussi être entièrement réalisée au moyen de la $GSFNT$.

5.3.3.6 Traitement par blocs de l'algorithme robuste PNLMS++ (RBPNLMS++)

Nous avons vu dans le paragraphe précédent que les différents algorithmes robustes ($RNLMS$, $RPNLMS$, $RPNLMS++$) permettent de limiter le problème de la divergence du filtre adaptatif pendant la phase de double parole et que l'algorithme $RPNLMS++$ présente une meilleure convergence par rapport aux autres algorithmes.

Il est alors intéressant de traiter l'algorithme $RPNLMS++$ par bloc ($RBPNLMS++$) afin de réduire sa complexité de calcul, par la mise en oeuvre de la $GSFNT$.

L'algorithme $RBPNLMS++$ est obtenu à partir des algorithmes, $RBNLMS$ et $RBPNLMS$ que nous présentons dans les paragraphes qui suivront.

5.3.3.6.a Algorithme RNLMS par bloc (RBNLMS)

L'algorithme $RBNLMS$ est obtenu à partir du traitement par bloc de l'algorithme $RNLMS$, qui est lui même défini à partir de l'algorithme $RLMS$ avec un pas d'adaptation variable. Nous proposons donc de traiter tout d'abord l'algorithme $RLMS$ par bloc d'échantillons. Pour cela, rappelons que la sortie du filtre adaptatif de longueur L dans le cas d'un traitement par bloc, définie dans le paragraphe 3.2.1 du chapitre 3, est donnée par :

$$\tilde{y}_k = \begin{bmatrix} x_{kN} & x_{kN-1} & \dots & x_{kN-L+1} \\ x_{kN+1} & x_{kN} & \dots & x_{kN-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{kN+N-1} & x_{kN+N-2} & \dots & x_{kN+N-L} \end{bmatrix} \begin{bmatrix} \hat{w}_k(0) \\ \hat{w}_k(1) \\ \vdots \\ \hat{w}_k(L-1) \end{bmatrix}$$

$$= \Re_k \hat{w}_k \tag{5.21}$$

où $\Re_k$ est une matrice de Toeplitz de taille $(L \times N)$ et k est l'indice de bloc $(k \in \mathbb{N})$.

Nous considérons les vecteurs de données $\tilde{d}_k$ (signal désiré), $\tilde{y}_k$ (écho estimé) et ϵ_k (signal d'erreur), tous de longueur N définis par :

$$\tilde{d}_k = \begin{bmatrix} d_{kN} & d_{kN+1} & \dots & d_{(k+1)N-1} \end{bmatrix}^T \tag{5.22}$$

$$\tilde{y}_k = \left[\begin{array}{cccc} \hat{y}_{kN} & \hat{y}_{kN+1} & \dots & \hat{y}_{(k+1)N-1} \end{array} \right]^T \tag{5.23}$$

$$\epsilon_k = \tilde{d}_k - \tilde{y}_k = \left[\begin{array}{cccc} e_{kN} & e_{kN+1} & \dots & e_{(k+1)N-1} \end{array} \right]^T \tag{5.24}$$

En utilisant le même principe de traitement par bloc, que celui utilisé en chapitre (3), la mise à jour des coefficients du filtre adaptatif par l'algorithme $RBLMS$ est donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \mu_B . \Re_k^T . sign\left\{\epsilon_k\right\} \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \tag{5.25}$$

où μ_B est le pas d'adaptation.

Les différents paramètres présentés dans cette équation sont définis dans le paragraphe (5.3.3.1).

Le calcul du produit matriciel, présent dans les équations (5.21) et (5.25), nécessite des opérations lourdes. Pour réduire la complexité de ce calcul, nous proposons de réaliser ce calcul matriciel par la convolution circulaire.

L'algorithme $RBLMS$ consiste à mettre en oeuvre la convolution circulaire entre le vecteur des coefficients du filtre $\tilde{W}_k$ et le vecteur du signal d'entrée $\tilde{X}_k$, tous deux de longueur $M \geq N+L-1$, et définis respectivement par :

$$\tilde{W}_k = \left[\begin{array}{cc} \hat{w}_k^T \mid & 0_{1\times M-L} \end{array} \right]^T \tag{5.26}$$

$$\tilde{X}_k = \left[\begin{array}{cc} \tilde{x}_k^T \mid & 0_{1\times M-(N+L-1)} \end{array} \right]^T \tag{5.27}$$

Notons $\tilde{Y}_k$ le produit de convolution circulaire entre $\tilde{X}_k$ et $\tilde{W}_k$:

$$\tilde{Y}_k = \tilde{X}_k * \tilde{W}_k \tag{5.28}$$

où $*$ représente le produit de convolution circulaire entre deux vecteurs.

Pour le calcul de la FFT, il faut choisir M égale à une puissance de 2. En général, on choisit $M = N+L-1$ si cette valeur est une puissance de 2 sinon il faut compléter les vecteurs par un nombre de coefficients nuls afin que leur dimensions soit une puissance de deux. Pour simplifier, on suppose que $M \geq N + L - 1$.

Le vecteur $\tilde{Y}_k$ est de longueur $N + L - 1$. Ses N dernières composantes correspondent au résultat de la convolution linéaire donnée par l'équation (5.21), c'est à dire aux composantes de $\tilde{y}_k$. Les $(M - N)$ premières composantes, résultant de la convolution circulaire, doivent être écartées.

De manière formelle, on écarte ces $(M - N)$ premières composantes en considérant la matrice P_2, de

dimension $(N \times M)$, définie par :

$$P_2 = \left[\begin{array}{c|c} 0_{N\times(M-N)} & I_N \end{array} \right] \tag{5.29}$$

où $0_{N\times(M-N)}$ est la matrice nulle de taille $N \times (M-N)$ et I_N est la matrice identité de taille $N \times N$.

Nous obtenons alors :

$$\begin{aligned} \tilde{y}_k &= P_2 \tilde{Y}_k \\ &= P_2 \left(\tilde{X}_k * \tilde{W}_k \right) \end{aligned} \tag{5.30}$$

Cette convolution, donnée par l'équation (5.30), peut être calculée dans le domaine de Fourier par :

$$\begin{aligned} \tilde{y}_k &= P_2 \tilde{Y}_k \\ &= P_2 FFT^{-1} \left(FFT\left(\tilde{X}_k\right) \bullet FFT\left(\tilde{W}_k\right) \right) \end{aligned} \tag{5.31}$$

En reprenant l'équation (5.24) exprimant le signal d'erreur, nous obtenons alors :

$$\epsilon_k = \tilde{d}_k - P_2 FFT^{-1} \left(FFT\left(\tilde{X}_k\right) \bullet FFT\left(\tilde{W}_k\right) \right) \tag{5.32}$$

Le calcul de l'erreur par blocs, nécessite ainsi, grâce à l'utilisation de la FFT, moins d'opérations que dans le cas du $RLMS$ temporel par blocs.

Comme pour le calcul d'erreur, la mise à jour du filtre adaptatif $\hat{w}_k$ par l'algorithme $RBLMS$ peut être réalisée par la convolution circulaire. En effet :

$$\begin{aligned} \hat{w}_{k+1} &= \hat{w}_k + \mu_B . \Re_k^T . sign\left\{\epsilon_k\right\} \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \\ &= \hat{w}_k + \mu_B \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \sum_{j=kN}^{kN+N-1} sign\left\{e_j\right\} . \chi_j \\ &= \hat{w}_k + \mu_B \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \varphi_k \end{aligned} \tag{5.33}$$

où $\chi_j = \left[\begin{array}{cccc} x_j & x_{j-1} & \ldots & x_{j-L+1} \end{array} \right]^T$.

$\varphi_k = \Re_k^T . sign\left\{\epsilon_k\right\}$ correspond à une corrélation croisée dont les éléments φ_{ik} sont donnés par :

$$\varphi_{ik} = \sum_{m=0}^{N-1} x_{kN+m-i} sign\left\{e_{kN+m}\right\}, \qquad 0 \leq i \leq L-1 \tag{5.34}$$

En posant $n = kN + m$, les éléments φ_{ik} peuvent s'écrire sous forme d'une convolution de la manière

suivante :

$$\varphi_{ik} = \sum_{n=kN}^{(k+1)N-1} x_{n-i} sign\left\{e_n\right\} = x_{-i} * sign\left(e_i\right)$$

soit :

$$\varphi_k = \tilde{x}_{-k} * sign\left(\epsilon_k\right) \tag{5.35}$$

L'équation (5.33) peut donc s'écrire :

$$\tilde{W}_{k+1} = \tilde{W}_k + \mu_B \left(\tilde{X}_{-k} * sign\left\{\tilde{E}_k\right\}\right) \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \tag{5.36}$$

où

$$\tilde{X}_{-k} = \left[\begin{array}{cccc|ccc} x_{kN+N-1} & x_{kN+N-2} & \ldots & x_{kN-L+1} & 0 & \ldots & 0 \end{array}\right]^T \tag{5.37}$$

$$= \left[\begin{array}{c|c} \tilde{x}_{-k}^T & 0_{1\times M-(N+L-1)} \end{array}\right]^T$$

$\tilde{E}_k$ est la séquence d'erreur définie par :

$$\tilde{E}_k = \left(\left[\begin{array}{c|c} \epsilon_k^T & 0_{1\times(L-1)} \end{array}\right]\right)^T \tag{5.38}$$

Soit, en introduisant la transformée de Fourier :

$$\tilde{W}_{k+1} = \tilde{W}_k + \mu_B FFT^{-1}\left(FFT\left(\tilde{X}_{-k}\right) \bullet FFT\left(sign\left\{\tilde{E}_k\right\}\right)\right) \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \tag{5.39}$$

Afin d'obtenir un filtre $\hat{w}_k$ de longueur L, il suffit de supprimer les $(M-L)$ premières composantes du vecteur $\tilde{W}_k$ en appliquant, au résultat de la transformée de Fourier inverse de l'équation (5.39), une matrice P_1 définie par :

$$P_1 = \left[\begin{array}{c|c} 0_{L\times(M-L)} & I_L \end{array}\right] \tag{5.40}$$

Dans ce cas, la mise à jour des coefficients du filtre adaptatif par l'algorithme $RBLMS$, mettant en oeuvre le calcul de la FFT est donnée par l'équation :

$$\hat{w}_{k+1} = \hat{w}_k + \mu_B.P_1 FFT^{-1}\left(FFT\left(\tilde{X}_{-k}\right) \bullet FFT\left(sign\left\{\tilde{E}_k\right\}\right)\right) \psi\left(\frac{|\epsilon_k|}{s_k}\right) s_k \tag{5.41}$$

A partir de l'algorithme $RBLMS$ défini par l'équation (5.41), nous pouvons définir l'algorithme $RBNLMS$ en remplaçant le pas d'adaptation μ_B par :

$$\mu_B(k) = \frac{\mu_B}{\mathbf{R}_{xx}(k) + \beta} = \frac{\mu_B}{(\tilde{x}_k * \tilde{x}_{-k}) + \beta} \tag{5.42}$$

où β est un facteur permettant de suivre plus ou moins rapidement les variations d'énergie dans le signal d'entrée $\{x\}$.

La fonction d'autocorrélation $\mathbf{R}_{xx}(k)$ peut être réalisée avec une complexité réduite de calcul par la mise en oeuvre de la FFT et de la $GSFNT$.

Cette normalisation du pas d'adaptation par rapport à la fonction d'autocorrélation $\mathbf{R}_{xx}(k)$ permet d'obtenir une convergence uniforme durant le processus d'adaptation.

En tenant compte des équations (5.41) et (5.42), la mise à jour des coefficients du filtre adaptatif par l'algorithme $RBNLMS$ est donnée par : [Benesty2001, Alaeddine2006]

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{\mathbf{R}_{xx}(k) + \beta}.(\tilde{x}_{-k} * sign\{\epsilon_k\})\,\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k \tag{5.43}$$

$$= \hat{w}_k + \frac{\mu_B}{P_1.FFT^{-1}\left(FFT\left(\tilde{X}_k\right) \bullet FFT\left(\tilde{X}_{-k}\right)\right) + \beta}.P_1 FFT^{-1}\left(FFT\left(\tilde{X}_{-k}\right) \bullet FFT\left(sign\left\{\tilde{E}_k\right\}\right)\right)\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k$$

Pour une réduction de la complexité de calcul des coefficients $\hat{w}_k$, nous proposons de substituer au calcul de la FFT celui de la $GSFNT$.

5.3.3.6.b Algorithme RPNLMS par bloc (RBPNLMS)

De la même manière que dans le paragraphe précédent, nous proposons de traiter l'algorithme $RPNLMS$ par bloc d'échantillons au lieu d'échantillon par échantillon. Cet algorithme résulte directement de l'algorithme $RBNLMS$ en remplaçant le pas d'adaptation μ_B par :

$$\mu_B(k) = \frac{\mu_B G_k}{G_k \mathbf{R}_{xx}(k) + \beta} = \frac{\mu_B G_k}{G_k(\tilde{x}_k * \tilde{x}_{-k}) + \beta} \tag{5.44}$$

G_k est une matrice diagonale définie dans le paragraphe 5.3.3.3.

Dans ce cas, la mise à jour des coefficients du filtre adaptatif par l'algorithme $RBPNLMS$ est pour un bloc k, définie par : [Benesty2001, Alaeddine2006]

$$\begin{aligned}\hat{w}_{k+1} &= \hat{w}_k + \frac{\mu_B G_k}{G_k \mathbf{R}_{xx}(k)+\beta}.\left(\tilde{x}_{-k} * sign\left\{\epsilon_k\right\}\right)\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k \\ &= \hat{w}_k + \frac{\mu_B G_k}{G_k.P_1 FFT^{-1}\left(FFT\left(\tilde{X}_k\right)\bullet FFT\left(\tilde{X}_{-k}\right)\right)+\beta}.P_1 FFT^{-1}\left(FFT\left(\tilde{X}_{-k}\right)\bullet FFT\left(sign\left\{\tilde{E}_k\right\}\right)\right)\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k\end{aligned} \tag{5.45}$$

L'algorithme $RPNLMS++$ présentant une meilleure performance par rapport aux autres algorithmes du filtrage adaptatif $RNLMS$ et $RPNLMS$, il est donc intéressant de le traiter par bloc afin de réduire sa complexité de calcul au maximum par la mise en oeuvre de la $GSFNT$.

5.3.3.6.c Algorithme RPNLMS++ par bloc (RBPNLMS++)

L'algorithme $RBPNLMS{+}{+}$, défini comme étant une combinaison entre les deux algorithmes $RBNLMS$ et $RBPNLMS$, la mise à jour des coefficients du filtre adaptatif de cet algorithme est, pour un bloc d'indice k, définie à partir des équations (5.43) et (5.45) de la manière suivante :

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{\mathbf{R}_{xx}(k)+\beta}.\left(\tilde{x}_{-k} * sign\left\{\epsilon_k\right\}\right)\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k, \text{ si l'indice } k \text{ du bloc est pair}$$

$$\hat{w}_{k+1} = \hat{w}_k + \frac{\mu_B}{G_k\mathbf{R}_{xx}(k)+\beta}.G_k\left(\tilde{x}_{-k} * sign\left\{\epsilon_k\right\}\right)\psi\left(\frac{|\epsilon_k|}{s_k}\right)s_k, \text{ si l'indice } k \text{ du bloc est impair}$$

L'adaptation du filtre adaptatif par l'algorithme $RBPNLMS++$ permettra ainsi d'améliorer le taux de convergence de ce filtre. Afin de comparer les performances des différents algorithmes robustes traités par blocs, nous présenterons, dans le paragraphe suivant, le taux de convergence du filtre adaptatif à partir de ces algorithmes.

5.4 Comparaison des performances des différents algorithmes robustes (BRNLMS, BRPNLMS, BRPNLMS++)

Des simulations numériques ont été réalisées pour évaluer les performances des algorithmes robustes traités par blocs ($BRNLMS$, $BRPNLMS$, $BRPNLMS++$) au moyen de la technique du Sliding Généralisé appliquée à la transformée en nombres de Fermat ($GSFNT$) . Pour ces simulations, les algorithmes de filtrage adaptatif par bloc ($BNLMS$, $BPNLMS$, $BPNLMS++$), présentés dans le chapitre 3, sont comparés à ces nouveaux algorithmes robustes, traités par blocs. Les différents algorithmes sont évalués à partir de deux signaux de parole, lointaine et locale, donnés par la figure (5.7), en comparant des mesures objectives comme la convergence du filtre adapté mesurée par N_m.

Pour ces simulations, la longueur N de la séquence d'entrée et celle L du filtre adaptatif sont respectivement fixées à 64 et 193. Les autres paramètres sont donnés par : $\mu_B = 0.8$, $\beta = 100$, $\delta = 10^{-2}$, $\rho = \frac{5}{L}$, $\lambda' = 0.6$, $(\lambda, \kappa_0) = (0.995, 1.1)$ et $s_0 = 1000$.

Toutes les opérations arithmétiques seront réduites par un modulo égal au nombre de Fermat $F_4 = 2^{16} + 1 = 65537$. Les convolutions rencontrées dans les différents algorithmes du filtrage adaptatif sont calculées en utilisant la $GSFNT$. Les différentes séquences à convoluer sont toutes de longueur égale à $M = 256$ échantillons. Le terme générateur α pour les calculs de la $GSFNT$ est fixé à $\alpha = \sqrt[8]{2} = 5574$.

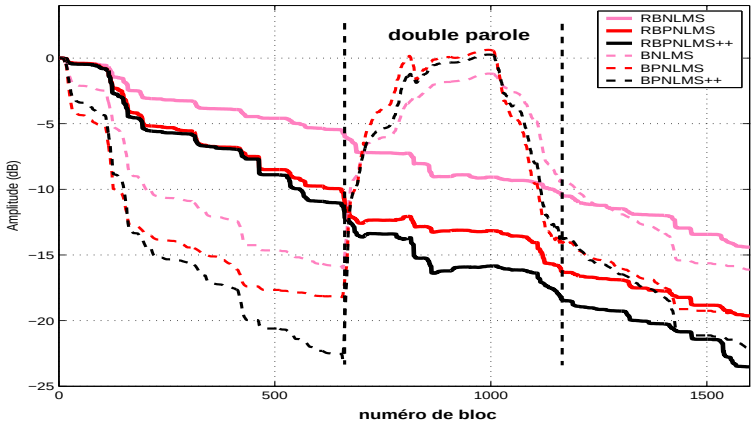

FIG. 5.9 – Convergence des coefficients du filtre adaptatif, mesurée par $\mathbf{N_m}$, pour les différents algorithmes robustes et non-robustes traités par bloc

La figure (5.9) montre que les algorithmes robustes proposés, traités par blocs (courbes en lignes continues), empêchent, pendant la phase de double parole, la divergence du filtre adaptatif provoquée par des algorithmes bloc non robustes (courbes en lignes discontinues). Cette figure montre aussi que l'algorithme $RBPNLMS++$ présente une meilleure convergence par rapport aux autres algorithmes du filtrage adaptatif. Il est alors intéressant d'utiliser cet algorithme dans notre système combiné ($RBPNLMS++$ et le post-filtre) afin de pouvoir envisager un système d'annulation d'écho acoustique complet avec une charge de calcul réduite grâce à l'utilisation de la technique du Sliding Généralisé appliquée à la FNT ($GSFNT$).

5.5 Résultats expérimentaux du système combiné proposé

De manière à évaluer les performances du système combiné défini ci-dessus, différentes simulations numériques, ont été réalisées.

Le système combiné a été testé au moyen de deux signaux de parole, un signal pour la parole lointaine et l'autre pour la parole locale présentés respectivement sur la figure 5.10-(a) et (c). Le système combiné est basé sur un détecteur de double parole (DTD). Ce DTD est composé de deux VAD, $VAD1$ pour détecter la parole lointaine et $VAD2$ pour détecter la parole locale (figure 5.2).

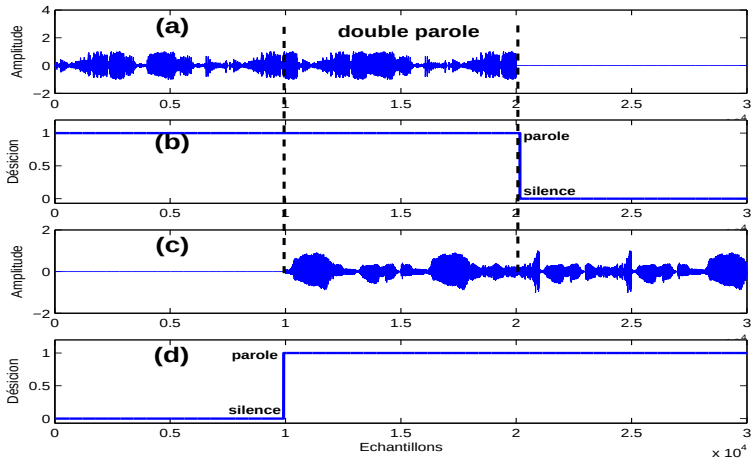

FIG. 5.10 – (a) : signal lointain $\{x\}$; (c) : signal proche $\{s\}$; (b) et (d) : décisions respectives des détecteurs d'activité de la voix VAD1 et VAD2

Pour les deux signaux de parole présentés par la figure (5.10), les bonnes décisions du détecteur de double parole (DTD) ont été déterminées en l'absence de bruit. Les décisions des deux détecteurs d'activité de la voix $VAD1$ et $VAD2$ sont respectivement représentées par les figures 5.10-(b) et (d).

A partir de ces décisions données par le détecteur de double parole, nous distinguons trois phases de parole :

• Phase 1 : $VAD1 = 1$ et $VAD2 = 0$, c'est à dire, le cas de simple parole. Dans ce cas, le filtrage adaptatif est déclenché pour annuler l'écho acoustique au moyen de l'algorithme $RBPNLMS++$. Le post-filtre est placé juste après l'annulateur d'écho pour atténuer au maximum l'écho résiduel en sortie de l'annulateur d'écho.

• Phase 2 : $VAD1 = 1$ et $VAD2 = 1$, c'est le cas de double parole. Dans ce cas, le filtrage adaptatif est arrêté et le post-filtre est déclenché.

• Phase 3 : $VAD1 = 0$ et $VAD2 = 1$. Dans ce cas, le signal d'écho est absent et nous n'avons besoin ni de filtrage adaptatif ni de post-filtre et le signal local $\{s\}$ est transmis à l'autre extrémité du canal.

Par la suite, une programmation de notre système d'annulation d'écho sous un environnement *MatLab* a été réalisée en utilisant, dans un premier temps, la transformée en nombres de Fermat (FNT) puis dans un second temps, la technique du Sliding Généralisé appliquée à la FNT ($GSFNT$) pour optimiser le nombre de convolutions. Les différentes séquences à traiter par la $GSFNT$ dans l'algorithme $RBPNLMS++$ seront de longueur $M = N + L - 1 = 256$, où la longueur L du filtre adaptatif est de 193 et le nombre N d'échantillons est de 64. La quantification est réalisée sur $b = 16$ bits et le terme générateur α est égal à la représentation de $\sqrt[8]{2} = 5574$ dans le corps de Galois $GF(F_4)$, avec $F_4 = 2^b + 1 = 2^{16} + 1 = 65537$.

Pour le traitement du post-filtre, les différentes séquences ne présentant pas de chevauchement, le nombre

d'opérations pour la mise en oeuvre de la FNT est le même que celui de la mise en oeuvre de la $GSFNT$. Dans ce cas, la longueur N de la séquence est fixée à 64 et le terme générateur α est fixé à $\sqrt{2}$.

Pour effectuer les différentes simulations, nous avons utilisé des mesures objectives telles que l'atténuation de l'écho $(ERLE)$, fournie par le système combiné pendant la phase de simple parole et le rapport signal à bruit (SNR) fourni pendant la phase de double parole.

Les mesures, données en dB, sont effectuées de manière classique en utilisant des blocs successifs de N échantillons. Pour un bloc d'indice k, l'atténuation de l'écho et le rapport signal à bruit définis précédemment s'écrivent respectivement :

$$ERLE\,(k) = 10log_{10}\left[\frac{\sum_{n=(k-1)N+1}^{kN}\left(y\,(n)\right)^2}{\sum_{n=(k-1)N+1}^{kN}\left(y\,(n)-\hat{s}\,(n)\right)^2}\right] \tag{5.46}$$

$$SNR\,(k) = 10log_{10}\left[\frac{\sum_{n=(k-1)N+1}^{kN}\left(s\,(n)\right)^2}{\sum_{n=(k-1)N+1}^{kN}\left(s\,(n)-\hat{s}\,(n)\right)^2}\right] \tag{5.47}$$

où $y\,(n)$ et $s\,(n)$ désignent respectivement le signal d'écho et le signal de parole locale. $\hat{s}\,(n)$ désigne l'écho résiduel traité par le système combiné (l'algorithme $RBPNLMS++$ et le post-filtre) dans le cas de simple parole et le signal de parole locale traité par le post-filtre dans le cas de double parole.

Pour la simulation, les valeurs des différents paramètres utilisés dans notre système combiné sont données par :

$\mu_B = 0.8$ est le pas d'adaptation, $\beta = 100$, $\delta = 10^{-2}$, $\rho = \frac{5}{L}$, $\lambda' = 0.6$, $(\lambda, \kappa_0) = (0.995, 1.1)$, $s_0 = 1000$ et $\tilde{\beta} = 0.99$.

Les résultats de simulations de ces mesures objectives sont représentés sur la figure (5.11).

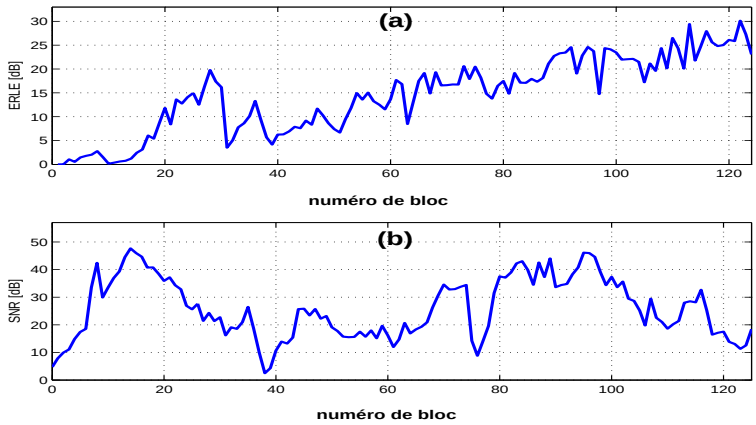

FIG. 5.11 – (a) : Atténuation d'écho fournie par le système combiné (b) : rapport signal à bruit entre les signaux d'entrée et sortie du système combiné

La figure (5.11-a), représentant une atténuation d'écho de l'ordre de 20 dB dans le cas de simple parole, montre le bon fonctionnement de l'annulateur d'écho.

La figure (5.11-b) montre que le rapport signal à bruit fourni par le post-filtre est de l'ordre de 30 dB, ce qui signifie que l'énergie du signal $(s - \hat{s})$ est inférieure à celle du signal de parole locale s.

L'autre résultat important concerne le signal $\{\hat{s}\}$ de parole locale estimé par le système combiné en sortie du post-filtre. Ce signal représenté sur la figure (5.12-c) montre que le système combiné, a rendu l'écho inaudible à la sortie du système dans le cas de simple parole. Dans le cas de double parole, cette figure (5.12) ne révèle pas de différence marquée entre le signal de parole locale et celui de la parole locale estimée.

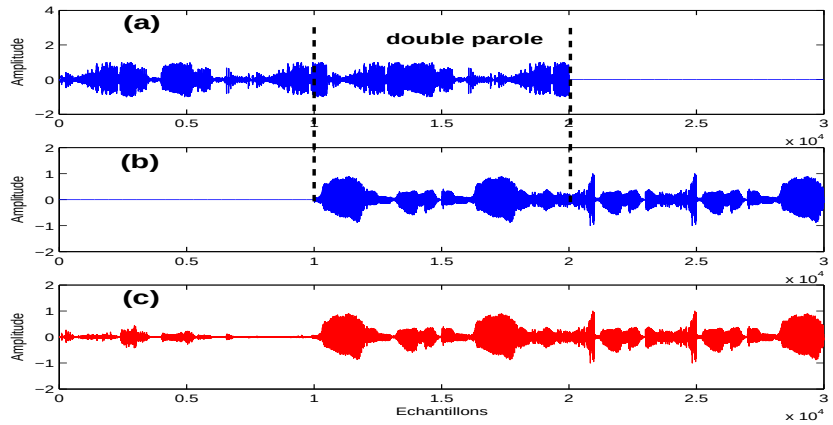

FIG. 5.12 – (a) : signal lointain (b) : signal proche (c) : signal proche estimé

A partir de ces résultats présentés par les figures (5.11) et (5.12), nous pouvons conclure que notre système donne de bons résultats dans la mesure où il rend l'écho inaudible sans engendrer de dégradations sur le signal de parole locale.

Concernant la complexité de calcul de notre système combiné, les figures (5.13) et (5.14) donnent une idée sur l'ordre de grandeur du nombre d'opérations requises par l'algorithme $RBPNLMS++$ pour deux méthodes d'implantation, la $GSFNT$ et la FNT.

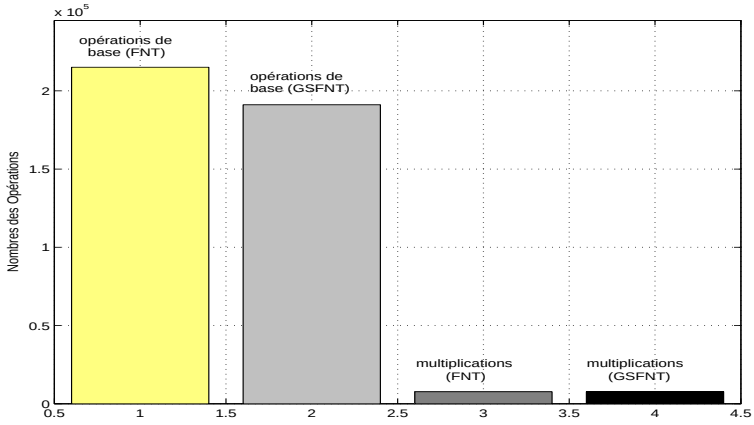

FIG. 5.13 – Opérations requises par l'algorithme RBPNLMS++ pour 10 blocs

La figure (5.13) montre tout l'intérêt que présente la $GSFNT$ en terme de réduction du nombre d'opéra-

tions. En effet, l'implantation de l'algorithme $RBPNLMS++$ par la $GSFNT$ nécessite moins d'opérations d'additions, de soustractions et de décalage de bits que celles que nécessite l'implantation de ce même algorithme par la FNT.

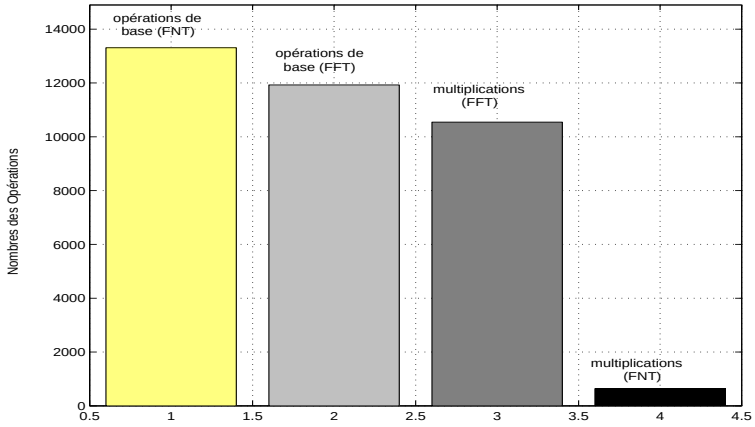

FIG. 5.14 – Opérations requises par le post-filtre pour 10 blocs

La figure (5.14) fournit une idée sur l'ordre de grandeur du nombre d'opérations requises par le post-filtre.

L'implantation de celui-ci par la FNT permet une réduction significative du nombre de multiplications réelles d'un facteur supérieur à 16 pour chaque bloc de signal traité.

5.6 Evaluations subjectives de la qualité d'écoute

Les observations objectives, telles que la mesure de la convergence des coefficients du filtre adaptatif ou de l'atténuation de l'écho acoustique, restant insuffisantes pour bien évaluer le système d'annulation d'écho acoustique, des tests d'écoute ont été réalisés pour obtenir une évaluation subjectives de la qualité d'écoute de l'écho résiduel issu de chacun des systèmes d'annulation d'écho suivants :

• Un système adapté par l'algorithme $BPNLMS++$ testé dans le cas de simple et de double parole.

• Un système combiné ($RBPNLMS++$ et post-filtre) testé dans le cas de double parole.

Les résultats de ces tests ont montré que :

- Le système d'annulation d'écho adapté par l'algorithme $BPNLMS++$ fonctionne correctement dans le cas de simple parole, dans la mesure où l'écho résiduel fourni est moins audible. En revanche, dans le cas de double parole, ce système fournit une dégradation sur le signal de la parole locale et un écho audible.

- Le système combiné proposé ($RBPNLMS++$ et post-filtre) testé dans le cas de double parole, fournit un écho moins audible sans engendrer de dégradation sur le signal de parole locale.

5.7 Conclusion

Ce chapitre a permis d'introduire le principe du système combiné pour réaliser l'annulation d'écho. L'objectif du système combiné est de dépasser les limites de l'approche classique, reposant sur l'identification de la réponse impulsionnelle de couplage à l'aide d'un filtre adapté par des algorithmes de gradient stochastique ($NLMS$, $PNLMS$, $PNLMS++$).

Un système combiné est donc la combinaison d'un annulateur d'écho avec un post-filtre dont le rôle est d'atténuer l'écho résiduel encore présent après l'annulation de l'écho et sans engendrer de dégradations perceptibles sur le signal de parole locale.

L'originalité de ce système combiné repose sur l'estimation de l'écho résiduel à partir d'un mélange entre le signal de parole locale et un signal d'erreur.

Notre étude a consisté à apporter une amélioration au système combiné. Cette amélioration réside dans l'adaptation du filtre adaptatif par un algorithme plus robuste qui limite le problème de divergence de ce filtre dans le cas de double parole. Cet algorithme, intitulé $RPNLMS++$, a été traité par blocs afin de pouvoir envisager une implantation du système combiné ($RBPNLMS++$ et post-filtre) dans un microprocesseur de traitement de signal. Pour une complexité de calculs réduite au maximum, nous avons proposé d'implanter ce système combiné par la mise en oeuvre de la $GSFNT$.

Des mesures objectives ont été réalisées sous le logiciel *MatLab* et ont permis de conclure que le système combiné proposé possède de meilleures performances concernant l'annulation d'écho.

Enfin, les tests d'écoute ont permis de vérifier les limites du système d'annulation d'écho adapté par l'algorithme $BPNLMS++$ et l'intérêt que présente le système combiné dans le cas de double parole.

Chapitre 6

Filtre à Délais Multiples

6.1 Introduction

Dans le cadre du problème d'annulation d'écho acoustique, la taille des réponses impulsionnelles peut atteindre plusieurs centaines de milli-secondes, ce qui entraîne, pour des fréquences d'échantillonnage de 8 kHz ou 16 kHz, des transformées de plusieurs milliers de coefficients. Ces tailles sont souvent inadaptées aux processeurs des signaux numériques (DSP) , sur lesquels sont implantés les algorithmes du filtrage adaptatif.

A ce titre, nous proposons d'utiliser un algorithme développé initialement dans le cadre de l'estimation des réponses impulsionnelles de grandes tailles. Cet algorithme consiste à décomposer le filtre adaptatif initial en sous blocs de moindre taille pour adapter l'implantation des algorithmes du filtrage adaptatif sur les DSP. Cette technique est connue sous le nom de Filtre à Délais Multiples ($MDF : Multi - Delay\,Filter$).

L'étude principale, présentée dans ce chapitre, consiste, dans un premier temps, à proposer une amélioration de l'algorithme MDF afin de réduire sa complexité de calcul. Nous proposons ensuite une modification de l'algorithme $BPNLMS++$ par la nouvelle procédure de l'algorithme MDF afin de minimiser la taille des transformées utilisées.

En complément, nous proposons une nouvelle méthode de gestion du pas d'adaptation des algorithmes du filtrage adaptatif, ce qui conduira à une amélioration des capacités de convergence même en présence des grandes variations de la puissance du signal d'entrée. Dans un premier temps, nous rappelons la définition et le rôle fondamental du pas d'adaptation dans les méthodes rapides du filtrage adaptatif. Dans un second temps, la méthode proposée sera explicitée et justifiée.

L'algorithme $BPNLMS++$, traité par la nouvelle procédure du MDF $(MDFP - BPNLMS++)$, utilisant la nouvelle méthode de gestion du pas d'adaptation, a ainsi été étudié et modifié pour permettre son implantation sur un processeur DSP à virgule fixe. Cette nouvelle procédure de l'algorithme $BPNLMS++$ implanté au moyen de la FNT a été évaluée par des mesures de qualité à l'aide du logiciel de programmation

MatLab.

6.2 Principe

Nous avons vu précédemment, dans le cas classique de l'annulation d'écho acoustique, que le signal d'écho $\{y\}$ est issu d'un signal distant provenant d'un locuteur lointain $\{x\}$. Le principe de l'annulation d'écho acoustique est basé sur le calcul du filtre adaptatif [Thomas1974, Simon1996]. Ce filtre, alimenté par le signal d'entrée noté $\{x\}$ (signal lointain), est adapté récursivement de façon à minimiser la moyenne quadratique de la différence entre sa sortie actuelle et le signal désiré noté $\{d\}$.

Une des procédures directes utilisées pour résoudre le problème d'annulation d'écho est l'algorithme du gradient stochastique connu sous le nom Block Least Mean Square ($BLMS : Block\,Least\,Mean\,Squares$), proposé initialement dans [Burrus1971, Clark1981]. Dans cet algorithme, les données d'entrée sont regroupées en blocs de N points et les coefficients du filtre sont maintenus constants sur la durée de chaque bloc k. Au cours du $k^{ème}$ bloc, l'équation d'adaptation du filtre est donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \mu_B \left(\tilde{x}_{-k} * \epsilon_k\right) \tag{6.1}$$

$*$ et μ_B représentent respectivement la convolution circulaire et le pas d'adaptation qui contrôle la convergence de l'algorithme $BLMS$.

$\hat{w}_k = \left[\begin{array}{cccc} \hat{w}_k(0) & \hat{w}_k(1) & \dots & \hat{w}_k(L-1) \end{array} \right]^T$ est le vecteur des coefficients du filtre adaptatif à l'instant k (L étant sa taille).

$\tilde{x}_k = \left[\begin{array}{cccc} x_{kN-L+1} & x_{kN-L+2} & \dots & x_{kN+N-1} \end{array} \right]^T$ est le vecteur d'entrée de taille $N+L-1$.

ϵ_k est le vecteur d'erreur ou le vecteur d'écho résiduel, défini comme la différence entre le signal désiré $\tilde{d}_k = \left[\begin{array}{cccc} d_{kN} & d_{kN+1} & \dots & d_{(k+1)N-1} \end{array} \right]^T$ et la sortie du filtre $\tilde{y}_k = \left[\begin{array}{cccc} \hat{y}_{kN} & \hat{y}_{kN+1} & \dots & \hat{y}_{(k+1)N-1} \end{array} \right]^T$ par :

$$\begin{aligned} \epsilon_k &= \tilde{d}_k - \tilde{y}_k \\ &= \left[\begin{array}{cccc} e_{kN} & e_{kN+1} & \dots & e_{(k+1)N-1} \end{array} \right]^T \end{aligned} \tag{6.2}$$

avec :

$$\tilde{y}_k = \tilde{x}_k * \hat{w}_k \tag{6.3}$$

$$\begin{aligned} \tilde{Y}_k &= \tilde{X}_k * \tilde{W}_k \\ &= \left(\left[\begin{array}{cc} \tilde{x}_k^T & \mid 0_{1\times(M-(N+L-1))} \end{array} \right]^T \right) * \left(\left[\begin{array}{cc} \hat{w}_k^T & \mid 0_{1\times(M-L)} \end{array} \right]^T \right) \end{aligned}$$

$$\begin{aligned}\tilde{Y}_k = FFT^{-1}\left(\tilde{\mathbf{Y}}_k\right) &= FFT^{-1}\left(FFT\left(\tilde{X}_k\right) \bullet FFT\left(\tilde{W}_k\right)\right) \\ &= FFT^{-1}\left(\tilde{\mathbf{X}}_k \bullet \tilde{\mathbf{W}}_k\right)\end{aligned} \tag{6.4}$$

où $\tilde{\mathbf{X}}_k = FFT\left(\tilde{X}_k\right)$ et $\tilde{\mathbf{W}}_k = FFT\left(\tilde{W}_k\right)$.

Le calcul de convolution donné par les équations (6.1) et (6.3) peut être calculé par la transformée de Fourier rapide (FFT) [Clark1981]. Dans ce cas, il faut compléter les vecteurs à transformer par des coefficients nuls pour effectuer la FFT sur M points tels que $M \geq N + L - 1$ et M est une puissance de 2.

Les transformations étant de longueur M, ceci peut poser des problèmes d'implantation temps réel pour des grandes valeurs de L. Ces tailles sont donc souvent inadaptées aux processeurs de signaux numériques DSP, sur lesquels sont implémentés les algorithmes du filtrage adaptatif. Pour résoudre ce problème, nous proposons d'utiliser un algorithme basé sur le principe de Filtre à Délais Multiples ($MDF : Multi - Delay\,Filter$) [Soo1987, Soo1990, Yon1994, Moulines1995, Amrane1992]. L'originalité du MDF consiste à diviser le traitement de la réponse impulsionnelle de taille L en K' sous blocs ($K' \in \mathbb{N}$) adjacents de taille L' tels que $L = K'L'$, afin de réduire la taille des transformées. Le schéma de la figure (6.1) représente une telle décomposition.

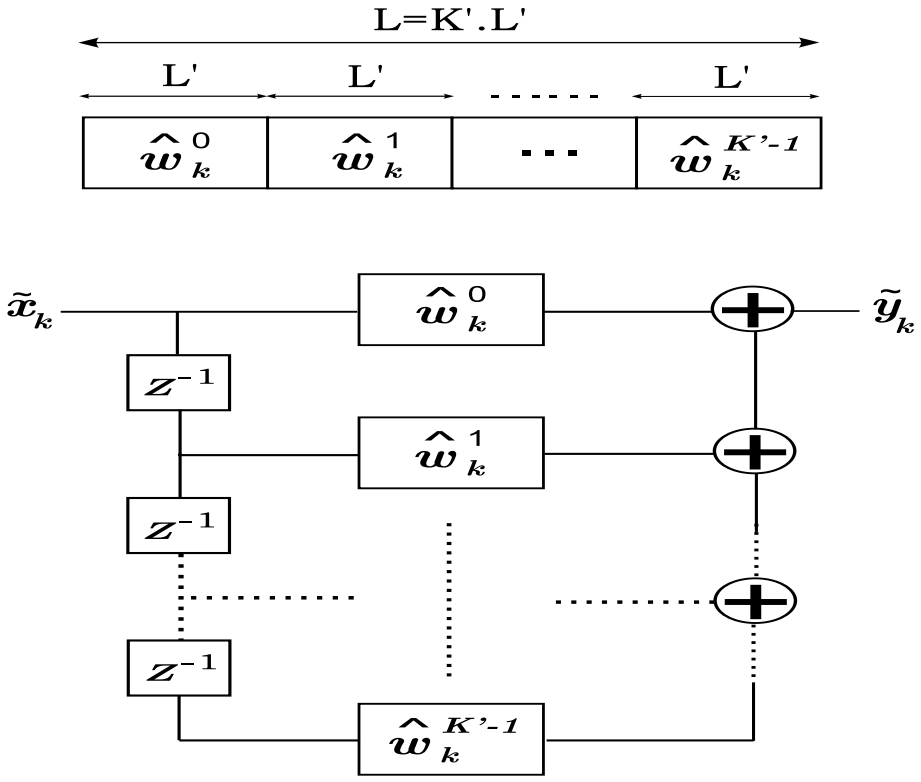

FIG. 6.1 – Décomposition d'un filtre en sous blocs pour l'implémentation du MDF. Le filtre original de taille L est décomposé en K' sous blocs de taille L' au moyen d'une structure de filtre à délais multiples

Nous définissons alors K' vecteurs temporels $\hat{w}_k^{k'}$ $(k' = 0, \ldots, K' - 1)$, de longueur $(L' \times 1)$ tels que :

$$\hat{w}_k^{k'} = \left[\begin{array}{cccc} \hat{w}_k\left(k'L'\right) & \hat{w}_k\left(k'L'+1\right) & \ldots & \hat{w}_k\left(\left(k'+1\right)L'-1\right) \end{array} \right]^T \tag{6.5}$$

avec $0 \leq k' \leq K'-1$. Soit donc :

$$\hat{w}_k = \left[\begin{array}{c} \hat{w}_k^0 \\ \ldots \\ \vdots \\ \ldots \\ \hat{w}_k^{K'-1} \end{array} \right] \tag{6.6}$$

Comme nous avons choisi de segmenter $\hat{w}_k$ en K' sous-blocs $\hat{w}_k^{k'}$, il est alors nécessaire d'effectuer K' produits de convolution entre les vecteurs $\hat{w}_k^{k'}$ et les vecteurs $\tilde{x}_k^{k'}$ de longueur $(N+L'-1)$ définis par :

$$\tilde{x}_k^{k'} = \left[\begin{array}{cccc} x_{kN-k'L'-L'+1} & \ldots & x_{kN-k'L'+N-1} \end{array} \right]^T \tag{6.7}$$

Pour cela, nous considérons deux vecteurs $\tilde{X}_k^{k'}$, $\tilde{W}_k^{k'}$ de longueur M' $(M' \geq N+L'-1)$ définis à partir des vecteurs $\tilde{x}_k^{k'}$ et $\hat{w}_k^{k'}$ en complétant ces derniers respectivement par $(M'-N-L'+1)$ et $(M'-L')$ coefficients nuls.

$$\tilde{X}_k^{k'} = \left[\begin{array}{cc} \tilde{x}_k^{k'T} & 0_{1\times(M'-(N+L'-1))} \end{array} \right]^T \tag{6.8}$$

$$\tilde{W}_k^{k'} = \left[\begin{array}{cc} \hat{w}_k^{k'T} & 0_{1\times(M'-L')} \end{array} \right]^T \tag{6.9}$$

Dans ce cas, $\tilde{Y}_k$ donné par l'équation (6.4) peut s'écrire :

$$\tilde{Y}_k = FFT^{-1}\left(\sum_{k'=0}^{K'-1} \left(FFT\left(\tilde{X}_k^{k'}\right) \bullet FFT\left(\tilde{W}_k^{k'}\right) \right) \right) \tag{6.10}$$

où $\bullet$ représente le produit terme à terme. Soit, en notant $\tilde{\mathbf{X}}_k^{k'} = FFT\left(\tilde{X}_k^{k'}\right)$, $\tilde{\mathbf{W}}_k^{k'} = FFT\left(\tilde{W}_k^{k'}\right)$ et $\tilde{\mathbf{Y}}_k = FFT\left(\tilde{Y}_k\right)$:

$$\tilde{\mathbf{Y}}_k = \sum_{k'=0}^{K'-1} \left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{W}}_k^{k'} \right) \tag{6.11}$$

Les N échantillons de la séquence $\tilde{y}_k$ sont alors obtenus par la relation :

$$\tilde{y}_k = P_{(N\times M')}.FFT^{-1}\left(\tilde{\mathbf{Y}}_k\right) = P_{(N\times M')}.FFT^{-1}\left(\sum_{k'=0}^{K'-1} \left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{W}}_k^{k'} \right) \right) \tag{6.12}$$

La matrice $P_{(N\times M')}$ de dimension $(N \times M')$ définie par $P_{(N\times M')} = \left[0_{N\times(L'-1)} \mid I_{N\times N}\right]$ permet d'écarter les $(M' - N)$ premières composantes du vecteur $\tilde{Y}_k$.

L'équation (6.2) donnant l'expression du signal d'erreur ϵ_k devient alors :

$$\epsilon_k = \tilde{d}_k - \tilde{y}_k = \tilde{d}_k - P_{(N\times M')}.FFT^{-1}\left(\sum_{k'=0}^{K'-1}\left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{W}}_k^{k'}\right)\right) \tag{6.13}$$

Par analogie avec l'équation (6.1), la mise à jour du filtre adaptatif est réalisée pour chaque indice k du bloc et chaque indice k' du sous-bloc du filtre adaptatif selon l'équation :

$$\hat{w}_{k+1}^{k'} = \hat{w}_k^{k'} + \mu_B P_{(L'\times M')} FFT^{-1}\left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{E}}_{-k}\right),\ \ 0 \le k' \le K' - 1 \tag{6.14}$$

La matrice $P_{(L'\times M')}$, de dimension $(L' \times M')$ définie par $P_{(L'\times M')} = \left[0_{L'\times(M'-L')} \mid I_{L'}\right]$, permet d'écarter les $(M' - L')$ premières composantes du vecteur $\left[FFT^{-1}\left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{E}}_{-k}\right)\right]$.

$\tilde{\mathbf{E}}_k$ est la FFT du vecteur $\tilde{E}_k$ obtenu à partir du vecteur erreur ϵ_k complété par $M' - N$ coefficients nuls :

$$\tilde{\mathbf{E}}_k = FFT\left(\tilde{E}_k\right) = FFT\left(\left[\begin{array}{c|c} \epsilon_k^T & 0_{1\times(M'-N)} \end{array}\right]\right)^T \tag{6.15}$$

Nous pouvons conclure, à partir de l'équation (6.14), que le traitement de l'algorithme $BLMS$ du filtrage adaptatif par le principe du MDF dans le contexte d'annulation d'écho acoustique, diminue la taille de la transformation utilisée, ce qui permet de mieux adapter son implémentation sur un processeur DSP avec un délai d'exécution réduit.

Pour une complexité réduite de calcul nous proposons, dans un premier temps, de simplifier l'algorithme MDF dans le cas particulier où $N = L'$. L'algorithme ainsi simplifié sera ensuite implanté de manière optimale au moyen de la transformée en nombres de Fermat (FNT).

6.3 Algorithme MDF Proposé (MDFP)

Le traitement des algorithmes du filtrage adaptatif présente une charge de calcul globale importante. C'est pourquoi, les problèmes de temps et de complexité d'exécution se posent toujours lorsque nous cherchons à implémenter un de ces algorithmes sur un processeur temps réel. Pour ces raisons, il est alors nécessaire de réduire au maximum les coûts des opérations à traiter pour pouvoir envisager une telle implémentation. L'idée est de proposer de nouveaux algorithmes plus performants que les algorithmes classiques et qui ne modifieront pas le traitement à réaliser.

Dans le paragraphe précédent, nous avons présenté le principe du MDF appliqué à l'algorithme $BLMS$ dans le cas de l'annulation d'écho acoustique. Cet algorithme apporte une amélioration au niveau du temps

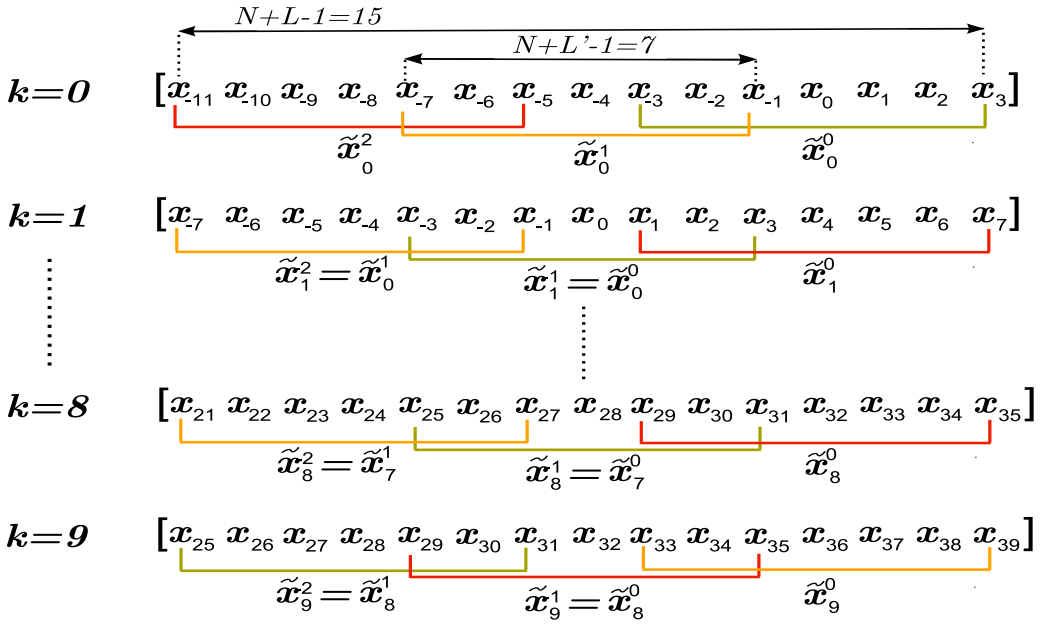

FIG. 6.2 – Composition des vecteurs $\tilde{x}_k^{k'}$ pour k=10 blocs, N=L'=4 et L=12

de calcul du fait du traitement des séquences de taille réduite.

Notre étude consiste à minimiser ce temps de calcul et la complexité d'exécution de cet algorithme. Pour cela nous prendrons $N = L'$, ce qui nous permet d'exécuter l'algorithme MDF sans changer le traitement à réaliser et de diminuer au maximum le coût de calcul en traitant à chaque itération le bloc du signal d'entrée le plus récent, c'est-à-dire $\tilde{x}_k^0$.

Pour bien illustrer l'intérêt de ce choix, nous présentons sur la figure (6.2) un exemple. Cet exemple montre une séquence de longueur $N + L - 1 = 15$, décomposée en $K' = 3$ sous-séquences de longueur $N + L' - 1 = 7$ avec $N = L' = 4$ et $L = 12$.

A partir de cet exemple, nous constatons que pour $N = L'$, nous avons $\tilde{x}_k^{k'} = \tilde{x}_{k-1}^{k'-1}$, où $k' = 1, 2, \ldots, K'-1$. Il suffit donc de calculer à chaque instant k, la FFT du sous bloc d'entrée le plus récent, c'est à dire $\tilde{x}_k^0$ sachant que la FFT des autres sous blocs d'entrée $\tilde{x}_k^{k'}$, où $k' = 1, 2, \ldots, K' - 1$, est calculée à l'instant précédent $k - 1$.

A partir de ces conclusions, nous pouvons reformuler l'équation (6.12) de la manière suivante :

$$\tilde{y}_k = P_{(N\times M')}.FFT^{-1}\left(\sum_{k'=0}^{K'-1}\left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{W}}_k^{k'}\right)\right)$$

$$= P_{(N\times M')}.FFT^{-1}\left[\tilde{\mathbf{X}}_k^0 \bullet \tilde{\mathbf{W}}_k^0 + \sum_{k'=1}^{K'-1}\left(\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{W}}_k^{k'}\right)\right]$$

$$= P_{(N\times M')}.FFT^{-1}\left[\tilde{\mathbf{X}}_k^0 \bullet \tilde{\mathbf{W}}_k^0 + \sum_{k'=1}^{K'-1}\left(\tilde{\mathbf{X}}_{k-1}^{k'-1} \bullet \tilde{\mathbf{W}}_k^{k'}\right)\right] \tag{6.16}$$

Cette dernière relation permet de calculer $\tilde{y}_k$ en faisant appel au calcul de la FFT du sous-bloc $\tilde{X}_k^0$ uniquement alors que la relation (6.12) nécessite, pour le calcul de ce même $\tilde{y}_k$, K' transformations FFT de la séquence $\tilde{X}_k^{k'}$.

L'équation (6.16) permet donc ainsi de minimiser le coût de calcul de $\tilde{y}_k$.

Le vecteur d'erreur ϵ_k peut donc s'écrire :

$$\begin{aligned}\epsilon_k &= \tilde{d}_k - \tilde{y}_k \\ &= \tilde{d}_k - P_{(N\times M')}.FFT^{-1}\left[\tilde{\mathbf{X}}_k^0 \bullet \tilde{\mathbf{W}}_k^0 + \textstyle\sum_{k'=1}^{K'-1}\left(\tilde{\mathbf{X}}_{k-1}^{k'-1} \bullet \tilde{\mathbf{W}}_k^{k'}\right)\right]\end{aligned} \tag{6.17}$$

Dans le cas de notre étude $(N = L')$, les N éléments du vecteur d'erreur ϵ_k sont donnés par :

$$\begin{cases} e_{kN+j} = \tilde{d}_{kN+j} - \Sigma_{i=0}^{L'-1}\hat{w}_k^0(i)\, x_{kN+j-i}^0, \ \text{ pour } k' = 0 \\ e_{kN+j} = \tilde{d}_{kN+j} - \Sigma_{k'=1}^{K'-1}\left(\Sigma_{i=0}^{L'-1}\hat{w}_k^{k'}(i)\, x_{(k-1)N-(k'-1)L'+j-i}^{k'-1}\right), \ \text{ pour } k' = 1, 2, \ldots, K'-1 \end{cases} \tag{6.18}$$

où $j = 0, \ldots, N-1$. A partir de l'équation (6.14), les L' éléments du filtre adaptatif $\hat{w}_k^{k'}$ sont donnés par :

$$\hat{w}_{k+1}^{k'}(i) = \hat{w}_k^{k'}(i) + \mu_B \Sigma_{j=0}^{N-1} e_{kN+j} x_{kN-k'L'+j-i}^{k'} \tag{6.19}$$

où $i = 0, \ldots, L'-1$.

Dans le cas où $n = kN - k'L' + j - i$, l'équation (6.19) devient :

$$\hat{w}_{k+1}^{k'}(i) = \hat{w}_k^{k'}(i) + \mu_B \Sigma_{n=kN-k'L'-i}^{(k+1)N-k'L'-i-1} e_{n+k'L'+i} x_n^{k'}$$

soit :

$$\hat{w}_{k+1}^{k'}(i) = \hat{w}_k^{k'}(i) + \mu_B\left(e_k(-i) * x_k^{k'}(i)\right) \tag{6.20}$$

En remplaçant $x_k^{k'}$ par $x_{k-1}^{k'-1}$ dans l'équation (6.20), nous avons :

$$\begin{cases} \hat{w}_{k+1}^0(i) = \hat{w}_k^0(i) + \mu_B\left(e_k(-i) * x_k^0(i)\right), \ \ k' = 0 \\ \hat{w}_{k+1}^{k'}(i) = \hat{w}_k^{k'}(i) + \mu_B\left(e_k(-i) * x_{k-1}^{k'-1}(i)\right), \ \ k' > 0 \end{cases} \tag{6.21}$$

Ainsi, la mise à jour du filtre adaptatif pour chaque sous-bloc k', mettant en oeuvre le calcul de la convolution circulaire par la FFT, est donnée par :

$$\begin{cases} \hat{w}_{k+1}^{0} = \hat{w}_{k}^{0} + \mu_B P_{(L' \times M')} FFT^{-1} \left(\tilde{\mathbf{X}}_{k}^{0} \bullet \tilde{\mathbf{E}}_{-k} \right), \ k' = 0 \\ \hat{w}_{k+1}^{k'} = \hat{w}_{k}^{k'} + \mu_B P_{(L' \times M')} FFT^{-1} \left(\tilde{\mathbf{X}}_{k-1}^{k'-1} \bullet \tilde{\mathbf{E}}_{-k} \right), \ k' = 1, 2, \ldots, K' - 1 \end{cases} \tag{6.22}$$

Cet algorithme modifié $(MDFP)$, comparé à l'algorithme initial (MDF), nous permet, à chaque itération, de réduire de $(K' - 1)$ le nombre de transformées utilisées pour le traitement d'un sous-bloc $\tilde{x}_k^{k'}$ du signal d'entrée. Pour améliorer davantage cette réduction de calcul, nous proposerons par la suite d'implanter cet algorithme modifié par la FNT.

Dans le paragraphe suivant, nous proposons de modifier le principe de calcul du pas d'adaptation afin d'améliorer les capacités de convergence du filtre adaptatif.

6.3.1 Modification du principe de calcul du pas d'adaptation

6.3.1.1 Rôle du pas d'adaptation

Le paramètre μ_B est le pas d'adaptation qui contrôle la mise à jour du filtre adaptatif. μ_B est donc un paramètre fondamental du point de vue de la convergence des algorithmes adaptatifs.

• Si le pas d'adaptation μ_B est petit à chaque itération, la vitesse de convergence est faible et donc les capacités de suivre la réponse du chemin d'écho sont limitées.

• Si le pas d'adaptation μ_B est grand, la vitesse de convergence est grande, ainsi que les capacités de suivre la réponse du chemin d'écho.

6.3.1.2 Pas d'adaptation normalisé

Dans les algorithmes classiques du filtrage adaptatif, le pas d'adaptation μ_B est, dans le cas des signaux stationnaires, une constante réelle. Ce pas n'est plus constant dans le cas des signaux de parole qui sont fortement non stationnaires, ce qui rend ainsi plus difficile le problème de la convergence du filtre adapté par les algorithmes adaptatifs. Il est cependant d'usage de faire varier μ_B comme l'inverse de la puissance du signal entrée $\{x\}$. L'utilité d'une telle démarche est d'adapter ainsi, au caractère non stationnaire, les coefficients du filtre adaptatif à chaque bloc d'entrée, ce qui conduit à une amélioration des capacités de convergence même en présence des grandes variations de la puissance du signal d'entrée.

Dans le domaine fréquentiel ou dans le domaine de la transformée de Fourier discrète (TFD), une même démarche est envisageable. Cependant la structure des algorithmes rapides permet d'ajouter un degré de liberté supplémentaire dans la définition du pas d'adaptation. Il est alors en effet possible de pondérer le pas d'adaptation avec un poids spécifique pour chaque coefficient de la FFT. Le pas d'adaptation μ_k est alors défini à partir de μ_B selon la relation matricielle suivante : [Soo1990]

$$\mu_k = diag\left[K'.\mu_B.\Gamma_k\right] = \begin{bmatrix} Z(0) & 0 & \dots & 0 \\ 0 & Z(1) & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & Z(M-1) \end{bmatrix} \tag{6.23}$$

où

$Z(r) = K'.\mu_B.\Gamma_k(r)$, $r = 0, 1, \dots, M-1$. K' étant le nombre de sous-blocs du filtre adaptatif.

Γ_k est le vecteur de normalisation dans le domaine transformé.

Ainsi chaque composante $\Gamma_k(r)$ de Γ_k permet de normaliser de manière adaptée la composante correspondante du gradient dans le domaine de la FFT. A cet effet, chaque composante $\Gamma_k(r)$ du vecteur Γ_k est définie comme l'inverse de la puissance spectrale P_k de $\tilde{\mathbf{X}}_k$ par :

$$\Gamma_k = \begin{bmatrix} \frac{1}{P_k(0)} & \frac{1}{P_k(1)} & \dots & \frac{1}{P_k(M-1)} \end{bmatrix} \tag{6.24}$$

Afin de prendre en compte la non stationnarité du signal d'entrée $\{x\}$, la $r^{ème}$ composante $P_k(r)$ de la puissance spectrale P_k est obtenue par la relation de récurrence suivante :

$$P_{k+1}(r) = \Upsilon P_k(r) + (1-\Upsilon)\left|\tilde{\mathbf{X}}_k(r)\right|^2 \tag{6.25}$$

où $\tilde{\mathbf{X}}_k(r)$ est la $r^{ème}$ composante de la FFT du vecteur d'entrée $\tilde{x}_k$ et Υ est un coefficient de lissage constant appartenant à l'intervalle $[0, 1]$.

Pour le MDF, comme la séquence $\tilde{x}_k$ du signal d'entrée est segmentée en K' sous vecteurs $\tilde{x}_k^{k'}$, la $r^{ème}$ composante $P_k(r)$ de la puissance spectrale est obtenue par la relation de récurrence suivante : [Soo1990]

$$P_{k+1}(r') = \Upsilon P_k(r') + (1-\Upsilon)\,\Sigma_{k'=0}^{K'-1}\left|\tilde{\mathbf{X}}_k^{k'}(r')\right|^2 \tag{6.26}$$

où $r' = 0, \dots, M'-1$. La normalisation du gradient par Γ_k est un des éléments clé des performances des algorithmes rapides du filtrage adaptatif.

6.3.1.3 Méthode proposée pour le calcul du pas d'adaptation

Afin d'assurer la convergence du filtre adaptatif tout en conservant un coût de calcul réduit, une méthode proposé pour le calcul du pas d'adaptation est mise au point. En effet, le calcul du pas d'adaptation, donné par l'équation (6.23), peut être rendu moins coûteux en temps de calcul, si on propose de réaliser le calcul matriciel par le calcul vectoriel.

Le nouveau pas d'adaptation proposé, μ'_k, est donné, à chaque itération k, par la relation vectorielle

suivante :

$$\mu'_k = K'\mu_B\Gamma_k \tag{6.27}$$

La $r^{ème}$ composante $P_k(r)$ est obtenue, dans cette nouvelle proposition du pas d'adaptation, selon une nouvelle relation de récurrence simplifiée. Le fait de choisir $N = L'$ nous permet de traiter, à chaque instant k, le sous-bloc de signal d'entrée le plus récent, c'est-à-dire $\tilde{X}_k^0$, les autres sous-bloc $\tilde{X}_k^{k'}$ où $\{k' = 1, 2, \ldots, K' - 1\}$ sont déjà connus et mis en mémoire à l'instant $k - 1$. Soit donc :

$$P_{k+1}(r') = \Upsilon P_k(r') + (1 - \Upsilon)\left|\tilde{\mathbf{X}}_k^0(r')\right|^2 \tag{6.28}$$

Avec cette nouvelle normalisation par le vecteur Γ_k, nous passons ainsi du calcul matriciel avec K' transformées FFT au calcul vectoriel du nouveau pas d'adaptation μ'_k avec une seule transformée FFT, ce qui réduit davantage la charge de calcul.

6.4 L'algorithme BPNLMS++, traité par la méthode MDF proposée (MDFP-BPNLMS++)

L'algorithme $BPNLMS++$, présentant une meilleure convergence par rapport aux autres algorithmes du filtrage adaptatif, $BNLMS$ et $BPNLMS$, il est alors intéressant de traiter l'algorithme $BPNLMS++$ par le $MDFP$ $(MDFP - BPNLMS++)$, en y apportant l'amélioration proposée sur le pas d'adaptation.

Avant de présenter cette nouvelle procédure de traitement de l'algorithme $BPNLMS++$, nous rappelons tout d'abord son principe de base. L'algorithme $BPNLMS++$ est une combinaison des deux algorithmes $BNLMS$ et $BPNLMS$.

L'équation d'adaptation du filtre est donnée par :

$$\hat{w}_{k+1} = \hat{w}_k + \mu_B \frac{G_k(\epsilon_k * \tilde{x}_{-k})}{G_k(\tilde{x}_k * \tilde{x}_{-k}) + \beta} \tag{6.29}$$

β est un facteur permettant de suivre plus au moins rapidement les variations d'énergie du signal d'entrée $\{x\}$.

Pour les indices k impairs, l'algorithme $BPNLMS++$ se réduit à celui de l'algorithme $BPNLMS$ où la matrice G_k est une matrice diagonale donnée par : [Duttweiler2000]

$$G_k = diag\left[\; g_k(0), \quad \ldots, \quad g_k(L-1) \;\right]$$

avec

$$g_k(n) = \frac{\gamma_k(n)}{\frac{1}{L}\sum_{m=0}^{L-1}\gamma_k(m)} \tag{6.30}$$

où $\gamma_k(n) = max\{\rho\nu_k, |\hat{w}_k(n)|\}$, $n \in \{0, L-1\}$ et où $\nu_k = max\left\{\delta,\ |\hat{w}_k(0)|, \quad \ldots, \quad |\hat{w}_k(L-1)|\right\}$. Les termes ρ et δ sont respectivement choisis égaux à $\frac{5}{L}$ et à 10^{-2}.

Pour les indices k pairs, l'algorithme $BPNLMS++$ se réduit à celui de l'algorithme $BNLMS$ où la matrice G_k est donnée par :

$$G_k = I_L \tag{6.31}$$

I_L étant la matrice identité d'ordre L. L'algorithme $BPNLMS++$ traité par le $MDFP$ et utilisant la nouvelle méthode de calcul du pas d'adaptation est décrit par les équations de mise à jour des coefficients du filtre adaptatif, définies à chaque indice k du bloc temporel par :

$$\hat{w}_{k+1} = \hat{w}_k + \frac{G_k.(\tilde{u}_k)}{\tilde{x}_k^T G_k \tilde{x}_k + \beta} \tag{6.32}$$

où $\tilde{u}_k$ est un vecteur de k éléments défini par :

$$\tilde{u}_k = \left[\begin{array}{cccc} u_k^0 & u_k^1 & \ldots & u_k^{K'-1} \end{array}\right] \tag{6.33}$$

Chaque élément $u_k^{k'}$ du vecteur $\tilde{u}_k$ est défini comme une convolution circulaire entre le sous-vecteur d'entrée $\tilde{x}_k^{k'}$ et le vecteur d'erreur ϵ_{-k} par :

$$\left\{\begin{array}{l} u_k^0 = P_{(L'\times M')}\left(\mu'_k \bullet FFT^{-1}\left(\tilde{\mathbf{X}}_k^0 \bullet \tilde{\mathbf{E}}_{-k}\right)\right), \text{ pour } k' = 0 \\ u_k^{k'} = P_{(L'\times M')}\left(\mu'_k \bullet FFT^{-1}\left(\tilde{\mathbf{X}}_{k-1}^{k'-1} \bullet \tilde{\mathbf{E}}_{-k}\right)\right), \text{ pour } k' = 1, 2, \ldots, K'-1 \end{array}\right. \tag{6.34}$$

$\mu_k^{'}$ est le nouveau pas d'adaptation proposé en (6.3.1.3) qui permet de contrôler la vitesse de convergence de l'algorithme $MDFP - BPNLMS++$ à chaque instant d'adaptation.

Le vecteur d'erreur $\tilde{\mathbf{E}}_k$ est défini par l'équation (6.15). Les différents éléments du vecteur $u_k^{k'}$ sont donnés par :

$$u_k^{k'}(i) = \mu'_k(i)\,\Sigma_{n=kN-k'L'-i}^{(k+1)N-k'L'-i-1} e_{n+k'L'+i} x_n^{k'}$$

soit :

$$\left\{\begin{array}{l} u_k^0(i) = \mu'_k(i)\left(e_k(-i) * x_k^0(i)\right), \text{ pour } k' = 0 \\ u_k^{k'}(i) = \mu'_k(i)\left(e_k(-i) * x_{k-1}^{k'-1}(i)\right), \text{ pour} k' = 1, 2, \ldots, K'-1 \end{array}\right. \tag{6.35}$$

Notre proposition apporte une amélioration théorique à l'algorithme $BPNLMS++$ au niveau de son implémentation en utilisant des transformations traitant des séquences de taille réduite.

Par la suite, nous développerons cet algorithme afin de l'implanter au moyen de la FNT.

6.5 Comparaisons des performances des différents algorithmes dérivés du BPNLMS++

Dans cette partie, nous allons présenter les résultats des simulations numériques issues d'une comparaison entre quatre algorithmes : $BPNLMS++$, $MDF-BPNLMS++$ tous deux utilisant un pas d'adaptation μ_B fixe, $MDF-BPNLMS++$ utilisant un pas d'adaptation matriciel donné par l'équation (6.23) et l'algorithme $MDFP-BPNLMS++$ utilisant un pas d'adaptation vectoriel donné par l'équation (6.27). Cette comparaison concerne la convergence des coefficients du filtre adaptatif mettant en oeuvre la transformée de Fourier rapide (FFT) et la complexité de calcul évaluée par le nombre d'opérations nécessaires à l'implantation de ces algorithmes.

Les différents algorithmes sont évalués avec un signal d'entrée et une réponse impulsionnelle du chemin d'écho représentés respectivement sur les figures (6.3-a) et (6.3-b).

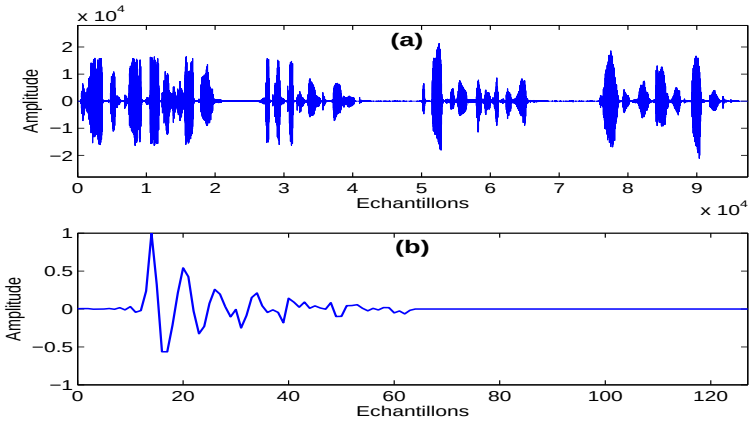

FIG. 6.3 – (a) Signal d'entrée ; (b) Réponse impulsionnelle du chemin d'écho

La réponse impulsionnelle est découpée en quatre sous-vecteurs de $L'=32$ échantillons. Pour toutes les simulations qui suivent, les valeurs des paramètres utilisés sont données par : $N=L'=32$, $L=128$, $\beta=70$, $\delta=\frac{1}{L}$, $\rho=10^{-2}$, $\mu_B=0.85$, et $\Upsilon=0.98$.

La mesure de la convergence des coefficients du filtre adaptatif par des différentes méthodes de filtrage est effectuée de manière classique en utilisant des blocs successifs de N échantillons. Pour un bloc d'indice k, cette convergence est mesurée en dB par :

$$N_m = 10log_{10}\left[\frac{\|w-\hat{w}\|^2}{\|w\|^2}\right] \tag{6.36}$$

La figure (6.4) présente la comparaison, en terme de convergence des coefficients du filtre adaptatif, des quatre algorithmes.

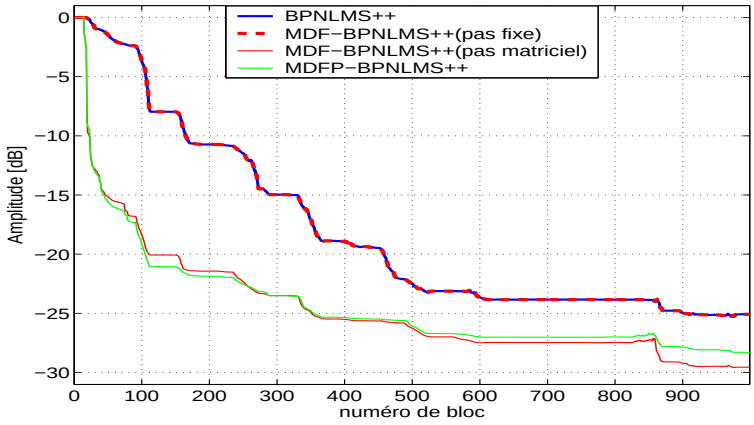

FIG. 6.4 – Convergence des coefficients du filtre adaptatif au moyen des algorithmes $BPPNLMS++$, $MDF-BPNLMS++$ avec $\mu_B = 0.85$, $MDF-BPNLMS++$ avec un pas d'adaptation matriciel et $MDFP-BPNLMS++$

Cette figure montre que l'adaptation des coefficients du filtre adaptatif par l'algorithme (*MDF-BPNLMS++*) présente la même convergence que celle donnée par l'algorithme $BPNLMS++$ tous deux réalisés avec le même pas d'adaptation μ_B fixe. Elle montre aussi que la normalisation du pas d'adaptation apporte une amélioration significative au niveau de la convergence. En effet, les algorithmes $(MDF-BPNLMS++)$, utilisant le pas d'adaptation matriciel, et $(MDFP-BPNLMS++)$, utilisant le pas d'adaptation vectoriel, présentent une meilleure convergence par rapport aux autres algorithmes, $BPNLMS++$ et $MDF-BPNLMS++$ tous deux utilisant un pas d'adaptation μ_B fixe. Cependant, ces simulations ne révèlent pas de différence marquée, en terme de convergence, entre les algorithmes $(MDF-BPNLMS++)$ avec un pas d'adaptation matriciel et $(MDFP-BPNLMS++)$.

La différence entre ces deux derniers algorithmes réside dans la complexité da calcul évaluée par le nombre d'opérations requises par leur implantation en FFT.

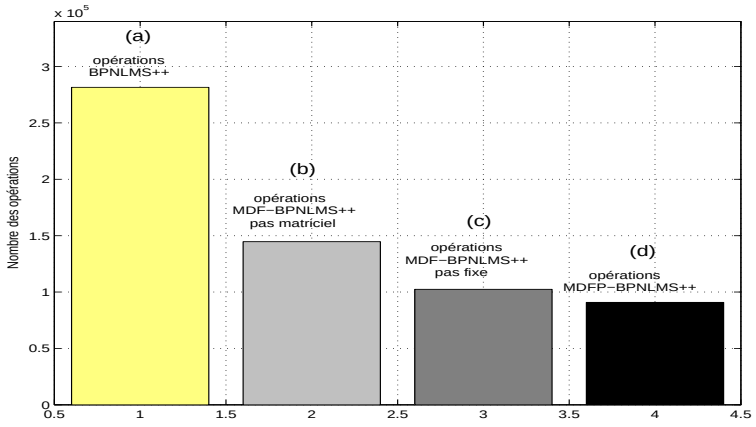

FIG. 6.5 – Nombre d'opérations simples (additions, soustractions) requises pour 10 blocs par les algorithmes : (a) : $BPNLMS++$; (b) : $MDF-BPNLMS++$ avec un pas d'adaptation matriciel ; (c) : $MDF-BPNLMS++$ avec un pas d'adaptation μ_B fixe ; (d) : $MDFP-BPNLMS++$

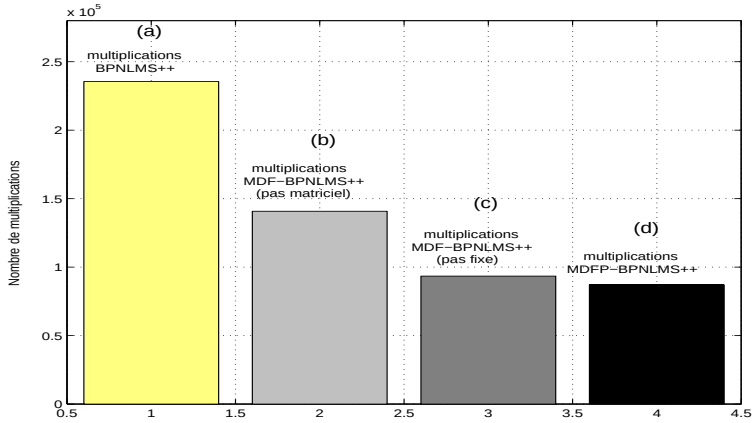

FIG. 6.6 – Nombre de multiplications requises pour 10 blocs par les algorithmes : (a) : $BPNLMS++$; (b) : $MDF-BPNLMS++$ avec un pas d'adaptation matriciel ; (c) : $MDF-BPNLMS++$ avec un pas d'adaptation μ_B fixe ; (d) : $MDFP-BPNLMS++$

Les figures (6.5) et (6.6), donnent une idée sur l'ordre de grandeur du nombre de ces opérations, montrent que l'implantation en FFT de l'algorithme $(MDFP-BPNLMS++)$ présente l'avantage d'exiger beaucoup moins d'opérations simples et de multiplications par rapport à une implantation en FFT des autres algorithmes.

Les deux avantages que présente l'algorithme $(MDFP-BPNLMS++)$, celui de la meilleure convergence des coefficients du filtre adaptatif et de la complexité réduite de calcul, associés à celui que présente la FNT nous permet de souligner l'intérêt que présenterait l'implantation en FNT de cet algorithme.

6.6 Implantation de l'algorithme MDFP-BPNLMS++ par la FNT

L'algorithme $MDFP-BPNLMS++$ que nous avons proposé précédemment a été développé pour permettre son implantation au moyen de la transformée en nombres de Fermat sur un processeur de $b = 16$ bits. Une implémentation de cet algorithme, basée sur la propriété de convolution circulaire, entraînera alors l'utilisation de transformée en nombres de Fermat sur des séquences de longueur $M' = 64$ échantillons. Toutes les opérations sont définies dans le corps de Galois modulo F_4, $GF(F_4)$, avec $F_4 = 2^b + 1 = 2^{16} + 1 = 65537$ et $\alpha = \sqrt{2}$.

6.6.1 Procédures d'implantation de l'algorithme MDFP-BPNLMS++ par la FNT

Les différentes étapes de la procédure d'implantation par la FNT de l'algorithme $MDFP-PNLMS++$, résumées par la figure (6.7), sont :

1. Fenêtrage du signal d'entrée $\{x\}$ (fenêtre de $N = 32$ échantillons).

2. Construction de $\tilde{x}_k$ par récupération des $L - 1 = 127$ échantillons du vecteur précédent, $\tilde{x}_{k-1}$.

3. Partitionnement du vecteur d'entrée $\tilde{x}_k$ en K' sous-vecteur $\tilde{x}_k^{k'}$ de longueur $N+L'-1 = 63$ échantillons, où $L' = 32$ est la longueur du filtre adaptatif $\hat{w}_k^{k'}$.

4. Construction du sous-vecteur $\tilde{X}_k^{k'}$ par ajout de $M' - (N + L' - 1)$ échantillons nuls à la fin de chaque sous-vecteur $\tilde{x}_k^{k'}$.

5. Mise à jour des coefficients $\hat{w}_k^{k'}$ du filtre adaptatif de longueur $L' = 32$. Les différentes étapes intermédiaires de la procédure de mise à jour sont décrites ci-dessous et par la figure (6.7-b) :

5.a Récupération des $L' = 32$ derniers échantillons du résultat de la FNT inverse du produit terme à terme $\tilde{\mathbf{X}}_k^{k'} \bullet \tilde{\mathbf{E}}_{-k}$ des vecteurs $\tilde{\mathbf{X}}_k^{k'}$ et $\tilde{\mathbf{E}}_{-k}$. $\tilde{\mathbf{X}}_k^{k'} = FNT\left(\tilde{X}_k^{k'}\right)$ et $\tilde{\mathbf{E}}_{-k} = FNT\left(\tilde{E}_{-k}\right)$ ($\tilde{E}_{-k}$: est le vecteur ϵ_{-k} complété par $M' - N$ coefficients nuls). Le vecteur ϵ_{-k} est calculé par application de la matrice $\overline{I}_{N\times N}$ au vecteur d'erreur ϵ_k.

5.b Mise à jour du calcul des coefficients $\hat{w}_k^{k'}$ du filtre de longueur $L' = 32$ par l'algorithme $MDFP-BPNLMS++$.

5.c Construction du vecteur $\tilde{W}_k^{k'}$ par ajout de $M' - L'$ échantillons nuls à la fin du vecteur $\hat{w}_k^{k'}$ avant d'appliquer la FNT. Le vecteur $\tilde{\mathbf{Y}}_k$ est calculé au moyen de la somme de K' produits terme à terme $\tilde{\mathbf{Y}}_k^{k'} = \tilde{\mathbf{W}}_k^{k'} \bullet \tilde{\mathbf{X}}_k^{k'}$. $\tilde{\mathbf{W}}_k^{k'} = FNT\left(\tilde{W}_k^{k'}\right)$.

6. Récupération des $N = 32$ derniers échantillons du vecteur $\tilde{Y}_k$: $\tilde{Y}_k = FNT^{-1}\left(\tilde{\mathbf{Y}}_k\right)$.

7. Construction du vecteur $\tilde{E}_{-k}$ par ajout de $M' - N = 32$ échantillons nuls à la fin du vecteur ϵ_{-k} après avoir calculé le vecteur d'erreur $\epsilon_k = \tilde{d}_k - \tilde{y}_k$.

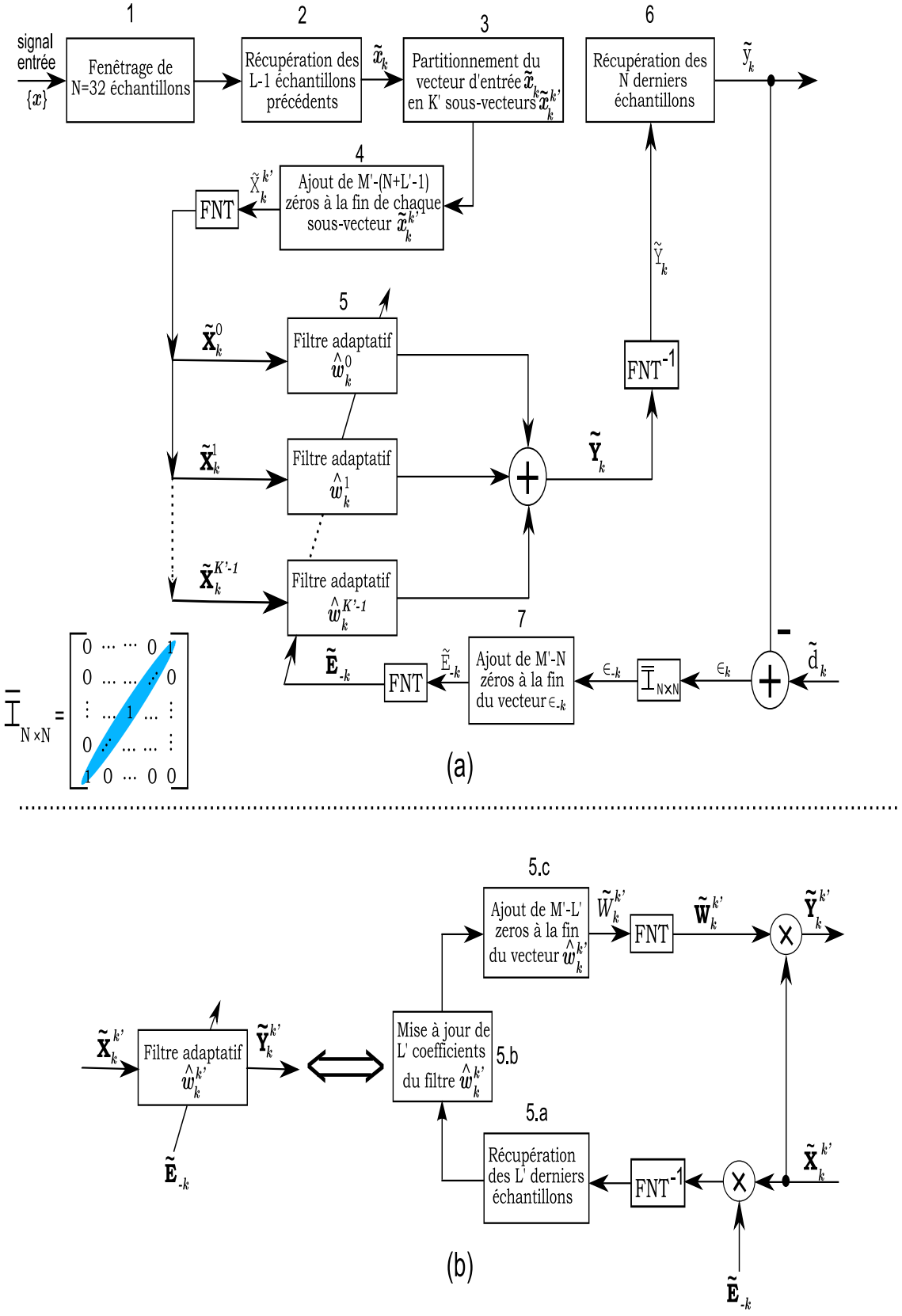

FIG. 6.7 – Système de filtrage adaptatif par MDF

6.6.2 Résultats des simulations de l'algorithme MDFP-BPNLMS++ proposé

Pour évaluer les performances de la nouvelle procédure de l'algorithme *BPNLMS++* (*MDFP−BPNLMS++*) nous avons procédé à l'implantation en *FNT* de cet algorithme que nous avons comparée à celle de l'algorithme *BPNLMS++*.

Pour l'implantation de l'algorithme *MDFP-BPNLMS++*, la transformée en nombres de Fermat a été appliquée sur des séquences de longueur $M' = 64$, avec un terme générateur $\alpha = \sqrt{2}$ et un nombre de Fermat $F_4 = 65537$. Ces mêmes paramètres sont fixés à $M = 256$ pour la longueur de séquence, $\alpha = \sqrt[8]{2}$ et $F_4 = 65537$ pour une implantation de l'algorithme *BPNLMS++*.

Cette comparaison porte sur l'atténuation de l'écho, $(ERLEC)$, fournie par l'annulateur d'écho, l'écho résiduel (ER), la puissance de l'écho résiduel et le nombre d'opérations requises par chacune des deux implantations.

La première comparaison porte sur l'atténuation de l'écho $(ERLEC)$ fourni par l'annulateur d'écho et l'écho résiduel (ER). Ces deux grandeurs sont respectivement données par les relations (6.37) et (6.38) et représentés par les figures (6.8-a) et (6.8-b).

$$ERLEC\,(k) = 10log_{10}\left[\frac{\sum_{n=(k-1)N+1}^{kN}(y_n)^2}{\sum_{n=(k-1)N+1}^{kN}(y_n-\hat{y}_n)^2}\right] \tag{6.37}$$

$$ER\,(k) = 10log_{10}\left[\sum_{n=(k-1)N+1}^{kN}(y_n-\hat{y}_n)^2\right] \tag{6.38}$$

où y_n et $\hat{y}_n$ représentent respectivement les échantillons du signal d'écho et de l'écho estimé, N le nombre d'échantillons et k l'indice du bloc.

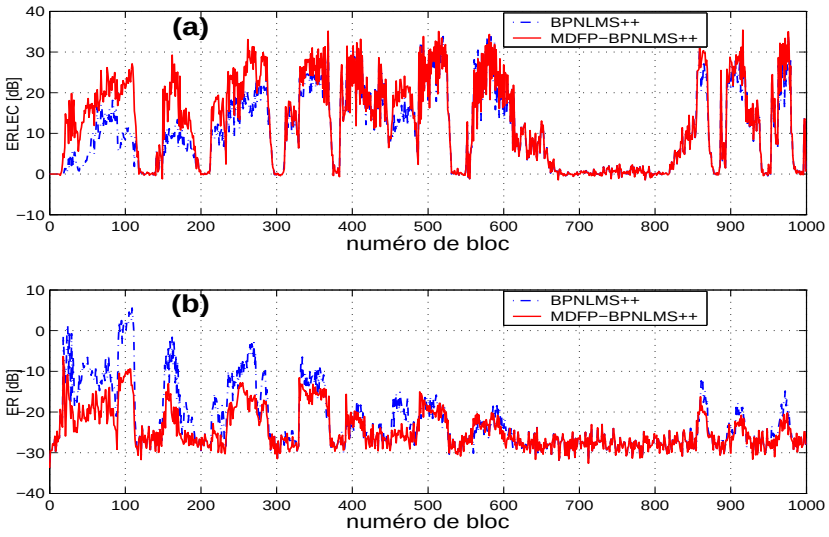

FIG. 6.8 – (a) Atténuation de l'écho (ERLEC) en dB (b) L'écho résiduel (ER) en dB

La courbe en pointillés donne les résultats de l'algorithme $BPNLMS++$ et celle tracé en ligne continue correspond aux performances de l'algorithme *MDFP-BPNLMS++*. La figure (6.8) montre que notre nouveau algorithme introduit une amélioration significative, au sens de l'atténuation d'écho, par rapport à l'algorithme *BPNLMS++*.

La deuxième comparaison concerne la puissance de l'écho et celle de l'écho résiduel obtenu à partir des deux algorithmes.

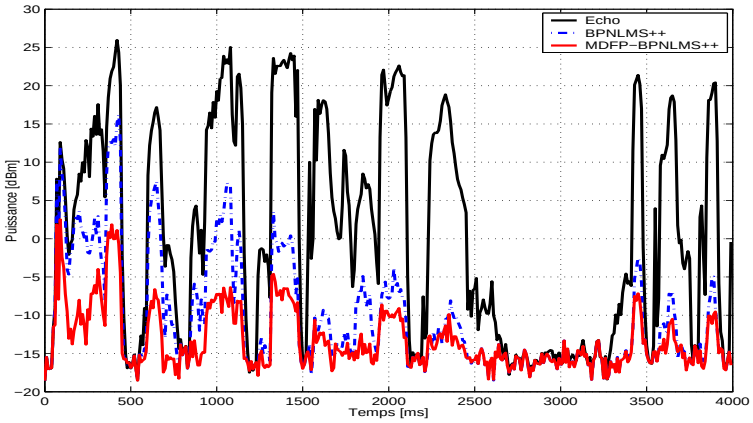

FIG. 6.9 – Puissance de l'écho (courbe gras), Puissance de l'écho résiduel par l'algorithme BPNLMS++ (courbe pointié) et par l'algorithme MDFP-BPNLMS++ (courbe rouge)

Les résultats de cette comparaison, portés sur la figure (6.9), montrent que l'algorithme *MDFP-BPNLMS++* apporte, par rapport à l'algorithme *BPNLMS++*, une meilleure atténuation de la puissance de l'écho.

L'ensemble des résultats de ces simulations montre que :

• La nouvelle méthode de normalisation du pas d'adaptation conduit à une nette amélioration au sens de rendre l'écho résiduel moins audible en sortie du système.

• Le traitement de l'algorithme du filtrage adaptatif $BPNLMS++$ par la nouvelle procédure du MDF ($MDFP - BPNLMS++$) a permis de diminuer la longueur de la transformée utilisée et par conséquence le délai de l'exécution de cet algorithme.

La troisième comparaison concerne le nombre d'opérations requises par l'implantation en FNT des deux algorithmes.

La figure (6.10) donne une idée sur l'ordre de grandeur du nombre de ces opérations.

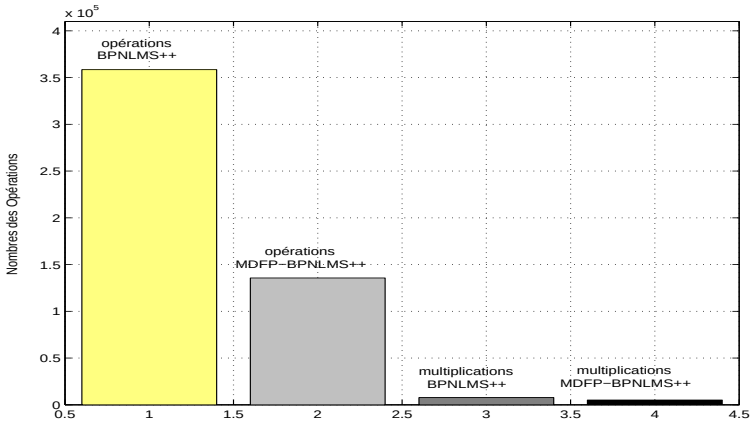

FIG. 6.10 – Opérations requises pour 10 blocs par les algorithmes $BPNLMS++$ et $MDFP-BPNLMS++$ tous deux implantés en FNT

Les résultats de cette figure montrent que l'implantation de l'algorithme *MDFP-BPNLMS++* par la FNT nécessite beaucoup moins d'opérations simples (additions, soustractions, décalages de bits) et un peu moins de multiplications par rapport à une implantation en FNT de l'algorithme $BPNLMS++$.

6.7 Conclusion

Dans le cadre du problème d'annulation d'écho acoustique, la taille des réponses impulsionnelles peut atteindre plusieurs centaines de milli-secondes, ce qui entraîne le calcul des transformées, FFT ou FNT, de plusieurs milliers de coefficients. Ce calcul est souvent inadapté aux processeurs des signaux numériques (DSP) sur lesquels sont implantés les algorithmes du filtrage adaptatif.

Nous avons donc proposé, dans un premier temps, d'utiliser un algorithme de filtrage adaptatif connu sous le nom de Filtre à Délais Multiples ($MDF : Multi-Delay\,Filter$) . Cet algorithme consiste à décomposer le filtre adaptatif initial en sous-blocs de moindre taille. Dans un deuxième temps, nous avons proposé et développé un algorithme dérivé du précédent et présentant une complexité réduite de calcul. Nous avons ensuite proposé une modification de l'algorithme $BPNLMS++$ par la nouvelle procédure $MDFP$ de l'algorithme MDF afin de minimiser la taille des transformées utilisées. En complément, nous avons proposé une méthode de gestion du pas d'adaptation, ce qui a conduit à une nette amélioration des capacités de convergence des algorithmes du filtrage adaptatif.

L'algorithme $BPNLMS++$, traité par la nouvelle procédure $MDFP$ ($MDFP$-$BPNLMS++$) et utilisant la nouvelle méthode du pas d'adaptation, a ainsi été étudié et développé pour permettre son implantation sur un processeur DSP à virgule fixe mettant en oeuvre la transformée en nombres de Fermat.

Conclusion et perspectives

L'utilisation d'un dispositif de communication mains libres est accompagnée de plusieurs problèmes critiques affectant la prise de son : la réverbération, l'écho acoustique et le bruit ambiant sont des facteurs, qui peuvent conduire à une dégradation inacceptable de la qualité de la communication. Le travail réalisé et exposé dans ce rapport fournit les premiers éléments pour évaluer la possibilité de développement d'un système d'annulation d'écho acoustique et de son intégration à un processeur DSP à virgule fixe de faible complexité.

Récapitulatif de l'étude

L'annulation d'écho acoustique est obtenue au moyen de filtres adaptatifs modélisant la réponse acoustique d'une salle en soustrayant ainsi du signal capté par le microphone, l'écho estimé . Comme il a été rappelé au chapitre 1, le filtre adaptatif nécessite une séquence d'apprentissage et une stratégie de mise à jour des coefficients de ce filtre dont l'objectif est la minimisation de la fonction coût. Cette stratégie de mise à jour est réalisée le plus souvent par des algorithmes d'optimisation du type gradient stochastique (LMS : *Least Mean Squares*, *NLMS* : *Normalized LMS*, *PNLMS* : *Proportionate NLMS* et *PNLMS++*). Dans le premier chapitre, nous avons présenté ces différents algorithmes adaptatifs, utilisés dans le système d'annulation d'écho acoustique, ainsi que leur comparaison au niveau de la convergence de leurs coefficients. Cette comparaison nous a permis de choisir, pour toute la suite de l'étude, l'algorithme *PNLMS++*, car celui-ci présente une meilleure convergence par rapport aux autres algorithmes adaptatifs.

Dans le deuxième chapitre, nous avons approfondi les bases mathématiques d'un outil qui est amené à trouver des applications de plus en plus diverses en traitement du signal. Cet outil, intitulé "transformée en nombre entiers" (NTT : *Number Theoretic Transform*) a été étudié afin d'en extraire quelques propriétés utiles au développement des algorithmes traités. A l'inverse de la transformée de Fourier, la transformée en nombres entiers effectue tous les calculs sans erreur d'arrondi en évitant le passage en nombres complexes des données. Nous avons introduit plus particulièrement la transformée en nombres de Fermat (FNT) car elle permet le calcul rapide des produits de convolution. Ainsi, son application aux algorithmes de filtrage

adaptatif laisse entrevoir une réduction importante de la complexité de calcul dans le système d'annulation d'écho acoustique.

Pour mettre en évidence l'intérêt de la transformée en nombres de Fermat, nous avons proposé de traiter, dans le troisième chapitre, les algorithmes adaptatifs (*NLMS*, *PNLMS*, *PNLMS++*) par blocs d'échantillons au lieu de les traiter échantillon par échantillon. Ce processus fait intervenir une série de produits de convolution qui peuvent être réalisés par la mise en oeuvre de la transformée de Fourier rapide (*FFT*) ou de la transformée en nombres de Fermat. Nous avons comparé les performances de ces différents algorithmes proposés (*BNLMS*, *BPNLMS*, *BPNLMS++*) afin de déterminer l'algorithme qui minimise l'écho résiduel. Les résultats de cette comparaison ont permis de conclure que c'est l'algorithme $BPNLMS++$ qui minimise l'écho résiduel.

Cette conclusion nous a conduit à proposer un développement de l'algorithme $BPNLMS++$ pour permettre son implantation au moyen de la FNT. Cette implantation par la FNT a permis une réduction significative du nombre de multiplications par rapport à une implantation par la FFT. En revanche, elle fait augmenter le nombre d'opérations telles que les additions, les soustractions et les décalages de bits. Pour réduire cette augmentation du nombre d'opérations, nous avons proposé d'utiliser, dans le quatrième chapitre, une technique intitulée "Sliding Généralisé" dans le calcul de la *FFT*. Cette technique (*GSFFT*) consiste à calculer la transformée de Fourier rapide d'une succession de séquences dans laquelle deux séquences successives possèdent des échantillons communs et a pour objectif de réduire la complexité de calcul d'une FFT. Le résultat des simulations d'implantation de l'algorithme $BPNLMS++$ au moyen de cette technique montre que celle-ci a permis de pallier à l'augmentation du nombre d'opérations classiques observée lors d'une implantation en FNT. Pour associer ce résultat intéressant de la $GSFFT$ à celui obtenu précédement au moyen de la FNT, nous avons proposé et développé une technique analogue appliquée à la FNT ($GSFNT$). La méthode d'implantation de cette dernière, présentant l'avantage de nécessiter moins d'opérations classiques (additions, soustractions, décalages de bits) qu'une implantation en FNT, a été utilisée pour implanter l'algorithme $BPNLMS++$ avec une faible complexité.

Cette implantation a pu fournir de bons résultats au niveau de l'annulation d'écho acoustique lorsqu'une seule voix est émise. Cependant si deux personnes parlent simultanément, l'algorithme d'adaptation ne va plus tendre vers une modélisation du chemin d'écho mais il aura tendance à diverger. Dans l'optique de réduire cette divergence du filtre adaptatif, nous avons étudié un système combiné qui consiste à associer un annulateur d'écho acoustique à un post-filtre. Ce dernier, placé juste après l'annulateur d'écho, permet de garantir une atténuation acceptable de l'écho audible sans engendrer de dégradation sur le signal de la parole locale.

L'analyse théorique d'un tel système nous a conduit à proposer son amélioration en adaptant les coefficients du filtre adaptatif par des algorithmes plus robustes que nous avons traités par blocs ($RBNLMS$,

$RBPNLMS$, $RBPNLMS++$) avant de les implanter en $GSFNT$. Ces algorithmes, dont l'intérêt est de limiter le problème de divergence du filtre adaptatif dans le cas de double parole ont été comparés, en terme de convergence des coefficients du filtre adaptatif, aux algorithmes non robustes. Les résultats de cette comparaison montrent que ces algorithmes présentent l'avantage d'empêcher, pendant la phase de double parole, la divergence du filtre adaptatif observée avec les algorithmes non robustes traités par blocs. Ces résultats, montrant aussi que l'algorithme $RBPNLMS++$ présente une meilleure convergence que les autres algorithmes adaptatifs, nous ont confortés dans le choix du système combiné que nous avons proposé. Ce système combiné a donc été développé au moyen de l'algorithme $RBPNLMS++$ associé au post-filtre avant d'être implanté en $GSFNT$ pour réduire au maximum sa complexité de calcul.

Dans le cadre du problème d'annulation d'écho acoustique, la taille des réponses impulsionnelles peut atteindre plusieurs centaines de milli-secondes, ce qui entraîne un calcul des transformées, FFT ou FNT, de plusieurs milliers de coefficients. Ce calcul, souvent inadapté aux processeurs des signaux numériques (DSP) sur lesquels sont implémentés les algorithmes du filtrage adaptatif, peut être réalisé en utilisant un algorithme de filtrage adaptatif, connu sous le nom de Filtre à Délais Multiples ($MDF : Multi-Delay\,Filter$), et qui consiste à décomposer le filtre adaptatif initial en sous-blocs de moindre taille. Après avoir décrit cet algorithme dans le sixième chapitre, nous avons proposé et développé un nouvel algorithme ($MDFP$), dérivé du précédent et présentant une complexité réduite de calcul. Nous avons ensuite proposé une nouvelle version de l'algorithme $BPNLMS++$ obtenue en traitant celui-ci par la nouvelle procédure $MDFP$ de l'algorithme MDF. Cette nouvelle version ($MDFP-BPNLMS++$) a été traitée en introduisant une nouvelle gestion du pas d'adaptation avant de comparer ses performances à celles de l'algorithme $BPNLMS++$. Les résultats de comparaison de ces deux algorithmes, implantés en FNT ont montré que :

• La nouvelle méthode de normalisation du pas d'adaptation conduit à une nette amélioration des capacités de convergence et rend ainsi l'écho résiduel moins audible en sortie du système.

• L'algorithme ($MDFP-BPNLMS++$) traité par la nouvelle gestion du pas d'adaptation a permis de minimiser la taille de la transformée utilisée et par conséquent de diminuer le délai de l'exécution de l'algorithme.

• L'implantation de l'algorithme ($MDFP-BPNLMS++$) par la FNT a permis une réduction significative de la complexité de calcul par rapport à une implantation en FNT de l'algorithme $BPNLMS++$.

La dernière partie de notre étude a été d'effectuer des tests d'écoute pour vérifier le bon fonctionnement du système d'annulation d'écho acoustique.

L'algorithme $BPNLMS++$ proposé a été testé dans les cas de simple et de double parole. Dans le cas de simple parole, ces tests ont permis de vérifier le bon fonctionnement du système d'annulation d'écho en terme de rendre l'écho moins audible. Dans le cas de double parole, ces tests ont montré une dégradation sur le signal de la parole locale. Pour remédier à cette dégradation, nous avons réalisé les tests avec le système

combiné proposé ($RBPNLMS++$ et post-filtre). Les résultats de ces tests ont permis de montrer que le système combiné proposé rend l'écho moins audible sans engendrer une dégradation sur le signal de parole locale.

Perspectives du travail

La première perspective concerne l'application des méthodes développées dans notre étude afin de les utiliser dans le système d'annulation d'écho acoustique pour la téléphonie IP ($Internet\ Protocol$). En effet, le développement de la technologie IP devrait permettre dans un avenir proche la mise en oeuvre d'une téléphonie sur réseau Internet, plus complémentaire et plus concurrentiel qu'un réseau téléphonique classique par commutation ($PSTN$: $Public\ Switched\ Telephone\ Network$). Les caractéristiques de la téléphonie IP telles que la possibilité d'une grande compression du signal de la parole ou l'utilisation optimale du réseau grâce à la transmission de l'information par paquets, en font une application ambitieuse. Le système combiné, que nous avons considéré comme un système complet pour l'annulation d'écho acoustique, sera difficile à mettre en oeuvre dans le cas d'une application en téléphonie IP, dans la mesure où l'écho représente un retard de l'ordre de 64 millisecondes. En effet, pour ce retard, l'application de la transformée en nombres entiers nécessite le traitement de séquences d'un millier d'échantillons et donc de filtres de tailles inadaptées aux processeurs de traitement du signal en virgule fixe. Pour résoudre ce problème, il serait intéressant d'utiliser la technique du MDF associée à celle de la gestion du pas d'adaptation pour le traitement de l'algorithme $RBPNLMS++$ du système combiné.

La seconde perspective du travail qu'il serait intéressant d'aborder par la suite, se situe au niveau de l'implémentation du système d'annulation d'écho acoustique sur une carte $FPGA$ en langage $VHDL$. Le principe de cette implémentation en virgule fixe mettant en oeuvre la FNT, sur une carte $FPGA$, est basé sur un circuit en chaîne "Pipeline". L'intérêt d'un tel circuit est la diminution non négligeable de la surface matérielle utilisée lors de l'implémentation. Ce circuit permet également de travailler en temps réel avec des durées réduites de calcul. La figure (7.1) donne une idée sur l'architecture FNT en pipeline pour une séquence $\{x\}$ de longueur M.

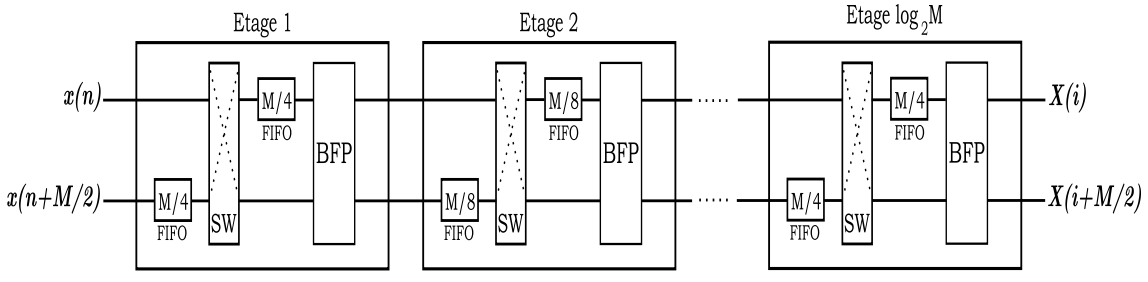

FIG. 7.1 - Architecture de la FNT en pipeline pour une séquence de longueur M

Le nombre de processeurs Butterflies nécessaires pour cet implémentation est $log_2 M$. Chaque processeur Butterfly est constitué de :

• Deux registres à décalages de type : "First In First Out : FIFO", dont la longueur dépend de l'étage et du choix de M.

• Un élément de routage (switch : SW) qui, sous l'action d'une commande, va envoyer les données en ligne ou bien les croiser.

• Une structure de type Butterfly (BFP) dont le principe d'implémentation en virgule fixe modulo un nombre de Fermat F_t est donnée par la figure (7.2).

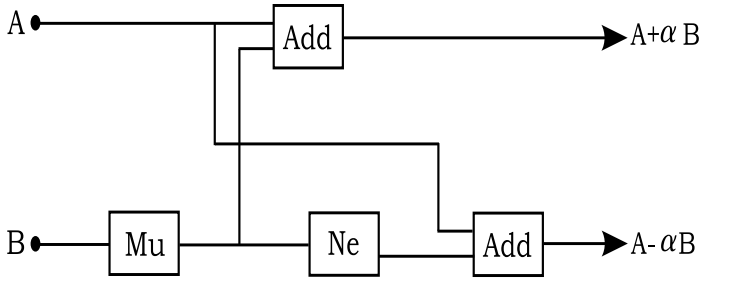

FIG. 7.2 - Processeur Butterfly

L'implémentation des fonctions réalisées modulo F_t (multiplication, négation, addition) est basée sur des opérateurs logiques de base tels que : ET, OU, NON, NON-ET et OU Exclusif.

Annexe A

Corps de Galois

Par définition, un corps de Galois $GF\ (q)$ d'ordre égal à un entier q est constitué de q éléments.

- Si q est premier, le corps de Galois $GF\ (q)$ est constitué de q éléments de l'ensemble $\left\{\ 0,\quad \ldots,\quad q-1\ \right\}$. $GF\ (q)$ sera dit primitif.

- Si q n'est pas premier, il existe un nombre p tel que $q = p^n$ où p est premier et n entier. $GF\ (p^n)$ est alors appelé extension du corps $GF\ (p)$.

Tous les éléments d'un corps de Galois $GF\ (q)$ peuvent être additionnés, soustraits, multipliés et divisés ; les opérations s'effectuant modulo q.

• Le corps est fini, si la somme ou le produit de n'importe quels éléments de $GF\ (q)$ est aussi un élément du corps de Galois $GF\ (q)$.

• Le corps contient un unique élément neutre pour l'addition et pour la multiplication, de telle sorte que n'importe quel élément k de $GF\ (q)$ vérifie les deux égalités suivantes :

$$\begin{cases} k+0=k \\ k\times 1=k \end{cases} \tag{A.1}$$

• Pour tout élément k de $GF\ (q)$, tel que $k \neq 0$, il existe un élément opposé $-k$ tel que :

$$k+(-k)=0 \tag{A.2}$$

Ainsi l'opération de soustraction d'un élément $l \in GF\ (q)$, est définie par :

$$k-l=k+(-l)\ \text{ où }\ (-l)\ \text{ est l'opposé de } l \tag{A.3}$$

• Pour tout élément k de $GF\ (q)$, tel que $k \neq 0$, il existe un inverse unique k^{-1}, tel que :

$$k.k^{-1} = 1 \tag{A.4}$$

Ainsi l'opération de division, par un élément $l \in GF\ (q)$ avec $l \neq 0$, est définie par :

$$\frac{k}{l} = k.l^{-1} \text{ où } l^{-1} \text{ est l'inverse de } l \tag{A.5}$$

• Dans un corps de Galois $(GF\ (q)\ ,\ +, \times)$, les propriétés d'associativité, de commutativité et de distributivité sont vérifiées.

Annexe B

Algorithme d'Euclide étendu

L'application de l'algorithme d'Euclide étendu au calcul d'un inverse modulo q utilise l'analogie existant entre un polynôme $a(z) = \sum_{i=0}^{n} a_i z^i$ et un nombre binaire $a = \sum_{i=0}^{n} a_i 2^i$ $(a_i = 0 \text{ où } 1,\ z = 2)$.

Etant donnés deux polynômes $a(z)$ et $b(z)$, respectivement d'ordre n et m, $m \leq n$.

$$\begin{cases} a(z) = \sum_{i=0}^{n} a_i z^i, & a_0 \neq 0 \\ b(z) = \sum_{i=0}^{m} b_i z^i, & m \leq n \end{cases} \tag{B.1}$$

L'algorithme d'Euclide permet de trouver l'unique quotient $k(z)$ et l'unique reste $r(z)$ tels que :

$$a(z) = k(z)\, b(z) + r(z), \ \deg(r(z)) < \deg(b(z)) \tag{B.2}$$

Ces polynômes peuvent ainsi être obtenus en procédant à une série de divisions successives :

$$\begin{cases} a(z) = k_1(z)\, b(z) + r_1(z) & \deg(r_1) < \deg(b) \\ b(z) = k_2(z)\, r_1(z) + r_2(z) & \deg(r_2) < \deg(r_1) \\ \dots & \dots \\ r_{p-3}(z) = k_{p-1}(z)\, r_{p-2}(z) + r_{p-1}(z) & \deg(r_{p-1}) < \deg(r_{p-2}) \\ r_{p-2}(z) = k_p(z)\, r_{p-1}(z) + 0 & \end{cases}$$

L'algorithme s'arrête lorsque $r_p(z) = 0$. Le Plus Grand Diviseur Commun ($PGCD$) de $a(z)$ et $b(z)$ est donné par le dernier reste non nul $r_{p-1}(z)$. Il existe alors deux polynômes uniques $x(z)$ et $y(z)$ tels que :

$$r_{p-1}(z) = x(z)\, a(z) + y(z)\, b(z) \tag{B.3}$$

Notons que cet algorithme peut être présenté sous la forme de divisions successives $\frac{S_{i-2}(z)}{S_{i-1}(z)}$ pour obtenir

les restes $S_i = rem\left(\begin{array}{cc} S_{i-2}, & S_{i-1} \end{array} \right)$, avec l'initialisation $S_0(z) = a(z)$ et $S_1(z) = b(z)$.

Une version étendue de l'algorithme d'Euclide permet de trouver le couple $\left\{ \begin{array}{cc} x(z), & y(z) \end{array} \right\}$. Dans le même temps que le calcul du $PGCD$ de $a(z)$ et $b(z)$, les polynômes $x(z)$ et $y(z)$ sont calculés par récursivité tant que $S_i \neq 0$:

$$\begin{cases} Q_i = ent\left(\frac{S_{i-2}}{S_{i-1}}\right) \\ x_i = x_{i-2} - Q_i x_{i-1} \\ y_i = y_{i-2} - Q_i y_{i-1} \end{cases}$$

où les conditions initiales sont données par $x_0 = y_1 = 1$, $x_1 = y_0 = 0$ et $i = 2$. Nous pouvons en déduire que les polynômes $a(z)$ et $b(z)$ sont premiers entre eux si et seulement si il existe deux polynômes $x(z)$ et $y(z)$ tels que (Identité de Bezout) :

$$x(z)\,a(z) + y(z)\,b(z) = 1 \tag{B.4}$$

avec $\deg(x(z)) < m$ et $\deg(y(z)) < n$.

Annexe C

Facteur d'échelle s

Le facteur d'échelle s, est un facteur variable qui diminue l'effet de double parole et donc de divergence du filtre adaptatif. Ce facteur peut être calculé à partir d'une fonction implicite J_k définie par : [Huber1981]

$$J_k = \sum_{n=0}^{k} \lambda_1^{k-n} \mathcal{F}\left(\frac{|e_n|}{s}\right) \tag{C.1}$$

où $0 < \lambda_1 < 1$.

La fonction $\mathcal{F}(\cdot)$ est définie par :

$$\mathcal{F}(\cdot) = \psi(\cdot) - \lambda' \tag{C.2}$$

où $\psi(\cdot)$ est une fonction limiteur donnée par :

$$\psi\left(\frac{|e_k|}{s_{k-1}}\right) = min\left\{\frac{|e_k|}{s_{k-1}}, \kappa_0\right\} \tag{C.3}$$

et λ' est une variable gaussienne définie par :

$$\lambda' = \frac{2}{\sqrt{2\pi}} \int_0^{\infty} \psi(z)\, e^{-\left(\frac{1}{2}\right)z^2} dz$$

$$= \sqrt{\frac{2}{\pi}} \left(1 - e^{-\left(\frac{1}{2}\right)\kappa_0^2}\right) + \kappa_0 \mathrm{erfc}\left(\frac{\kappa_0}{\sqrt{2}}\right) \tag{C.4}$$

où κ_0 est une constante et la fonction $\mathrm{erfc}(x)$ est définie par :

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt \tag{C.5}$$

Une relation de récurrence du facteur d'échelle s est donnée, selon le principe du gradient stochastique,

par :

$$s_k = s_{k-1} - \left[\nabla_s \left(J_k\right)\right]^{-1} J_k \tag{C.6}$$

Le gradient ∇_s de la fonction J_k est donné par :

$$\nabla_s \left(J_k\right) = \sum_{n=0}^{k} -\lambda_1^{k-n} \frac{|e_n|}{s^2} \mathcal{F}' \left(\frac{|e_n|}{s}\right) \tag{C.7}$$

Ce qui peut s'écrire :

$$\nabla_s \left(J_k\right) = -\frac{1}{s} b_k = -\frac{1}{s}\left[\frac{|e_k|}{s} \mathcal{F}' \left(\frac{|e_k|}{s}\right) + \lambda_1 b_{k-1}\right] \tag{C.8}$$

avec :

$$b_{k-1} = \lambda_1^{k-1} \frac{|e_0|}{s} \mathcal{F}' \left(\frac{|e_0|}{s}\right) + \ldots + \frac{|e_{k-1}|}{s} \mathcal{F}' \left(\frac{|e_{k-1}|}{s}\right)$$

En utilisant les équations (C.6) et (C.8) et en faisant les approximations $J_k \simeq \mathcal{F}\left(\frac{|e_k|}{s}\right)$, $\nabla_s \left(J_k\right) \simeq \nabla_s \left(J_{k-1}\right)$, la relation (C.6) devient :

$$s_k = s_{k-1} + \frac{s_{k-1}}{b_{k-1}} \mathcal{F}\left(\frac{|e_k|}{s_{k-1}}\right) \tag{C.9}$$

où b_k est défini par :

$$b_k = \lambda_1 b_{k-1} + \frac{|e_k|}{s_{k-1}} \mathcal{F}' \left(\frac{|e_k|}{s_{k-1}}\right)$$

L'expression de s_k, donnée par la relation (C.9), montre la complexité de calcul de ce facteur. Cependant, dans le cas où $\nabla_s \left(J_k\right)$ est stationnaire et s_k est un processus gaussien, ce dernier est donné par une relation de récurrence simplifiée. En effet :

$$E\left\{\nabla_s \left(J_k\right)\right\} = \sum_{n=0}^{k} -\lambda_1^{k-n} E\left\{\frac{|e_n|}{s^2} \mathcal{F}' \left(\frac{|e_n|}{s}\right)\right\}$$

$$= \sum_{n=0}^{k} -\lambda_1^{k-n} \frac{1}{s} E\left\{\frac{|e_n|}{s} \mathcal{F}' \left(\frac{|e_n|}{s}\right)\right\} \tag{C.10}$$

$$= \sum_{n=0}^{k} -\lambda_1^{k-n} \frac{1}{s} \frac{2}{\sqrt{2\pi}} \int_0^{\kappa_0} z e^{-\left(\frac{1}{2}\right)z^2} dz \tag{C.11}$$

$$= \sum_{n=0}^{k} -\lambda_1^{k-n} \frac{1}{s} \sqrt{\frac{2}{\pi}} \left(1 - e^{-\left(\frac{1}{2}\right)^{\kappa_0^2}}\right)$$

Ce dernier terme tend vers $-\frac{1}{s}\left(\frac{\gamma_1}{1-\lambda_1}\right)$ quand $n \longrightarrow +\infty$.

$$\text{avec } \gamma_1 = \sqrt{\frac{2}{\pi}} \left(1 - e^{-\left(\frac{1}{2}\right)^{\kappa_0^2}}\right) \tag{C.12}$$

L'ensemble des équations (C.2), (C.6) permet d'obtenir la relation de récurrence simplifiée de s_k :

$$s_k = s_{k-1} + \frac{(1-\lambda_1)\, s_{k-1}}{\gamma_1} \left\{ \psi\left(\frac{|e_k|}{s_{k-1}}\right) - \lambda' \right\} \tag{C.13}$$

$$= \lambda s_{k-1} + \frac{1-\lambda}{\lambda'} s_{k-1} \psi\left(\frac{|e_k|}{s_{k-1}}\right) \tag{C.14}$$

où $\lambda = 1 - \frac{\lambda'}{\gamma_1}(1-\lambda_1)$.

Bibliographie

[Agarwal1974] R.C. Agarwal, C.S. Burrus, "Fast convolution using Fermat number transform with application to digital filtering", IEEE trans. on Acoustics, Speech and Signal Processing, vol. ASSP-22, **N**°2, pp.87-97, April 1974.

[Agarwal1975] R.C. Agarwal, C.S. Burrus, "Number Theoretic Transforms to implement Fast Digital Convolution", IEEE Proceedings, vol. 63, **N**°4, pp.550-560, April 1975.

[Aho1974] A.V. Aho, J.E. Hopcroft, J.D. Ullman, "The design and analysis of computer algorithms", Addison-Wesley Series in Computer Science and Information Processing, 1974.

[Alaeddine2006] H. Alaeddine, E-H. Baghious, G. Madre, G. Burel, "Realization of Block Robust Adaptive Filters using Generalized Sliding Fermat Number Transform", 14th European Signal Processing Conference (EUSIPCO), Florence (Italy), September 2006.

[Alfredsson1996] L-I. Alfredsson, "VLSI Architecture and Arithmetic Operations with Application to the Fermat Number Transform", Linköping University, Sweden, 1996.

[Amrane1992] Ait Amrane O., "Identification de systèmes à réponse impulsionnelle longue par filtrage adaptatif en fréquence : application à l'annulation d'écho acoustique", Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, Paris, France, 1992.

[Baghious2004] E.H. Baghious, G. Madre, H.Alaeddine, G. Burel, "Realization of adaptive blocks digital filters using Fermat Number Transforms", International Symposium on Image/Video Communications over fixed and mobile networks (ISIVC'04), Brest, France, July 2004.

[Benesty1991] Benesty J., "Algorithmes du type gardient à complexité de calcul réduite et à capacité de poursuite et de vitesse de convergence accrues. Application à l'annulation de l'écho acoustique", Note technique CRPE/193, 1991.

[Benesty2000] J. Benesty, D. R. Morgan, J. H. Cho, "A new class of Double talk Detectors based on Cross-correlation", IEEE Transactions on Speech and Audio Processing, vol. 8, **N**°2, March 2000.

[Benesty2001] J. Benesty, T. Gansler, D.R. Morgan, M.M. Sondhi, S.L. Gay, "Advances in Network and Acoustic Echo Cancellation", Springer-Verlag, 2001.

[Bernard1975] Bernard Widrow, John R. Glover Jr., John M. McCool, John Kaunitz, Charles S. Williams, Robert H. Hearn, James R. Zeidler, Eugene Dong Jr., Robert C. Goodlin, "Adaptive Noise Cancelling : Principles and Applications", Proceedings of the IEEE, vol. 63, **N°12**, pp. 1692-1716, December 1975.

[Bernard1976] Bernard Widrow, John M. McCool, Michael G. Larimore, C. Richard Johnson Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter", Proceedings of the IEEE, vol. 64, **N°8**, pp. 1151-1162, August 1976.

[Bernard1985] Bernard Widrow, Samuel D. Stearns, "Adaptive Signal Processing", Englewood Cliffs, N.J., Prentice-Hall, 1985, ISBN 0130040290.

[Berouti1979] Berouti M., Schwartz R., Makhoul J., "Enhancement of speech corrupted by acoustic noise", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'79, pp. 208-211, April 1979.

[Blahut1985] R. Blahut, "Fast algorithms for digital signal processing", Addison-Wesley Publishing Company, 1985.

[Boll1979] Boll S. F., "Suppression of acoustic noise in speech using spectral substraction", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 27, **N°**2, pp. 113-120, April 1979.

[Burrus1971] C. Burrus, "Block implementation of digital filters", IEEE Trans. Circuit Theory, vol. CT-18, **N°**6, November 1971.

[Cappé1993] Cappé O., "Techniques de réduction de bruit pour la restauration d'enregistrements musicaux", Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, **N°**93 E019, September 1993.

[Chevillat1978] P. R. Chevillat, "Transform-Domain Digital Filtering with Number Theoretic Transforms and Limited Word Lengths", IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP-26, **N°**4, pp. 284-290, August 1978.

[Cho1999] J. H. Cho, D. R. Morgan, J. Benesty, "An objective technique for evaluating Double-talk Detectors in acoustic echo cancelers", IEEE Transactions on Speech and Audio Processing, vol. 7, **N°**6, November 1999.

[Clark1981] G.A. Clark, S.K. Mitra, and S.R. Parker, "Block implementation of adaptive digital filters", IEEE Trans. on Circuits and Systems, vol. CAS-28, pp. 585-592, June 1981.

[Clark1983] G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters", IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-31, pp. **N°**5, 1073-1083, October 1983.

[Compernolle1992] Compernolle D. V., "DSP techniques for speech enhancement", Workshop on Speech Processing in Adverse Conditions, pp. 21-30, November 1992.

[Cooley1965] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Mathematics Computation, vol. 19. pp. 297-301, April 1965.

[David1984] David G. Messerschmitt, "Echo Cancellation in Speech and Data Transmission", IEEE Journal on Selected Areas in Communications, vol. SAC-2, **N°2**, pp. 283-297, March 1984.

[Duhamel1982] P. Duhamel, H. Hollmann, "Number Theoretic Transforms with 2 as a root of unity", Electronics letters, vol. 18, **N°**22, pp. 978-980, 28 October 1982.

[Duttweiler2000] D.L. Duttweiler, "Proportionate normalized least mean square adaptation in echo cancelers", IEEE Trans. Speech Audio Processing, vol. 8, **N°5**, pp.508-518, September 2000.

[Emmanuel1993] Emmanuel C. Ifeachor, Barrie W. Jervis, "Digital Signal Processing : A Pratical Approach", Addison-Wesley, 1993, ISBN 020154413X.

[Ephraim1983] Ephraim Y., Malah D., "Speech enhancement using optimal nonlinear spectrale amplitude estimation", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83, vol. 8, pp.1118-1121, Boston, USA, 1983.

[Ephraim1984] Ephraim Y., Malah D., "Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. 32 (6), pp.1109-1121, 1984.

[Farhang1992] B. Farhang-Boroujeny and Y. C. Lim, "A comment on the computational complexity of sliding FFT", IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing, vol. 39, **N°12**, December 1992.

[Farhang1994] B. Farhang-Boroujeny and S. Gazor, "Generalized Sliding FFT and its Application to Implementation of Block LMS adaptive filters", IEEE Trans. Signal Processing, vol. 42, pp. 532-538, March 1994.

[Farhang1998] B. Farhang-Boroujeny, "Adaptive Filters : Theory and Apllications", Wiley & Sons, ISBN 978-0-471-98337-8, October 1998.

[Ferrara1980] E.R. Ferrara, "Fast implementation of LMS adaptive filters", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-28, pp. 474-475, 1980.

[Feur1985] A. Feur, E. Weinstein, "Convergence Analysis of LMS Filters with Uncorrelated Gaussian Data", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33, pp. 222-230, February 1985.

[Gänsler1998] T. Gänsler, "A double-talk resistant subband echo canceller", Signal Processing, vol. 65, **N°1**, pp. 89-101, 1998.

[Gänsler2000] T. Gänsler, S.L. Gay, M.M. Sondhi, and J. Benesty, "Double-talk Robust Fast Converging Algorithms for Network Echo Cancellation", IEEE Transactions on Speech and Audio Processing, Vol. 8, **N°**6, pp. 656-663, November 2000.

[Gay1998] S.L, Gay, "An efficient fast converging adaptive filter for network echo cancellation", Asilomar Conference on Signal, Systems and Computers, USA, November 1998.

[Gazor1992] S. Gazor and B. Farhang-Boroujeny, "A state space approach for efficient implementation of block LMS adaptive filters", Proc. Singapore Int. Conf. Commun. Syst. ICCS/ISITA'92, (Singapore), pp. 808-812, November 1992.

[George1999] George-Othon Glentis, Kostas Berberidis, Sergios Theodoridis, "Efficient Least Squares Adaptive Algorithms for FIR Transversal Filtering", IEEE Signal Processing Magazine, pp. 13-41, July 1999.

[Gersho1983] A. Gersho, "Adaptive equalization of highly dispersive channels for data transmission", Bell Syst. Technical Journal., vol. 48, **N°1**, pp.55-70, Jan 1983.

[Gilloire1987] Gilloire A., Julien J.P., "L'acoustique des salles dans les télécommunications", L'écho des recherches, **N°127**, pp. 43-54, 1987.

[Gilloire1994] Gilloire A., Vetterli M., "Performance evaluation of acoustic echo controls : required values and measurement procedures", Annales des Télécommunications, vol 49, **N°7 – 8**, pp. 368-372, 1994.

[Gnanasekaran1977] R. Gnanasekaran and S. K. Mitra, "A note on block implementation of IIR digital filters", Proc. IEEE, vol. 65, pp. 1063-1064, July 1977.

[Gritton1984] C.W.K. Gritton, D.W. Lin, "Echo Cancellation Algorithms", IEEE Acoutics, Speech and Signal Processing Magazine, pp. 30-38, April 1984.

[Haykin1986] Haykin S., "Adaptive filter theory", Prentice-Hall, Englewood Cliffs, N.J. 07632.

[Huber1981] P. J. Huber, Robust Statistics. New York : Wiley, 1981, pp. 68-71, 109, 135-138.

[IUTG131] Recommandation UIT-T G.131 (08/96), Réduction de l'écho pour le locuteur.

[IUTG729] ITU-T Recommendation. G.729, "Coding of Speech at 8 kbits/s using Conjugate Algebraic Code-Excited Linear Prediction (CS-ACELP)", June 1995.

[Julien1982] J.P.Julien, “Acoustique des salles, prise et restitution du son, analysées à partir de la notion de canal acoustique”, Note technique CNET NT/LAA/TSS/128, Août 1982.

[Julien1984] J.P.Julien, A. Gilloire et A. Saliou, “Mesure de réponses impulsionnelles en acoustique”, Note technique CNET NT/LAA/TSS/181, Juillet 1984.

[Julien1991] G.A. Julien, “Number Theoretic Techniques in Digital Signal Processing,” Academic Press, vol. 80, Chap. 2,pp. 69-163, 1991.

[Kazuo1990] Kazuo Murano, Shigeyuki Unagami, Fumio Amano, “Echo Cancellation and Applications”, IEEE Communications Magazine, pp.49-55, Jan 1990.

[Knuth1969] D. E. Knuth, “The Art of Computer Programming”, Vol. 2 : Seminumerical Algorithms, Addison-Wesley, 1969.

[Kurt1976] Kurt H. Mueller, “A New Digital Echo Canceler for Two-wire full-Duplex Data Transmission”, IEEE Transactions on Communications, vol. 24, **N°9**, pp. 956-962, September 1976.

[Kuttruff1991] Kuttruff H., “Rooms acoustics”, Elsevier Applied Sciences, 1991.

[Lee1983] J. C. Lee, “A Class of Adaptive Digital Filters and Their Applications”, Ph. D. dissertation, Dep. Elec. Eng., Korea Advanced Inst. Sci. Technol., Seoul, June 1983.

[Lee1985] J. C. Lee, B.K. Min, and M. Suk, “Realization of Adaptive Digital Filters Using the Fermat Number Transform”, IEEE Trans on Acoust, Speech and Signal Processing, vol. ASSP-33, **N°3**, pp. 1036-1039, Aug 1985.

[Leibowitz1976] L. M. Leibowitz, “A Simplified Binairy Arithmetic for the Fermat Number Transform”, IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-24, **N°5**, pp. 356-359, October 1976.

[Lim1979] Lim J. S., Oppenheim A. V., “Enhancement and bandwidth compression of noisy speech”, Proceedings of the IEEE, Vol. 67, **N°**12, pp. 1586-1604, December 1979.

[Ljung1983] L.Ljung and T.Söderström, “Theory and Practice of Recursive Identification”, M.I.T.Press. October 1983.

[Lockwood1991] Lockwood P., Boudy J., “Experiments with a non-linear spectral subtractor, hidden Markov models and the projection, for robust speech recongnition in cars”, Peoceedings of EUSIPCO’91, pp. 79-82, 1991.

[Macchi1998] Macchi O., Bellanger M., “Le filtrage adaptatif transverse”, Traitement du signal, vol.5, pp. 115-132, 1988.

[Martin1995] Martin R., Altenhoner J., "Coupled adaptive filters for acoustic echo control and noise reduction", Proceedings of ICASSP'95, pp. 3043-3046, Detroit, MI, USA, May 1995.

[McClellan1976] J. H. McClellan, "Hardware Realization of a Fermat Number Transform", IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-24, **N°3**, PP. 216-225, June 1976.

[Moorer1986] Moorer J. A., Berger M., "Linear-phase bandsplitting : Theory and applications", Journal of the Audio Engineering Society, vol. 34 (3), pp. 143-153, 1986.

[Moulines1995] E. Moulines, O. Amrane, Y. Grenier, "The Generalized Multidelay Adaptive Filter : Structure and Convergence Analysis", IEEE Transactions on Signal Processing, vol. 43, **N°1**, pp. 14-28, January 1995.

[Odile1995] Odile Macchi, "Adaptive Processing the Least Mean Squares Approach with Applications in Transmission", Wiely & Sons, 1995.

[Pieter1974] Pieter Eykhoff, "System Identification", Wiley & Sons, 1974, ISBN 0471249807.

[Porter1984] Porter J. E., Boll S. F., "Optimal estimators for spectral restoration of noisy speech", Proceedings of ICASSP'84, pp. 53-56, Mar 1984.

[Prado1994] Prado J., Moulines E., "Frequency domain adaptive filtering with applications to acoustic echo cancellation", Annales des télécommunications, vol. 49, **N°7 − 8**, pp. 414-428, 1994.

[Rabiner1975] L. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing", Prentice-Hall, 1975.

[Rader1972] C.M. Rader, "Discrete convolutions via Mersenne transforms", IEEE Trans. Comput., vol. C-21, pp.1269-1273, Dec. 1972.

[Regalia1989] P. A. Regalia and S. K. Mitra, "Kronecker products, unitary matrices and signal processing applications", SIAM Rev., vol. 31. pp. 586-613, Dec 1989.

[Rosen1993] K. H. Rosen, "Elementary Number Theory and its Applications", Third Edition, Addison-Wesley, 1993.

[Saeed1996] Saeed V. Vaseghi, "Advanced Signal Processing and Digital Noise Reduction", Wiley Teubner, 1996, ISBN 0471958751.

[Sen1997] Sen Kuo, Chein Chen, "An Implementation of Adaptive Filters with the TMS320C25 or the TMS320C30 ; Application Report : SPRA116", Texas Instruments, 1997.

[Shankar1983] S. Shankar Narayan, Allen M. Peterson, Madihally J. Narasimha, "Transform Domain LMS Algorithm", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-31, **N°3**, pp. 609-615, June 1983.

[Shu1988] W. Shu, Y. Tianren, "Algorithm for linear Convolution using Number Theoretic Transforms", Electronics Letters, vol. 24, **N°5**, pp. 249-250, March 1988.

[Simon1996] Simon Haykin, "Adaptive Filter Theory", Prentice Hall, 1996, ISBN 013322760.

[Simon1999] Simon Haykin, "Lessons on Adaptive Systems for Signal Processing, Communications, and Control", IEEE Signal Processing Magazine, pp. 39-48, September 1999.

[Sondhi1967] Sondhi M. M. , "An adaptive echo canceler", Bell Syst. Tech. J., vol. XLVI-3, pp. 497-510, Mar. 1967.

[Soo1987] J. Soo, K. Pang, "A New Structure for Block FIR Adaptive Filtering", Proc. IREECON, pp. 364-367, 1987.

[Soo1990] J. Soo, K. Pang, "Multidelay Block Frequency Domain Adaptive Filter", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, **N°2**, Februray 1990.

[Stasinski1990] R. Stasinski, "Adaptive filters in domains of adaptive transforms", Proc. Singapore Int. Conf. Commun. Syst. ICCS'90 (Singapore), Nov 1990, pp. 18.2.1-18.2.5.

[Stephen1984] Stephen B. Weinstein, "Echo Cancellation in the Telephone Network", IEEE Communications Magazine, pp.8-15, Janvier 1977.

[Thomas1974] Thomas Kailath, "A View of Three Decades of Linear Filtering Theory", IEEE Transactions on Information Theory, vol. 20, **N°2**, pp. 146-181, March 1974.

[Thomas1981] Thomas Kailath, "Equations of Wiener-Hopf Type in Filtering Theory and Related Applications", Norbert Wiener Collected Works, edited by P.Masani, MIT Press, vol, III, pp. 63-64, 1982.

[Tourneur1991] Le Tourneur G., "High quality hands free telephone with acoustic echo control", Proceedings of the International Workshop on Acoustic Echo Control, pp. 99-108, L'Aquila, Italie, September 1991.

[Turbin1997a] Turbin V., Gilloire A., Scalart P., "Comparison of three post-filtering algorithms for residual acoustic echo reduction", Proceedings of ICASSP'97, Vol. 1, pp. 307-310, Munich, Germany, April 1997.

[Turbin1997b] Turbin V., Gilloire A., Scalart P., Beaugeant C., "Using psychoacoustic criteria in acoustic echo cancellation algorithms", Proceedings of the International Workshop on Acoustic Echo and Noise Control, pp. 53-56, London, UK, September 1997.

[Vaseghi1992] Vaseghi S. V., "Advanced signal processing and digital noise reduction", Chapter 9, Wiley Teubner Communications, Queen's University of Belfast, UK, 1996.

[Westall1993] F.A. Westall, S.F.A. IP, “BT Telecommunications Series : Digital Signal Processing in Telecommunications”, Chapter 4.3, ”Echoes in Telephony”, A. Lewis, pp. 114-120, Chapman & Hall, 1993, ISBN 0412477602.

[Weste1993] N. H. E. Weste, K. Eshraghian, “Principles of CMOS VLSI Design : A Systems Perspective”, Second Edition, Addison-Wesley Publ. Comp., 1993.

[Xu1992] S. Xu, L. Dai, S. C. Lee, “Autocorrelation analysis of speech signals using Fermat Number Transform (FNT)”, IEEE Transactions on signal processing, vol. 40, **N°8**, August 1992.

[Yagle1995] A. E. Yagle, “Fast algorithms for matrix multiplication using pseudo-number-theoretic transforms”, IEEE Trans. on Signal Processing, vol. 43, **N°**1, pp. 71-76, January 1995.

[Ye1991] Ye H., Wu X., “A new double-talk detection algorithm based on the orthogonality theorem”, IEEE Trans. on Communications, vol. 39, **N°**11, pp. 1542-1545, November 1991.

[Yon1994] C. H. Yon, C. K. Un, “Fast Multidelay Block Transform-Domain Adaptive Filters Based on a Two-Dimensional Optimum Block Algorithm”, IEEE Transactions on Circuits and Systems-II : Analog and Digital Processing, vol. 41, **N°5**, pp. 337-345, May 1994.

Liste de Publications

Publications dans des journaux internationaux avec comité de lecture

• **H. Alaeddine**, E. H. Baghious, G. Madre, G. Burel, "Realization of Multi-Delay Filter Using Fermat Number Transforms", Journal of The Institut of Electronics, Information and Communication Engineers (IEICE). (*soumise*)

Communications internationales avec comité de lecture

• E. H. Baghious, G. Madre, **H. Alaeddine**, G. Burel, "Realization of adaptive filters blocks digital filters using Fermat Number Transforms", International Symposium on Image/Video Communications over fixed and mobile networks (ISIVC'04), Brest, France, July 2004.

• **H. Alaeddine**, E. H. Baghious, G. Madre, G. Burel, "Realization of Block Robust Adaptive Filters Using Generalized Sliding Fermat Number Transforms", 14^{th} European Signal Processing Conference (EUSIPCO), Florence, Italy, September 2006.

résumé

Résumé : Le principal objectif de notre étude est d'évaluer la possibilité d'un développement en temps réel d'un système d'annulation d'écho acoustique. Pour réduire le coût de calcul de ce système, nous avons approfondi les bases mathématiques de la transformée en nombres entiers (NTT : Number Theoretic Transform) qui est amenée à trouver des applications de plus en plus diverses en traitement du signal. Nous avons introduit plus particulièrement la transformée en nombres de Fermat (FNT : Fermat Number Transform) qui permet une réduction, par rapport à la FFT (Fast Fourier Transform), des nombres de multiplications nécessaires à la réalisation de certaines fonctions telles que les produits de convolution. Pour mettre en évidence cette transformée, nous avons proposé et étudié de nouveaux algorithmes d'annulation d'écho de faible complexité que nous avons traités par blocs et rendus robustes avant de les implanter au moyen de la FNT. Le résultat de cette implantation, comparée à une implantation par la FFT, a montré une forte réduction du nombre de multiplications accompagnée d'une augmentation du nombre d'opérations classiques. Pour réduire cette augmentation, nous avons proposé une nouvelle technique de la transformée, intitulée Generalized Sliding FNT (GSFNT). Celle-ci consiste à calculer la FNT d'une succession de séquences qui diffèrent d'un certain nombre d'échantillons l'une de l'autre. Le résultat des simulations des performances de ces algorithmes d'annulation d'écho, traités au moyen de cette technique, a montré que celle-ci permet de pallier à l'augmentation du nombre d'opérations classiques observée lors d'une implantation en FNT. Enfin, l'implantation des algorithmes d'annulation d'écho en FNT et par une nouvelle procédure de l'algorithme MDF (Multi-Delay Filter) associée à la nouvelle méthode de calcul du pas d'adaptation, a permis une réduction significative de la complexité de calcul.

Mots Clés : ***Filtrage adaptatif, Annulation d'écho acoustique, Filtre à Délais Multiples (MDF), Transformée en nombres entiers (NTT), Transformée en nombres de Fermat (FNT), Sliding Généralisé FFT (GSFFT), Sliding Généralisé FNT (GSFNT).***

Summary : The principal objective of our study is to evaluate the posibility of an acoustic canceler system development in real time. To reduce the computational cost of this system, we looked further into the mathematical bases of the Number Theoretic Transform (NTT) which is meant to find more and more various applications in signal processing. We introduced more particularly the Fermat Number Transform (FNT), which, compared to the Fast Fourier Transform (FFT), allows reduction of several multiplications which are necessary to achieve certain functions such as convolution products. To highlight this transformation, we proposed and studied new algorithms for echo cancelers of low complexity, which we treated by blocks and made robust before implanting them using the FNT. The result of this implementation, compared to an implemetation by the FFT, has shown a strong reduction in the number of multiplications along with an increase in the number of classical operations. To reduce this rise, we proposed a new technique of the transform, entitled Generalized Sliding FNT (GSFNT), which consists in calculating the FNT of a succession of sequences that differ from a certain number of samples from one to another. The numerical simulations show that a GSFNT-based echo canceler helps to remedy the increase in the number of classical operations observed by FNT-based echo canceler. Finally, the implementation of algorithms for echo canceler and through a new procedure of Multi-Delay Filter (MDF) algorithm associated with the new method for the step-size adaptation coefficient, has permitted a significant reduction in the computational complexity.

Keywords : ***Adaptive Filtering, Acoustic Echo Canellation, , Multi-Delay-Filter (MDF), Theoretic Number Transform (NTT), Fermat Number Transform (FNT), Generalized Sliding FFT (GSFFT), Generalized Sliding FNT (GSFNT).***