Évaluation analytique de la précision des systèmes en virgule fixe

Romuald Rocher

N^o d'ordre : 3469

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE RENNES 1

pour obtenir

le grade de : *DOCTEUR DE L'UNIVERSITÉ DE RENNES 1*

Mention : Traitement du Signal

par

Romuald ROCHER

Équipe d'accueil :	Institut de Recherche en Informatique et Systèmes Aléatoires
École Doctorale :	Mathématiques, Télécommunications, Informatique, Signal, Systèmes et Électronique
Composante Universitaire :	École Nationale Supérieure de Sciences Appliquées et de Technologie

Evaluation analytique de la précision des systèmes en virgule fixe

SOUTENUE LE 07 Décembre 2006 devant la commission d'examen

COMPOSITION DU JURY :

MM.	J.M. BROSSIER	Maitre de Conférences (HDR) à l'INPG		Rapporteur
	M. BELLANGER	Professeur au CNAM Paris		Rapporteur
	G. DEMOMENT	Professeur à l'Université Paris XI		
	O. SENTIEYS	Professeur à l'Université de Rennes 1	ENSSAT	
	D. MENARD	Maitre de Conférences à l'Université de Rennes 1	ENSSAT	
	P. SCALART	Professeur à l'Université de Rennes 1	ENSSAT	
	A. GILLOIRE	Directeur scientifique FT R&D		Invité

Resumé

Lors du développement des applications de traitement numérique du signal, les algorithmes sont spécifiés en virgule flottante pour s'affranchir des problèmes liés à la précision des calculs. Cependant, pour satisfaire les contraintes de coût et de consommation, l'implantation de ces applications dans les systèmes embarqués nécessite l'utilisation de l'arithmétique virgule fixe. Ainsi, l'application définie en virgule flottante doit être convertie en une spécification virgule fixe. Pour réduire les temps de mise sur le marché des applications, des outils de conversion automatique de virgule flottante en virgule fixe sont nécessaires. Au sein de ces outils, une étape importante correspond à l'évaluation de la précision de la spécification virgule fixe. En effet, l'utilisation de l'arithmétique virgule fixe se traduit par la présence de sources de bruits liées à l'élimination de bits lors d'un changement de format. Ces bruits se propagent au sein du système et modifient la précision des calculs en sortie de l'application. La dégradation de la précision des calculs doit être maîtrisée afin de garantir l'intégrité de l'algorithme et les performances de l'application. La précision de l'application peut être évaluée par des simulations virgule fixe, mais celles-ci requièrent des temps de calcul élevés. L'utilisation de ce type d'approche pour l'optimisation d'une spécification virgule fixe conduit à des temps d'optimisation prohibitifs. De ce fait, l'objectif de cette thèse est de proposer une nouvelle approche pour l'évaluation automatique de la précision des systèmes en virgule fixe basée sur un modèle analytique. La précision est évaluée en déterminant l'expression du Rapport Signal à Bruit de Quantification (RSBQ) de l'application considérée. Des méthodes ont été proposées pour traiter les systèmes linéaires et invariants dans le temps (LTI) ainsi que les systèmes non-LTI non-récursifs. Ainsi, l'objectif de la thèse est de proposer une méthode d'évaluation automatique de la précision en virgule fixe pour tout type de système et notamment, les systèmes non-LTI présentant une recursion dans le graphe, comme les filtres adaptatifs. Dans un premier temps, des modèles d'évaluation de la précision dédiés aux filtres adaptatifs sont proposés. Dans un second temps, une extension vers un modèle général pour tout type de système est présentée.

Le premier aspect de ce travail concerne le développement de modèles analytiques d'évaluation de la précision dédiés à des applications particulières issues du domaine du filtrage adaptatif. En effet, ces applications non-LTI ne peuvent être traitées par les techniques automatiques classiques. Pour ces systèmes, les modèles dédiés existants ne sont valables que pour une loi de quantification par arrondi convergent. Les modèles proposés par notre approche prennent en compte toutes les lois de quantification, notamment la loi de quantification par troncature. Pour les différents algorithmes adaptatifs et notamment les algorithmes du gradient, une expression analytique de la puissance du bruit en sortie du système est proposée. Ces modèles ont été intégrés au sein d'un générateur d'IP (Intellectual Properties) permettant de générer un composant matériel ou logiciel optimisé d'un point de vue arithmétique.

Le second aspect de notre travail correspond à la définition d'une approche générale d'évaluation analytique de la précision valable pour l'ensemble des systèmes composés d'opérations arithmétiques. Cette méthode se base sur une approche matricielle permettant de traiter plus facilement certains algorithmes de transformée (FFT, DCT). Pour les systèmes récursifs, le déroulement de la récurrence est mis en œuvre. La complexité de notre approche a été déterminée et un modèle de prédiction linéaire a été proposé afin de réduire celle-ci. Ce modèle permet d'accélérer le déroulement de la récurrence. Le modèle a été implanté sur l'outil Matlab et s'intègre au flot général de conversion automatique de virgule flottante en virgule fixe. Cette approche permet ainsi d'optimiser la largeur des opérateurs dans un processus d'optimisation du coût de l'application (consommation d'énergie, surface de l'architecture).

Ces deux approches sont évaluées et comparées en termes de précision et de temps de calcul pour différentes applications, et plus particulièrement, le Least Mean Square (LMS) ou les Algorithmes de Projection Affine (APA). Les deux méthodes permettent d'obtenir des valeurs de la puissance du bruit en sortie du système très proches des valeurs réelles. Le temps d'exécution du modèle sous Matlab a été évalué. L'approche par prédiction linéaire permet de réduire de manière significative le temps de calcul de la puissance du bruit. Les temps d'exécution, dans le cas d'un processus d'optimisation de la largeur des opérateurs, ont été mesurés et comparés à ceux obtenus par une approche par simulations virgule fixe. Notre approche permet de réduire le temps de calcul par rapport aux approches basées sur la simulation au bout de quelques itérations seulement. Ces résultats montrent l'intérêt de notre méthodologie pour réduire le temps de développement des systèmes en virgule fixe.

Table des matières

Table des figures

Liste des tableaux

Glossaire

A	Arrondi
AC	Arrondi Convergent
APA	Algorithmes de Projection Affine
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction-set Processor
CA2	Complément à 2
DSP	Digital Signal Processor
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMS	Least Mean Square
LSB	Last Significant Bit
MCR	Moindres Carrés Récursifs
MSB	Most Significant Bit
NLMS	Normalize Least Mean Square
RLS	Recursive Least-Square
RSBQ	Rapport Signal sur Bruit de Quantification
T	Troncature
TNS	Transmission Numérique du Signal

Notations

E	Terme d'espérance
q	Pas de quantification
$x(n)$	Donnée scalaire à un instant n
$X(n)$	Donnée vectorielle à un instant n
$\mathbf{X}(n)$	Donnée matricielle à un instant n
m_X	Moyenne de la donnée $X(n)$ scalaire ou matricielle
μ_x	Moyenne du scalaire $x(n)$
σ_x^2	Variance du scalaire $x(n)$
$\mathbf{R}_{XX}(\theta) = E[X(n)X^t(n-\theta)]$	Matrice d'autocorrélation des données $X(n)$
$\mathbf{C}_{XX}(\theta) = E\left[(X(n) - m_X)\left(X^t(n-\theta) - m_X^t\right)\right]$	Matrice de covariance des données $X(n)$

Introduction

Les activités dans le secteur de la défense puis l'essor important des télécommunications ces dernières années, ont permis le développement du domaine du traitement numérique du signal et de l'image. Les applications de traitement du signal se retrouvent maintenant dans de nombreux domaines tels que les télécommunications, la défense, le multimédia, l'électronique grand public, le transport mais aussi dans l'instrumentation scientifique, industrielle et médicale. Les avancées technologiques permettent d'implanter des applications de plus en plus complexes. En quelques années, le domaine de la téléphonie mobile a rapidement évolué et plus particulièrement avec l'introduction des systèmes troisième génération. En conséquence, de nombreuses applications à base d'algorithmes de traitement du signal cohabitent au sein d'un même appareil. Ces mobiles intègrent différentes technologies de communication numérique (e.g. GPRS, WCDMA, Bluethooth, WIFI). De plus, de nouveaux services permettent de traiter différents médias tels que la parole (e.g. codeur/décodeur AMR), l'image (e.g. compression JPEG), le son (e.g. codec MP3), la vidéo (e.g. visiophonie H264, DVB-x, MPEGx).

A l'horizon 2010, les technologies permettront d'implanter plusieurs milliards de transistors sur une même puce. Les avancées techniques dans le domaine matériel permettent de répondre aux besoins croissants des applications de traitement numérique du signal. Cependant, la durée de vie des différents produits diminuant, le temps de mise sur le marché (*Time-to-Market*) de ces produits doit être réduit. Ainsi, pour satisfaire les contraintes de temps de mise sur le marché et l'augmentation de la complexité des applications utilisées, des outils de haut-niveau sont nécessaires pour la conception et le développement de ces applications. Ces outils ont un intérêt à toutes les étapes du cycle de développement d'une application de traitement numérique du signal. Ils doivent par exemple faciliter l'implantation de l'application dans les systèmes embarqués en aidant le concepteur à choisir l'architecture adéquate. Ces outils doivent permettre de passer rapidement d'un haut niveau d'abstraction à une description bas niveau.

Contexte de l'étude

Le cycle de développement d'une application de traitement numérique du signal est représenté sur la figure 1. Tout d'abord, le cahier des charges de l'application est défini. Celui-ci contient les différentes fonctions devant être réalisées par le système et les différentes contraintes architecturales du système au niveau temporel et de la consommation d'énergie. Les spécifications opératoires déterminent les performances minimales du système. Par exemple, dans le cadre de systèmes d'annulation d'écho basés sur des techniques de filtrage adaptatif, les performances sont spécifiées à travers l'erreur quadratique moyenne.

L'étape suivante correspond à la spécification de l'application. Cette phase permet d'aboutir à la description complète de l'algorithme de traitement du signal. Pour vérifier que l'algorithme ainsi défini respecte les différents critères liés au besoin de l'application, des simulations sont effectuées. Elles permettent de s'assurer que les performances du système sont satisfaites. Ces simulations peuvent être effectuées au sein d'outils permettant de simuler des algorithmes de TNS tels que Matlab (Mathworks) [68], Scilab (INRIA) [5], SPW (CoWare) [33]. Cette étape utilise

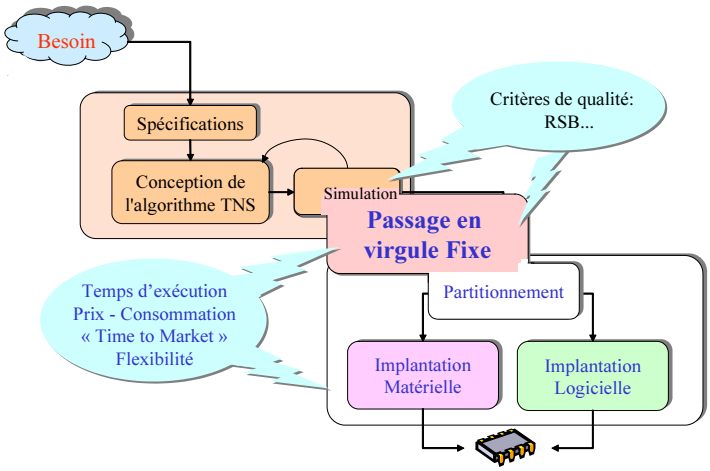

FIG. 1 – *Cycle de développement d'une application de TNS*

l'arithmétique virgule flottante permettant de s'affranchir des problèmes de précision des calculs.

Une fois les algorithmes de TNS définis, ceux-ci sont décrits sous forme d'une code C et implantés dans le système embarqué. Les solutions architecturales permettant d'implanter ces algorithmes sont les ASIC (Application Specific Integrated Circuit), les FPGA (Field Programmable Gate Array), les ASIP (Application Specific Instruction-set Processor), ou bien les DSP (Digital Signal Processor). Ces différentes structures peuvent être associées au sein d'un même système sur puce (SoC : System on Chip). Les différentes contraintes de coût, de consommation d'énergie, de flexibilité et de temps de développement définissent les différentes solutions architecturales choisies. Les contraintes de coût et de consommation propres aux systèmes embarqués requièrent l'utilisation de l'arithmétique virgule fixe. La largeur des données codées en virgule fixe étant plus faible que celle utilisée pour l'arithmétique virgule flottante, la consommation d'énergie et le coût du circuit sont moindres. En général, les structures basées sur l'arithmétique virgule flottante codent les données sur 32 bits alors que les données virgule fixe sont codées sur des largeurs plus faibles. De nombreux DSP travaillent avec des données codées sur 16 bits. De plus, les opérateurs en virgule fixe sont moins complexes que ceux utilisant l'arithmétique virgule flottante. Pour cette dernière, la partie matérielle doit permettre de gérer la mantisse et l'exposant. En conséquence, l'arithmétique virgule fixe est privilégiée dans les systèmes embarqués. Néanmoins, les temps de développement en virgule fixe sont plus importants car les formats des différentes données doivent être définis. Ces formats doivent garantir l'absence de débordement au sein de l'application.

L'application spécifiée en virgule flottante doit donc être convertie en virgule fixe. Cette tâche est longue, fastidieuse et source d'erreurs. La conversion peut représenter 30% du temps lié à l'implantation du système [50, 7]. De plus, la spécification virgule fixe doit satisfaire les contraintes de performance associées l'application. Le temps de mise sur le marché des produits nécessite d'automatiser certaines tâches. En conséquence, des outils de conversion automatique de virgule flottante en virgule fixe sont nécessaires. Ces outils doivent optimiser le coût de l'implantation tout en satisfaisant les contraintes de précision permettant de maintenir les performances de l'application.

Problématique de l'étude

Une méthodologie d'implantation automatique d'applications spécifiées en virgule flottante sur des systèmes embarqués en virgule fixe a été définie au sein de l'équipe R2D2 de l'IRISA.

L'approche pour une implantation logicielle est présentée dans [75] et pour une implantation matérielle dans [54]. Le synoptique de l'approche globale est présentée à la figure 2 et se décompose en plusieurs parties.

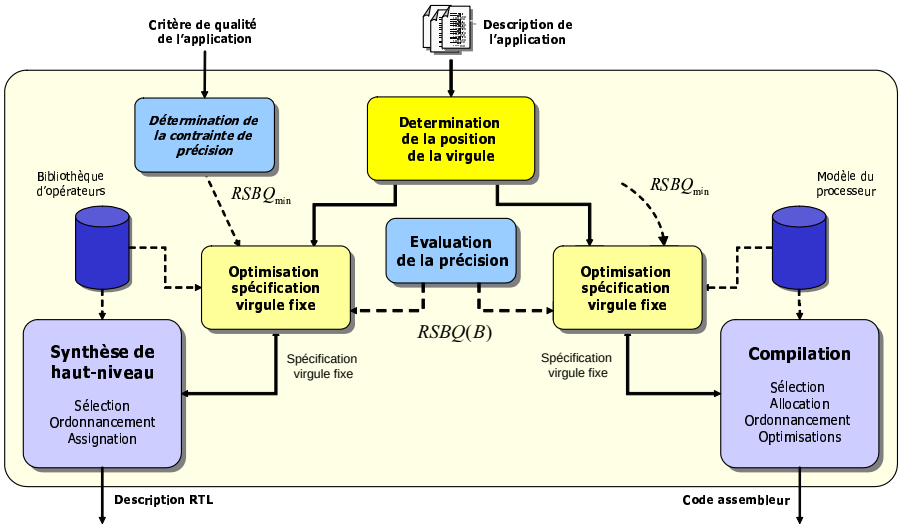

FIG. 2 – *Synoptique de la méthodologie de conversion en virgule fixe développée dans l'équipe R2D2 de l'IRISA*

L'application est spécifiée en langage C à l'aide de types utilisant l'arithmétique virgule flottante. L'utilisateur fournit les contraintes architecturales et les performances devant être satisfaites par l'application après conversion en virgule fixe. La première partie de la méthodologie correspond à la détermination de la position de la virgule. Tout d'abord, la dynamique des données est déterminée afin de connaître l'étendue des valeurs prises par cette donnée. Ensuite, le nombre de bits nécessaires pour coder la partie entière des données est déduit de la dynamique. Cette position de la virgule doit permettre d'éviter la présence de débordements. Des opérations de recadrage sont insérées afin d'aligner la virgule des opérandes d'entrée des additions, modifier la position de la virgule de certaines données ou adapter le codage des données à leur dynamique.

La seconde partie correspond à l'optimisation de la spécification virgule fixe. Cette étape permet de définir le nombre de bits nécessaires pour la partie fractionnaire des données. Ce nombre de bits est un compromis entre le coût de l'implantation et la précision des calculs réalisés. L'arithmétique virgule fixe induit la réalisation des calculs en précision finie se traduisant par l'apparition d'erreurs entre les calculs réalisés en virgule fixe et ceux effectués en précision infinie. L'implantation de l'application en virgule fixe est valable uniquement si les critères de qualité associés à cette application sont respectés malgré les erreurs liées à l'arithmétique virgule fixe. La précision des calculs est évaluée à travers le Rapport Signal à Bruit de Quantification (RSBQ). Une métrique unique est utilisée pour mesurer la qualité des calculs et un lien avec les métriques de qualité propres à l'application est effectué. Une méthode a été proposée dans [97] pour déterminer la valeur minimale du RSBQ ($RSBQ_{min}$) permettant de respecter les performances associées à l'application. L'optimisation de la spécification virgule fixe s'effectue sous une contrainte de précision définie à travers la valeur minimale du RSBQ. Une fois la spécification virgule fixe obtenue, une simulation en virgule fixe est effectuée pour s'assurer que les performances associées à l'application sont bien respectées.

Dans le cadre d'une implantation logicielle au sein de DSP ou de micro-contrôleur, l'architecture du processeur est figée et la largeur des opérateurs pré-définie. Cependant, les processeurs supportent différents types d'instructions. Les instructions de type SWP (Sub-Word Parallelism)

permettent d'accélérer les traitements en exécutant plusieurs opérations en parallèle sur un même opérateur. En contrepartie, la largeur des opérandes est plus faible par rapport aux instructions classiques. A l'opposé, les processeurs proposent des instructions multi-précision. Ces instructions permettent de manipuler des données stockées en mémoire avec une précision plus importante. Chaque instruction multi-précision étant composée d'une suite d'instructions arithmétiques classiques, son temps d'exécution est plus important. L'objectif de la méthodologie est d'optimiser le codage des données afin de minimiser le temps d'exécution et la taille du code sous contrainte de précision. Ainsi, le temps d'exécution du code T_{exe} est minimisé tant que la contrainte de précision $RSBQ_{min}$ est satisfaite tel que présenté à l'équation (1). Les variables de ce problème d'optimisation correspondent au vecteur B représentant les largeurs des données présentes dans l'application. Les valeurs prises par les données appartiennent à l'ensemble E_B regroupant l'ensemble des largeurs supportées par le processeur.

$$\min_{B \in E_B} \Big(T_{exe}(B) \Big) \quad \text{tel que} \quad RSBQ(B) \geq RSBQ_{min} \tag{1}$$

Le temps d'exécution du code est modifié à travers le choix du type d'instruction sélectionnée (classique, SWP, multi-précision) et le placement des opérations de recadrage.

Dans le cas d'une implantation matérielle dans un ASIC ou un FPGA, l'architecture du système doit être définie. Le nombre d'opérateurs et leurs caractéristiques doivent être déterminés lors de la conception de l'architecture. En particulier, la largeur des différents opérateurs arithmétiques est à fixer. Ces choix au niveau des opérateurs conditionnent la précision des calculs réalisés. Ainsi, l'objectif de la méthodologie est de minimiser le coût de l'architecture en termes de surface ou de consommation d'énergie pour une précision des calculs donnée. L'architecture est obtenue à partir de l'assemblage d'éléments issus d'une bibliothèque. Chaque élément est caractérisé en termes de latence, de surface et de consommation d'énergie. Le coût global de l'architecture C_{arch} est déterminé à partir des coûts associés à chaque élément présent dans l'architecture. Ainsi, le coût de l'architecture C_{arch} est minimisé tant que la contrainte de précision $RSBQ_{min}$ est satisfaite tel que présenté à l'équation (2). Les valeurs prises par les largeurs des données appartiennent aux entiers naturels. Chaque variable B_i du processus d'optimisation pouvant prendre un nombre relativement important de valeurs, l'espace de conception en virgule fixe est vaste. En conséquence, le nombre d'itérations du processus d'optimisation peut être élevé.

$$\min_{B \in \mathbb{N}} \Big(C_{arch}(B) \Big) \quad \text{tel que} \quad RSBQ(B) \geq RSBQ_{min} \tag{2}$$

L'étape clé de ces processus d'optimisation de la spécification virgule fixe (équations (2) et (1)) est l'évaluation de la précision des calculs. Cette évaluation de la précision peut s'effectuer au moyen de techniques basées sur la simulation en virgule fixe, ou par une approche analytique. A chaque itération du processus d'optimisation, la précision des calculs ($RSBQ(B)$) est évaluée. En conséquence, les techniques basées sur la simulation conduisent à des temps d'optimisation prohibitifs. Ainsi, dans un souci de réduction du temps d'optimisation par rapport aux méthodes basées sur la simulation, une approche analytique est proposée dans le cadre de ces travaux de recherche. Tout d'abord, l'expression analytique ($RSBQ(B)$) de la métrique de précision est déterminée. Ensuite, le processus d'optimisation est mis en œuvre et à chaque itération de ce processus, la valeur du RSBQ est déterminée et correspond à l'évaluation d'une expression mathématique en un point donnée. En conséquence, ce temps d'évaluation est négligeable.

Des approches analytiques ont été proposées dans [74] ou [28] concernant uniquement les systèmes linéaires et invariants dans le temps (LTI). D'autre part, des approches analytiques pour les systèmes non-LTI non-récursifs ont été définies dans [76] ou [112]. Les systèmes non-LTI récursifs ne peuvent être traités par ces approches. L'objectif de ce travail de recherche est de proposer une approche supportant tous les systèmes à base d'opérations arithmétiques et plus particulièrement les systèmes non-LTI tels que les filtres adaptatifs. Cette approche doit supporter l'ensemble des lois de quantification et ne pas faire d'hypothèse sur la spécification virgule fixe.

Systèmes	Système non-récursif	Système récursif
Système LTI		
Système non-LTI		

TAB. 1 – *Classification des différents systèmes*

Ces systèmes non-LTI sont soit non-linéaires, soit linéaires variants dans le temps (LTV). Cette dernière catégorie regroupe les filtres adaptatifs. Au sein de ces systèmes, une récursion apparaît dans le graphe, permettant ainsi la mise à jour des coefficients. De ce fait, ces systèmes sont considérés comme des systèmes non-LTI récursifs. Les différents systèmes traités peuvent être classifiés selon le tableau 1.

Contributions

Ce travail de recherche s'articule autour de deux axes principaux. Dans un premier temps, des modèles analytiques dédiés pour l'évaluation de la précision ont été établis pour les filtres adaptatifs basés sur l'algorithme du gradient. Les méthodes d'évaluation automatique de la précision ne permettent pas de traiter ce type d'application. Les modèles existants pour les algorithmes LMS et NLMS sont valides uniquement pour une loi de quantification par arrondi convergent car ces modèles ne prennent pas en compte la moyenne du bruit. Ainsi, un modèle valide pour toutes les lois de quantification a été proposé pour les algorithmes LMS, NLMS et Leaky-LMS [88]. A notre connaissance, aucune étude du comportement en virgule fixe n'a été publiée pour les algorithmes APA. Une analyse du fonctionnement en virgule fixe de cet algorithme a été effectuée et un modèle a été proposé [89]. Ces différents modèles définissent l'expression de la puissance du bruit en sortie de ces filtres et sont valides pour toutes les lois de quantification car la moyenne des bruits de quantification est prise en compte.

Ces modèles de précision spécifiques associés à une application donnée trouvent leur place dans le cadre de la conception de composants dédiés. Ainsi, un nouveau type de générateur d'IPs (Intellectual Properties) optimisé d'un point de vue arithmétique a été proposé [86], [87]. L'utilisateur définit la contrainte de précision associée à son application et cet outil génère une architecture pour laquelle la largeur des différents opérateurs est optimisée sous contrainte de précision. Cette approche permet de réduire la surface de l'architecture et la consommation d'énergie par rapport à une approche classique.

Dans un second temps, un modèle général d'évaluation analytique de la précision est proposé pour tout type de système à base d'opérations arithmétiques. Les approches existantes d'évaluation de la précision basées sur les simulations en virgule fixe conduisent à des temps d'optimisation prohibitifs. Les méthodes analytiques existantes sont restreintes à certains types de système (en général des systèmes LTI). Le modèle général proposé est analytique et permet d'obtenir des temps de calcul raisonnables. Ce modèle est valide pour tout type de système à base d'opérations arithmétiques et les résultats connus pour les systèmes LTI sont retrouvés avec notre approche. Ce modèle général se base sur une modélisation matricielle de la propagation des sources de bruit dans le système, permettant ainsi de traiter plus facilement les systèmes manipulant des vecteurs ou des matrices tels que les algorithmes de transformée (FFT, DCT).

Les systèmes récursifs sont traités en déroulant les équations récurrentes associées à l'application. La complexité de cette nouvelle méthode a été déterminée. Une approche basée sur la prédiction linéaire a été proposée pour réduire la complexité de la méthode. Cette approche permet de réduire significativement le temps de calcul nécessaire pour obtenir l'expression du RSBQ. Des gains d'un facteur 100 ont été obtenus pour le temps de calcul. Le biais généré par l'ap-

proche par prédiction linéaire a été mesuré et celui-ci influe peu sur la qualité de l'estimation de la puissance du bruit.

Cette méthode générale a été développée sous l'outil Matlab et s'intègre au flot global de conversion de virgule flottante en virgule fixe. Cette partie d'évaluation de la précision est une étape clé du processus de conversion et notre approche analytique permet d'obtenir des temps d'optimisation raisonnables. Une comparaison de cette méthode par rapport aux approches basées sur des simulations virgule fixe a été effectuée. Celle-ci illustre les gains de temps obtenus et l'intérêt d'utiliser des approches analytiques. Après seulement quelques itérations du processus d'optimisation, notre approche conduit à des temps de calcul plus faibles.

Pour les modèles de bruit dédiés et la méthode générale, la qualité de l'estimateur a été étudiée à travers la mesure de l'erreur relative entre notre estimation et celle basée sur des signaux issus de simulations en virgule fixe. Les résultats montrent que cette erreur est faible et nettement suffisante pour la conception de systèmes embarqués virgule fixe.

Plan de l'étude

Ce travail de recherche est présenté au travers de quatre parties. Le premier chapitre permet d'introduire l'arithmétique virgule fixe. Dans un premier temps, les caractéristiques de cette arithmétique sont présentées. Ceci permet de comparer cette dernière à l'arithmétique virgule flottante et de justifier son utilisation dans le cadre des systèmes embarqués. Dans un deuxième temps, les modèles statistiques des bruits générés par l'arithmétique virgule fixe sont introduits. Dans une dernière partie, les différentes méthodologies existantes pour l'évaluation de la précision des systèmes sont présentées. Les méthodes basées sur la simulation virgule fixe et les méthodes analytiques sont explicitées. Une comparaison entre ces deux types d'approche permet de mettre en avant l'intérêt de l'utilisation d'une méthode analytique.

Le deuxième chapitre présente l'évaluation de la précision dans le cadre des filtres adaptatifs. Tout d'abord, ces filtres sont introduits et leur implantation en virgule fixe est étudiée. Les différentes approches existantes et analysant le comportement de ces systèmes en virgule fixe sont détaillées. Ensuite, les modèles proposés pour les différents filtres adaptatifs basés sur les algorithmes du gradient sont présentés. Les expressions déterminant la puissance du bruit généré en sortie du système en virgule fixe sont développées pour les algorithmes Least Mean Square (LMS), Normalized Least Mean Square (NLMS), Leaky Least Mean Square (Leaky-LMS) ou encore les Algorithmes de Projection Affine (APA). De plus, un modèle général pour les algorithmes du gradient est obtenu et présenté. Dans une dernière partie, la qualité de ces estimateurs est mesurée.

Dans le troisième chapitre, une méthode générale d'évaluation de la précision pour tout type de système est proposée. Dans un premier temps, les caractéristiques statistiques des sources de bruit sont présentées ainsi que leurs modèles de propagation au sein des différentes opérations arithmétiques. Dans une deuxième temps, le système en virgule fixe traité est modélisé. Une analyse du système permettant de modéliser analytiquement la propagation des sources de bruit est effectuée. Cette analyse est ensuite étendue à la composition de systèmes. Ainsi, la méthode générale permet d'étudier une application complète ou chaque bloc la composant est analysé de manière indépendante. Dans un troisième temps, l'expression de la puissance du bruit en sortie du système est calculée de manière analytique en fonction des statistiques des bruits d'entrée et de la modélisation du système parcouru. Cette expression permet de déterminer la précision de la spécification virgule fixe testée. Après une étude de la complexité de notre approche, une technique basée sur la prédiction linéaire est proposée pour réduire cette dernière. Cette méthode générale est illustrée à travers l'exemple de l'algorithme LMS. La qualité de cette approche pour évaluer la précision des calculs est mesurée et les résultats sont présentés dans une dernière partie.

La quatrième partie est consacrée à l'exploitation de ces modèles dans le cadre d'outils de conception de haut-niveau. Tout d'abord, un nouveau type de générateur de composants dédiés utilisant les modèles présentés dans le second chapitre sont présentés. Les différents éléments de cet outil sont détaillés et les résultats d'expérimentations sont présentés pour illustrer l'utilisation de l'outil et montrer l'intérêt de ce type d'approche.

Dans la seconde partie de ce chapitre, l'outil d'évaluation automatique de la précision basé sur le modèle général présenté dans le troisième chapitre est détaillé. Les différentes étapes de cet outil sont décrites et plus particulièrement la partie finale de l'outil correspondant à l'application du modèle général. Les temps d'exécution mesurés pour la partie finale de l'outil implanté sous Matlab sont fournis. L'intérêt de notre approche est souligné à travers les résultats obtenus et la comparaison avec les méthodes basées sur la simulation.

Chapitre 1

Arithmétique virgule fixe

Sommaire

L'objectif de cette partie est de présenter l'arithmétique virgule fixe ainsi que les différentes méthodologies d'évaluation de la précision existantes. Dans un premier temps, l'arithmétique virgule fixe est introduite. Cette arithmétique, plus simple d'implantation que l'arithmétique virgule flottante, conduit à l'existence de bruits dans le système appelés bruits de quantification. Ces bruits, dont le modèle est détaillé dans la seconde partie de ce chapitre, se propagent dans l'ensemble du système pour aboutir à un bruit global en sortie de celui-ci. La qualité de l'application est ainsi modifiée. La précision de l'application en virgule fixe doit être évaluée de façon à s'assurer que les performances de l'application ne sont pas trop dégradées. Ainsi, dans la troisième partie, les différentes méthodes d'évaluation de la précision existantes sont explicitées. Elles sont basées sur deux approches distinctes. Tout d'abord, les méthodes d'évaluation de la précision par simulation sont détaillées. Pour ces méthodes, la précision est évaluée en effectuant des simulations en virgule fixe de l'application. La seconde approche désigne les méthodes analytiques d'évaluation de la précision pour lesquelles une expression mathématique du bruit de quantification en sortie du système est exprimée. Ces deux approches sont par la suite comparées.

1.1 Les différents types de codage

Dans cette section, les différentes arithmétiques utilisées dans un système numérique sont présentées. Tout d'abord, l'arithmétique virgule flottante est développée puis, l'arithmétique virgule fixe est étudiée plus en détails. Finalement, les deux types d'arithmétique sont comparés.

1.1.1 Codage virgule flottante

Les données codées en virgule flottante sont composées de deux parties distinctes : l'exposant et la mantisse, comme représenté sur la figure 1.1. L'exposant permet d'obtenir un facteur d'échelle explicite et variable au cours du traitement. Celui-ci est défini comme une puissance de deux. La mantisse représente la valeur de la donnée divisée par le facteur d'échelle. Afin d'éviter toute ambiguïté, le premier bit de la mantisse représente le coefficient $\frac{1}{2}$ et est fixé à 1. La valeur de ce bit restant fixe au cours du traitement, celui-ci n'est pas représenté dans le code.

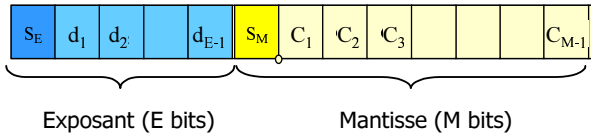

FIG. 1.1 – *Représentation des données en virgule flottante [62]*

La mantisse et l'exposant sont codés avec une représentation en signe valeur absolue, la valeur de la donnée x est la suivante :

$$x = 2^u(-1)^{S_M}\Big(\frac{1}{2} + \sum_{i=1}^{M-1} C_i 2^{-i-1}\Big) \quad \text{avec} \quad u = (-1)^{S_E}\sum_{i=1}^{E-1} d_i 2^i \tag{1.1}$$

D'après l'équation (1.1) la valeur 0 n'est pas représentable. De ce fait, le domaine de définition des valeurs représentables avec ce codage est composé de deux parties distinctes :

$$\mathcal{D}_R = \Big[-2^K; -2^{-K-1}\Big] \bigcup \Big[2^{-K-1}; 2^K\Big] \quad \text{avec} \quad K = 2^{E-1} - 1 \tag{1.2}$$

Le pas de quantification désigne l'écart minimal entre deux valeurs représentables par le code considéré. Il est fonction de la valeur représentée. Pour les valeurs de x comprises dans l'intervalle $[-2^u, -2^{u-1}] \bigcup [2^u, 2^{u-1}]$, le pas de quantification est égal à :

$$q = 2^u 2^{-(M+1)} \tag{1.3}$$

L'expression (1.4) détermine les bornes minimales et maximales du pas de quantification relatif. Celui-ci peut être considéré comme pratiquement constant pour l'ensemble des valeurs de x. Ainsi, la présence d'un facteur d'échelle explicite permet d'adapter le pas de quantification à la valeur de la donnée à coder.

$$2^{-(M+1)} < \frac{q}{|\,x\,|} < 2^{-M} \tag{1.4}$$

La norme IEEE 754 utilise cette représentation en signe valeur absolue. Dans ce cas, la donnée est composée d'un exposant codé sur 8 bits et d'une mantisse sur 24 bits.

1.1.2 Codage virgule fixe

Les données en virgule fixe sont composées d'une partie fractionnaire et d'une partie entière pour lesquelles le nombre de bits alloués reste figé au cours du traitement. L'exposant associé à chaque donnée est implicite et fixe. En conséquence, le facteur d'échelle attaché à la donnée est constant. La figure 1.2 représente une donnée en virgule fixe composée d'un bit de signe et de $b-1$ bits répartis entre la partie entière et la partie fractionnaire. Soit m le paramètre représentant la position de la virgule par rapport au bit le plus significatif (MSB) et n la position de la virgule par rapport au bit le moins significatif (LSB). Dans le cadre d'une donnée signée, les différents paramètres respectent la relation $b = m + n + 1$. Les paramètres m et n sont des entiers et représentent la distance, en nombre de bits, entre la virgule et les bits MSB et LSB. Si

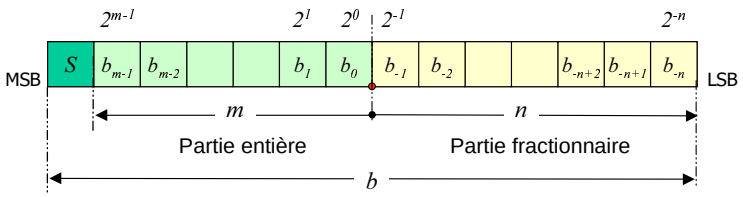

FIG. 1.2 – *Représentation des données en virgule fixe*

ces paramètres sont positifs, ils correspondent respectivement au nombre de bits pour la partie entière et pour la partie fractionnaire.

Le format d'une donnée en virgule fixe est entièrement défini par la représentation choisie et la largeur de sa partie entière et de sa partie fractionnaire. La représentation en complément à deux [62] est détaillée ci-dessous. Cette représentation est très utilisée car elle possède des propriétés arithmétiques très intéressantes pour l'addition et la soustraction. En effet, même si les résultats intermédiaires d'une série d'additions sont en dehors du domaine de définition du codage, le résultat final sera correct si celui-ci appartient au domaine de définition du codage. De plus, l'implantation dans les processeurs numériques des opérateurs traditionnels utilisant ce code est plus simple. Les valeurs négatives de r sont codées par $2^m - r$, ainsi la valeur de la donnée x est égale à :

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i \tag{1.5}$$

Par rapport au code signe valeur absolue, ce code a l'avantage de ne posséder qu'une seule représentation de la valeur zéro. En conséquence, le domaine de définition de ce code n'est pas symétrique par rapport à l'origine, il est composé de $2^{b-1} - 1$ valeurs positives et de 2^{b-1} valeurs négatives :

$$\mathcal{D} = [-2^m; 2^m - 2^{-n}] \tag{1.6}$$

Le pas de quantification est égal à $q = 2^{-n}$.

1.1.3 Comparaison virgule flottante et virgule fixe

Dans cette section, une comparaison entres les codages en virgule flottante et virgule fixe est présentée. Cette analyse est faite selon deux points de vue. Tout d'abord, l'aspect arithmétique est regardé puis l'implantation matérielle de ces deux types de codage est étudiée.

Comparaison d'un point de vue arithmétique

Au niveau arithmétique, pour analyser la qualité du codage, la dynamique et le Rapport Signal à Bruit de Quantification (RSBQ) sont étudiés.

Comparaison de la dynamique La dynamique D_{dB} d'un code correspond au rapport logarithmique entre les valeurs maximale x_{max} et minimale x_{min} de ce code :

$$D_{dB} = 20log\left(\frac{x_{max}}{x_{min}}\right) \tag{1.7}$$

La dynamique permet d'illustrer l'espace des valeurs représentables par le code. L'intérêt d'une dynamique importante est d'assurer l'absence de débordement. Dans le cas de l'arithmétique virgule flottante, la dynamique des données s'exprime sous la forme suivante [122] :

$$D_{dB} \simeq 20\log(2^{2K+1}) \quad \text{avec} \quad K = 2^{E-1} - 1 \tag{1.8}$$

où E désigne le nombre de bits alloués pour l'exposant. Dans le cas de l'arithmétique virgule fixe, la dynamique est donnée par la relation suivante :

$$D_{dB} = 20(b-1)\log(2) = 6(b-1) \tag{1.9}$$

L'évolution de la dynamique pour chacune des deux arithmétiques en fonction de l'évolution du nombre de bits alloués est représentée à la figure 1.3. La dynamique de l'arithmétique virgule fixe est bien linéaire comme défini à l'équation (1.9). De plus, pour un nombre de bits inférieur à 16, la dynamique de l'arithmétique virgule fixe est supérieure à la dynamique de l'arithmétique virgule flottante. Par contre, pour un nombre de bits supérieur à 16, l'arithmétique virgule flottante a une dynamique supérieure. Pour un nombre de bits élevé (supérieur à 25), l'arithmétique virgule flottante montre son intérêt par sa dynamique très importante.

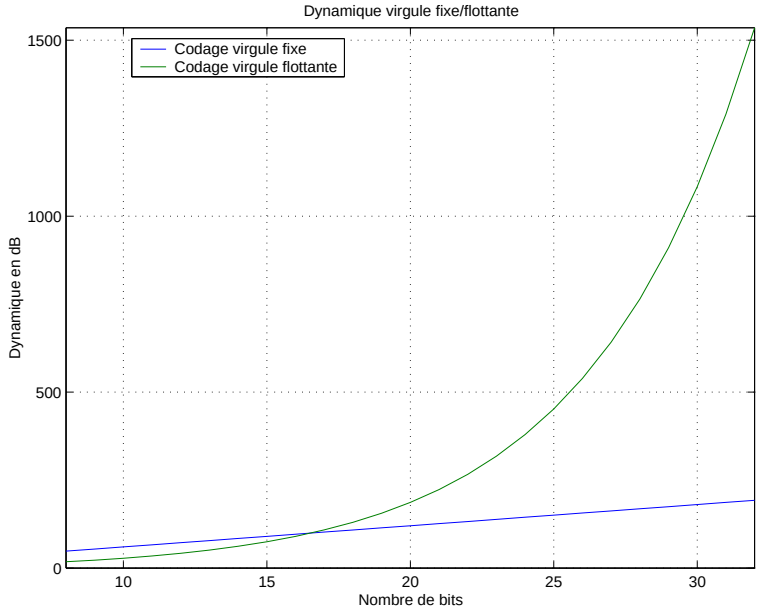

FIG. 1.3 – *Evolution du niveau de dynamique des codages en virgule fixe et en virgule flottante en fonction du nombre de bits utilisés*

Comparaison du RSBQ Le Rapport Signal à Bruit de Quantification est défini comme le rapport entre la puissance du signal P_x et la puissance de l'erreur de quantification P_e. L'expression du RSBQ en dB est la suivante :

$$RSBQ_{dB} = 10log\left(\frac{P_x}{P_e}\right) \tag{1.10}$$

Soit D_x la dynamique linéaire du signal x et K_x le rapport entre la racine carrée de la puissance du signal P_x et sa dynamique D_x. Ainsi, P_x est défini selon l'expression

$$P_x = (K_x D_x)^2 \tag{1.11}$$

Dans le cas de l'arithmétique virgule fixe, le RSBQ est défini par

$$\begin{aligned} RSBQ_{dB} &= 20log(D_x) + 20log(K_x) - 10log(P_e) \\ &= D_{dB} + 20log(K_x) - 10log(P_e) \end{aligned} \tag{1.12}$$

L'expression du RSBQ en fonction de la dynamique est une relation linéaire. Pour l'arithmétique virgule flottante, l'étude de son RSBQ est obtenue à partir de l'expression (1.4) et présentée

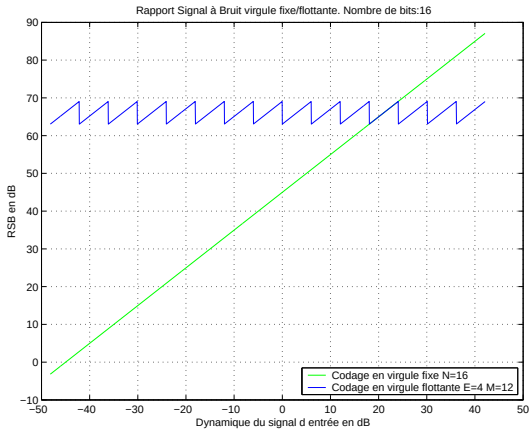

FIG. 1.4 – *Evolution du RSBQ en fonction de la dynamique du signal d'entrée*

dans [57]. La figure 1.4 illustre l'évolution du RSBQ en fonction de la dynamique pour les deux arithmétiques pour des données codées sur 16 bits.

Pour l'arithmétique virgule flottante, le RSBQ est quasi constant en fonction de la dynamique du signal. Ceci est dû à l'utilisation de l'exposant évoluant au cours du temps et permettant ainsi d'adapter le RSBQ à la dynamique du signal. Pour un niveau de dynamique bas, le RSBQ de la virgule flottante est supérieur à celui de la virgule fixe. Cette tendance s'inverse pour des signaux à dynamique élevée dans le cas d'une représentation sur 16 bits.

Comparaison au niveau implantation

Le choix de l'arithmétique pour un système est fonction des contraintes associées à l'implantation. L'arithmétique virgule flottante présente l'avantage d'avoir une plus grande dynamique pour des données de taille 32 bits ainsi qu'un meilleur RSBQ par rapport à l'arithmétique virgule fixe. Néanmoins, comme l'exposant est variable pour les différents données codées, les opérations en arithmétique virgule flottante sont complexes à effectuer. Par exemple, l'addition en virgule flottante s'effectue en trois étapes. Tout d'abord, les données sont dénormalisées pour obtenir leur valeur réelle. Ensuite, l'addition est effectuée. Enfin, la normalisation est mise en œuvre pour adapter l'exposant à la valeur de la sortie de l'addition. Ainsi, le coût d'une opération en virgule flottante est très élevé par rapport au coût d'une opération en virgule fixe pour laquelle les bits sont simplement décalés pour aligner la virgule dans le cas d'une addition. Les systèmes embarqués doivent avoir une consommation en énergie très limitée obtenue grâce à la virgule fixe. En effet, l'arithmétique virgule flottante basée sur le norme IEEE-754 nécessite d'utiliser des données codées sur au moins 32 bits. Or, la majorité des données virgule fixe sont codées sur 16 bits. Ainsi, la largeur des bus et des mémoires au sein des architectures virgule fixe sont plus faibles. Par conséquent, ajouté au fait que les opérateurs en virgule fixe sont moins complexes, la consommation d'énergie et le coût sont moins importants pour une architecture basée sur l'arithmétique virgule fixe. De ce fait, dans le cadre d'une implantation de l'algorithme dans un système embarqué, l'arithmétique virgule fixe sera privilégiée.

Cependant, l'arithmétique virgule fixe présente le désavantage d'avoir un RSBQ plus faible et donc une précision des calculs moins bonne. Ainsi, il est nécessaire de porter attention à la précision des calculs lors d'une implantation en virgule fixe. La précision des calculs peut être déterminée soit par des simulations soit par des méthodes analytiques comme présenté dans la partie 1.3. Pour développer des méthodes analytiques, le bruit de quantification est tout d'abord modélisé. Ces modèles de bruits sont présentés dans la partie suivante.

1.2 Modélisation du bruit de quantification

L'élimination de certains bits lors des calculs génère des bruits appelés bruits de quantification. Dans cette partie, la modélisation du bruit de quantification est étudiée. Une distinction est faite dans le cas où le signal à quantifier est à amplitude continue ou discrète puisque le modèle de bruit en est modifié.

1.2.1 Introduction

Dans cette partie, la modélisation du processus de quantification est étudiée. Les résultats obtenus dans [115] et [116] sont présentés. La quantification d'un signal x de fonction de densité de probabilité (FDP) continue $p_x(x)$ génère un nouveau signal y de densité de probabilité $p_y(y)$ discrète. Cette densité de probabilité est alors composée de M valeurs notées p_l. Chacune de ces valeurs désigne la probabilité que le signal d'entrée x soit compris dans un intervalle Δ_l. La valeur de cet intervalle dépend de la loi de quantification choisie. Celles-ci sont présentées par la suite. La valeur de p_l correspond à l'intégrale de $p_x(x)$ sur l'intervalle Δ_l.

$$p_l = \int_{\Delta_l} p_x(x)dx \tag{1.13}$$

Ces valeurs discrètes p_l sont distantes d'une constante q appelée pas de quantification. Ainsi, la FDP du signal quantifié y s'écrit

$$p_y(y) = \sum_{i=1}^{M} p_i \delta(y - iq) \tag{1.14}$$

La fonction caractéristique est définie comme la Transformée de Fourier de la FDP. En faisant une analogie avec les théories de l'échantillonage, Widrow dans [118] démontre que si la fonction caractéristique du signal d'entrée $\phi_x(u)$ est nulle à l'extérieur de l'intervalle $\mathcal{D} = -[\frac{\pi}{q}, \frac{\pi}{q}]$, alors le signal y peut se modéliser comme la somme de deux variables aléatoires x et e de FDP $p_x(x)$ et $p_e(e)$. Ainsi, la quantification du signal x peut se modéliser comme la somme du signal d'entrée x et d'une variable aléatoire e comme le montre la figure 1.5. Cette variable aléatoire e appelée bruit de quantification est étudiée en détails dans les prochaines parties, en distinguant les cas où le signal d'entrée x est à amplitude discrète ou à amplitude continue.

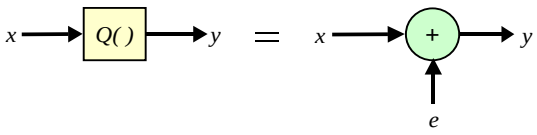

FIG. 1.5 – *Modélisation du processus de quantification du signal x*

1.2.2 Signal à amplitude continue

Dans cette partie, le cas où le signal d'entrée x est à amplitude continue est étudié. Dans ce cas, la FDP du signal x est bien continue. Pour étudier plus en détails le bruit e, les différents intervalles Δ_l sont définis en fonction du type de quantification (troncature et arrondi). Ensuite, les différentes caractéristiques statistiques des bruits sont présentées.

Lois de quantifications

Pour chaque loi de quantification, la densité de probabilité ainsi que la puissance du bruit généré par la quantification sont calculés.

Loi de quantification par troncature La loi de quantification par troncature consiste à choisir la valeur représentable immédiatement inférieure. Cependant, cette loi de quantification génère un écart entre la valeur réelle et la valeur codée. L'intervalle Δ_l représente l'intervalle entre deux valeurs représentables par le codage $[p_l,p_{l+1}[$

$$\Delta_l = [p_l,p_{l+1}[\tag{1.15}$$

La valeur de l'erreur est alors uniformément répartie sur l'intervalle $\mathcal{D} = [0,q]$. La fonction densité de probabilité, représentée sur la figure 1.6, est rectangulaire et s'exprime selon l'équation suivante :

$$p_e(e) = \frac{1}{q}\text{rect}(\frac{e-\frac{q}{2}}{q}) \tag{1.16}$$

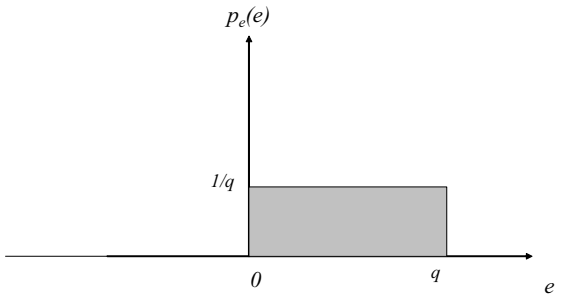

FIG. 1.6 – *Densité de probabilité de la loi de troncature continue*

Dans le cas d'une quantification par troncature, la moyenne et la variance du bruit sont données par l'expression suivante [108] :

$$\mu_e = \int_{-\infty}^{\infty} e\ p(e)\ de = \int_0^q \frac{1}{q}\ e\ de = \frac{q}{2} \tag{1.17}$$

$$\sigma_e^2 = \int_{-\infty}^{\infty} (e-\mu_e)^2\ p(e)de = \int_0^q \frac{1}{q}\Big(e-\frac{q}{2}\Big)^2\ de = \frac{q^2}{12} \tag{1.18}$$

Sa puissance est définie par la somme de sa variance et de sa moyenne au carrée.

$$E(e^2) = \mu_e^2 + \sigma_e^2 = \frac{q^2}{3} \tag{1.19}$$

Loi de quantification par arrondi Pour obtenir une erreur e de moyenne nulle, la loi de quantification par arrondi est introduite. Cette loi affecte à chaque donnée d'entrée x la valeur représentable p_l la plus proche. L'intervalle Δ_l est alors défini par

$$\Delta_l =]p_l - \frac{q}{2},p_l + \frac{q}{2}[\tag{1.20}$$

où q est le pas de quantification. La distribution du bruit est uniforme sur l'intervalle $\mathcal{D} = [-\frac{q}{2},\frac{q}{2}]$ comme le montre la figure 1.7. Sa FDP est définie par l'expression suivante :

$$p_e(e) = \frac{1}{q}\text{rect}(\frac{e}{q}) \tag{1.21}$$

Les expressions de sa moyenne et de sa variance sont définies par Widrow dans [115] et [116] :

$$\mu_e = \int_{-\infty}^{\infty} e\ p(e)de = \int_{-q/2}^{q/2} \frac{1}{q}\ e\ de = 0 \tag{1.22}$$

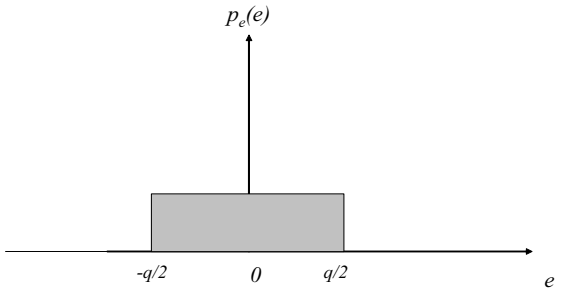

FIG. 1.7 – *Densité de probabilité de la loi d'arrondi continue*

$$\sigma_e^2 = \int_{-\infty}^{\infty} (e - \mu_e)^2 \; p(e) de = \int_{-q/2}^{q/2} \frac{e^2}{q} de = \frac{q^2}{12} \tag{1.23}$$

Sa puissance est alors seulement égale à sa variance de valeur $\frac{q^2}{12}$.

Statistiques du bruit de quantification

Les statistiques du bruit de quantification e sont présentées dans cette partie. Les valeurs de la moyenne et de la variances ont été étudiées pour les différents types de quantification. Pour les autres caractéristiques statistiques, une étude générale est faite.

Autocorrélation du bruit e Dans ce paragraphe, l'étude de la corrélation des bruits entre eux est effectuée. Soient e_1 et e_2 deux variables aléatoires. Dans [122], l'auteur définit la FDP conjointe des deux bruits

$$p_{e_1,e_2}(e_1,e_2) = rect(\frac{e_1}{q}) rect(\frac{e_2}{q}) [p_{x_1,x_2}(x_1,x_2) * (f(e_1).f(e_2))] \tag{1.24}$$

L'auteur a montré que la FDP conjointe $p_{e_1,e_2}(e_1,e_2)$ est égale à :

$$p_{e_1,e_2}(e_1,e_2) = \frac{1}{q} rect(\frac{e_1}{q}) \frac{1}{q} rect(\frac{e_2}{q}) = p_{e_1}(e_1) p_{e_2}(e_2) \tag{1.25}$$

La FDP conjointe étant égale au produit des deux FDP de chacun des bruits, les deux bruits sont indépendants. De plus, la fonction d'autocorrélation $\mathcal{R}_{ee}(\theta)$ à un instant θ est définie par l'expression suivante

$$\mathcal{R}_{ee}(\theta) = \sigma_e^2 \delta(\theta) + \mu_e^2 \tag{1.26}$$

avec δ la fonction Dirac. L'erreur de quantification est donc un bruit blanc. Pour une valeur nulle de θ, la puissance est retrouvée.

Corrélation avec le signal Dans ce paragraphe, l'étude de la corrélation entre le signal d'entrée x et l'erreur e est étudiée. Pour ce faire, Zolzer dans [122] étudie le moment d'ordre deux du signal quantifié y. Comme $y = x + e$, le moment d'ordre deux de y est défini par l'équation suivante

$$E(y^2) = E(x^2) + E(e^2) + 2E(xe) \tag{1.27}$$

Or, le moment d'ordre k du signal correspond à la dérivée $k^{\text{ième}}$ de la fonction caractéristique à l'origine. La dérivée seconde de la fonction caractéristique du signal y permet d'avoir une nouvelle expression de la corrélation entre x et e

$$E(xe) = \sum_{i=-\infty}^{\infty} q\Phi(-i\frac{2\pi}{q})\frac{(-1)^i}{\pi i} \tag{1.28}$$

Donc si $\Phi(-i\frac{2\pi}{q}) = 0$ quelle que soit la valeur de k, l'erreur de quantification n'est pas corrélée au signal. Cette condition est vérifiée dès que le pas de quantification q est inférieur à l'écart-type du signal [122]. Dans le cas contraire, le bruit généré a une puissance supérieure à celle du signal et le bon fonctionnement du système n'est plus assuré.

Conclusion

En conclusion, la quantification d'un signal à amplitude continue génère un bruit de quantification modélisé sous la forme d'un bruit blanc, additif, non-corrélé avec le signal, indépendant des autres signaux et dont la distribution est uniforme sur son espace de représentation.

1.2.3 Signal à amplitude discrète

Dans cette partie, la cas où le signal est à amplitude discrète est traité. La quantification d'un signal à amplitude discrète est présente lors de l'élimination de plusieurs bits d'une donnée. Ce changement de format de la donnée entraîne la génération d'un bruit à amplitude discrète. Ces bruits apparaissent lors d'une mise en mémoire ou lors d'une opération (addition, multiplication) pour lesquels les formats doivent être identiques pour les entrées et les sorties. Ces bruits ont des caractéristiques différentes selon le type de quantification.

Lois de quantifications

Dans le cas de la quantification d'un signal à amplitude discrète, le bruit généré n'a plus une distribution continue mais discrète dont la représentation varie selon le type de loi considérée. Constantinides propose dans [27] un modèle de calcul des moments du bruit en fonction du nombre k de bits éliminés.

Loi de quantification par troncature Cette loi de quantification est la plus utilisée car la plus simple d'implantation. En effet, la troncature consiste à éliminer simplement les bits de poids faible ne pouvant être conservés dans le nouveau format appliqué à la donnée traitée. L'implantation est très simple car seuls les bits de poids forts de la donnée sont conservés. La distribution du bruit est représentée sur la figure 1.8 où k désigne le nombre de bits éliminés. Le modèle développé par Constantinides dans [27] considère l'équiprobabilité des valeurs 0 et 1 pour les bits considérés. Les valeurs des moments sont exprimées dans l'équation suivante :

$$\mu_e = \frac{q}{2}(1 - 2^{-k}) \tag{1.29}$$

$$\sigma_e^2 = \frac{q^2}{12}(1 - 2^{-2k}) \tag{1.30}$$

Loi de quantification par arrondi Dans le cas de la quantification d'un signal à amplitude discrète, l'espace de représentation n'est plus centré autour de 0. En effet, la difficulté revient à déterminer la valeur à affecter à une donnée située sur la médiane d'un intervalle de représentation $p_l + \frac{q}{2}$. Par choix, celle-ci est fixée à la valeur représentable supérieure p_{l+1}. Néanmoins, par ce choix, la moyenne de l'erreur n'est pas nulle mais le biais est beaucoup plus faible que pour la loi de quantification par troncature. Pour résumer, la valeur quantifiée $y = Q(x)$ d'un signal x peut s'exprimer sous la forme

$$y = Q(x) = \begin{cases} p_l & \text{si } p_l \leq x < p_l + \frac{q}{2} \\ p_{l+1} & \text{si } p_l + \frac{q}{2} < x \leq p_{l+1} \\ p_{l+1} & \text{si } x = p_l + \frac{q}{2} \end{cases} \tag{1.31}$$

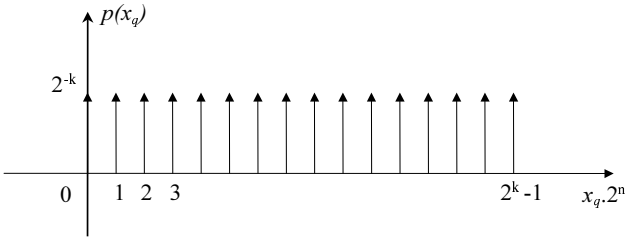

FIG. 1.8 – *Densité de probabilité de la loi de troncature discrète*

En conséquence, la moyenne d'une loi d'arrondi n'est plus nulle comme le montre la figure 1.9.

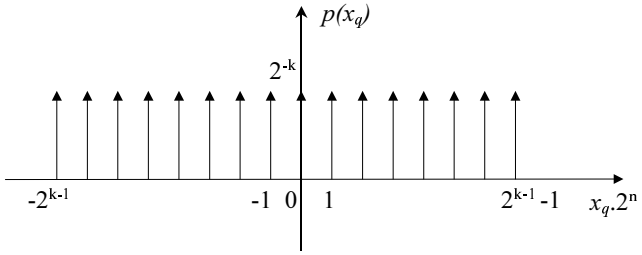

FIG. 1.9 – *Densité de probabilité de la loi d'arrondi discrète*

Ses moments du premier et second ordre sont exprimés par l'expression suivante [27] :

$$\mu_e = \frac{q}{2}(2^{-k}) \tag{1.32}$$

$$\sigma_e^2 = \frac{q^2}{12}(1 - 2^{-2k}) \tag{1.33}$$

Loi de quantification par arrondi convergent Pour éliminer en totalité le biais présent au niveau de l'erreur, la loi par arrondi convergent est introduite [63]. Cette loi est la même que la loi par arrondi conventionnel sauf pour le traitement de la valeur médiane. Celle-ci est alternativement fixée à la valeur supérieure puis à la valeur inférieure représentable. Le biais est alors annulé. Tout ceci est résumé dans l'expression suivante :

$$y = Q(x) = \begin{cases} p_l & \text{si } \; p_l \leq x < p_l + \frac{q}{2} \\ p_{l+1} & \text{si } \; p_l + \frac{q}{2} < x \leq p_{l+1} \\ p_l \text{ ou } p_{l+1} & \text{si } \; x = p_l + \frac{q}{2} \end{cases} \tag{1.34}$$

avec les probabilités égales dans le dernier cas. La valeur médiane est alors distribuée de manière égale sur les deux valeurs représentables les plus proches. La figure 1.10 représente la distribution du bruit. La moyenne de celui-ci est bien nulle.

Ses moments sont alors exprimés par les équations suivantes :

$$\mu_e = 0 \tag{1.35}$$

$$\sigma_e^2 = \frac{q^2}{12}(1 + 2^{-2k+1}) \tag{1.36}$$

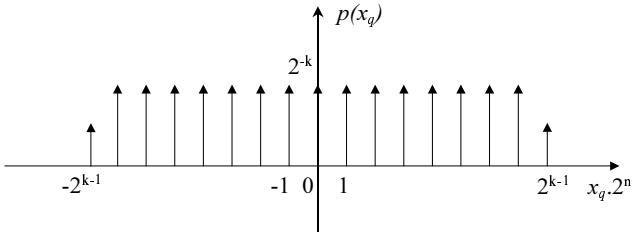

FIG. 1.10 – *Densité de probabilité de la loi d'arrondi convergente discrète*

Conclusion La quantification d'un signal à amplitude discrète génère un bruit dont la distribution n'est plus continue mais discrète. De plus, les moments d'ordre un et deux du bruit font intervenir le nombre de bits k éliminés. Ce modèle s'adapte au bruit généré au niveau des additions, multiplications mais aussi lors des mises en mémoire. Dans [74], l'auteur propose un modèle où le nombre de bits éliminés et valant 0 est pris en compte. Ce modèle se base sur l'hypothèse d'équiprobabilité des bits de valeurs 0 et 1. En effet, un bit éliminé et valant 0 n'introduit pas de bruit. Ce modèle est notamment utilisé dans le cas de la multiplication d'un signal par une constante.

Multiplication par une constante

Soit un signal x multiplié par une constante C. La multiplication de ces deux termes s'effectue sur un nombre N de bits dont n bits pour la partie fractionnaire. Le codage d'une constante contient un certain nombre de 0 dans les bits de poids faible. Ces 0 se retrouvent lors de la multiplication de cette constante par le signal x. Soit l_1 le nombre de bits de poids faibles nuls. Lors de la quantification du résultat de la multiplication, un nombre k de bits est éliminé. Or, ces k bits contiennent les l_1 bits de poids faible nuls comme le montre la figure 1.11

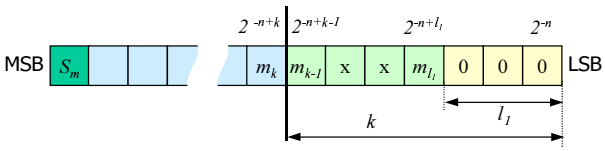

FIG. 1.11 – *Modélisation du résultat de la multiplication d'un signal par une constante*

Ces l_1 bits éliminés n'apportent pas de bruit car ils sont nuls. Il est donc nécessaire de les prendre en compte pour avoir un modèle réaliste. Dans [74], l'auteur étudie la probabilité de chaque bit du résultat du produit xxC et fournit les moments d'ordre un et deux du bruit généré dans le cas de la multiplication d'un signal par une constante en fonction du nombre de bits éliminés k et du nombre de bits de poids faibles l_1 initialement nuls. Les valeurs des moyennes et des variances des différents bruits sont repris dans le tableau 1.1 en fonction de la loi de quantification.

1.2.4 Conclusion

La quantification d'un signal à amplitude continue ou discrète génère une erreur appelée bruit de quantification modélisée par un bruit blanc additif non-corrélé avec le signal, indépendant des autres bruits et dont les caractéristiques statistiques du premier et second ordre dépendent du type de loi de quantification utilisé ainsi que du type de signal (à amplitude continue ou discrète). Le tableau 1.1 résume ces valeurs pour les différentes lois.

Paramètres statistiques	Amplitude-continue	Amplitude discrète	
		$z = x \times y$, $z = x + y$	$z = x \times C$
Arrondi convergent			
μ_e	0	0	0
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k+1})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)+1})$
Arrondi			
μ_e	0	$\frac{q}{2}(2^{-k})$	$\frac{q}{2}(2^{-(k-l_1)})$
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)})$
Troncature			
μ_e	$\frac{q}{2}$	$\frac{q}{2}(1 - 2^{-k})$	$\frac{q}{2}(1 - 2^{-(k-l_1)})$
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)})$

TAB. 1.1 – *Paramètres statistiques du bruit généré*

1.3 Méthodes existantes d'évaluation de la précision

L'utilisation de l'arithmétique virgule fixe limite la précision des calculs et conduit à une erreur de quantification en sortie de l'algorithme correspondant à la différence entre le signal en précision infinie et en précision finie. Il est nécessaire de vérifier que l'emploi de l'arithmétique virgule fixe modifie le comportement de l'algorithme dans une limite raisonnable. L'analyse des performances de l'application en virgule fixe peut être réalisée directement à l'aide de simulations en virgule fixe de l'application. Cependant, cette approche conduit à des temps d'exploration de l'espace de conception en virgule fixe prohibitifs. Ainsi, des métriques intermédiaires sont utilisées pour évaluer la précision des calculs. Le concepteur du système doit ensuite vérifier que sa spécification virgule fixe conduit à une valeur de la métrique de précision supérieure à un seuil pré-défini. Dans une première partie, une présentation des différentes métriques utilisées pour évaluer la précision des systèmes en virgule fixe est développée. Pour déterminer la métrique de précision en sortie de l'algorithme, deux types d'approche peuvent être utilisés. La première consiste à déterminer la valeur de cette métrique à partir de la simulation de l'algorithme en virgule fixe et en virgule flottante. Cette approche est présentée dans la seconde partie. L'approche alternative détermine l'expression analytique de cette métrique en propageant un modèle de bruit au sein du graphe flot de l'algorithme. Les différents modèles analytiques sont détaillés dans la troisième partie. Dans une quatrième partie, un bilan et une comparaison des différents modèles existants sont présentés.

1.3.1 Métriques d'évaluation de la précision

L'évaluation de la précision des applications peut se faire au moyen de différentes métriques présentées ci-dessous. Le critère le plus utilisé pour évaluer la précision est le Rapport Signal à Bruit de Quantification (RSBQ)[74, 59]. Celui-ci correspond au rapport entre la puissance du signal P_y et la puissance du bruit de quantification P_b :

$$RSBQ = \frac{P_y}{P_b} \tag{1.37}$$

En pratique, le RSBQ est exprimé en dB aboutissant à l'équation suivante

$$RSBQ_{dB} = 10log\left(\frac{P_y}{P_b}\right) \tag{1.38}$$

Il permet de déterminer la puissance du bruit en rapport avec celle du signal associé. Cette métrique est très utilisée dans le traitement numérique du signal.

Le seconde métrique modélise les bornes du bruit [85, 43]. L'intervalle $\mathcal{D} = [b_{min},b_{max}]$ définit les valeurs prises par le bruit. Ainsi cette métrique permet d'encadrer et de borner les valeurs du bruit. Elle est utilisée pour garantir que l'erreur de quantification ne dépassera pas une valeur maximale ou minimale.

La dernière métrique représente le nombre de bits significatifs associé à une valeur [22]. Cette méthode évalue le nombre de bits d'une donnée n'étant pas modifiés par le bruit environnant et représentant ainsi de manière correcte la donnée. Par exemple, considérons une donnée codée sur 16 bits. Les bruits de quantification modifient les valeurs des 3 derniers bits. De ce fait, le nombre de bits significatifs est ici de 13. Il représente le nombre de bits non bruités et permettant de modéliser correctement le signal.

1.3.2 Méthodes statistiques par simulation

Introduction

Dans cette partie, un panorama des méthodes statistiques d'évaluation de la précision est dressé. Tout d'abord, les techniques utilisées pour simuler un système en virgule fixe sont présentées puis les méthodes statistiques sont détaillées selon les différentes métriques utilisées.

Les différentes techniques présentées dans cette partie déterminent la valeur de la métrique de précision à partir de la simulation du système en virgule fixe. Afin d'obtenir des résultats précis, il est nécessaire d'utiliser un nombre d'échantillons en entrée du système (N_{ech}) élevé. Soit N_{ops} le nombre d'opérations présentes dans la description de l'algorithme et N_i le nombre de fois que la métrique de précision associée à cette description est évaluée. Une première estimation du nombre de points (N_{pts}) à calculer, présentée à l'équation (1.39) met en évidence la nécessité d'utiliser un simulateur efficace afin d'obtenir des temps de simulation raisonnables.

$$N_{pts} = N_{ech}N_{ops}N_i \tag{1.39}$$

En effet, le temps de simulation est proportionnel au nombre de points N_{pts} et dans l'optique d'une optimisation de la spécification virgule fixe, le nombre de simulation effectuées peut être important. En effet, l'optimisation de la spécification en virgule fixe nécessite de réaliser une nouvelle simulation lorsque le format d'une des données de l'application est modifié. Soit τ le temps de simulation d'un point. Le temps de simulation global du système complet est alors donné par la relation suivante :

$$T_{sim} = \tau N_{ech}N_{ops}N_i \tag{1.40}$$

Pour réduire le temps d'optimisation de la spécification en virgule fixe, les méthodes proposées vont agir sur les différents paramètres de l'expression (1.40).

Tout d'abord, l'objectif des simulateurs virgule fixe est de minimiser le temps de calcul τ d'un point en utilisant des simulateurs efficaces. De plus, le but des méthodes d'évaluation de la précision est de limiter le nombre d'échantillons N_{ech} nécessaires pour obtenir une estimation assez précise. Certaines méthodologies de conversion en virgule fixe vont chercher à réduire le nombre d'itération N_i du processus d'optimisation en limitant l'espace de recherche. Dans la méthode définie par Rupp dans [9] et [8], les largeurs possibles sont restreintes à une série du

type 8,16,32. L'espace de recherche étant limité, ces techniques ne conduisent pas forcément à la solution optimale.

Simulation virgule fixe

Les différentes méthodes statistiques présentées dans cette partie sont basées sur l'utilisation d'un outil permettant de simuler un système en virgule fixe. Cet outil nécessite d'émuler sur une station de travail utilisant l'arithmétique virgule flottante les mécanismes de l'arithmétique virgule fixe (processus de dépassement et de quantification). Pour émuler les mécanismes de l'arithmétique virgule fixe, les méthodes existantes sont soit basées sur les propriétés de la surcharge des opérateurs au sein des langages objet soit sur l'utilisation des types disponibles sur la machine hôte.

La méthode présentée par Kim dans [61] permet de simuler une spécification en virgule fixe, en utilisant les concepts de surcharge des opérateurs au niveau du langage C++. Le type *gFix* [59] [60] est introduit. Celui-ci est composé de la valeur de la donnée, de sa largeur totale, de la largeur de sa partie entière et de différents attributs (type de codage, lois de quantification et de dépassement). Les différents opérateurs arithmétiques sont surchargés afin d'implanter les mécanismes de l'arithmétique virgule fixe. La surcharge des opérateurs met en évidence que l'exécution de l'algorithme en virgule fixe sera nettement plus longue qu'une simulation en virgule flottante. Dans [30], les temps de simulation en virgule flottante et en virgule fixe sont comparés pour quelques applications de traitement du signal. Les temps de simulation obtenus avec la librairie SystemC classique et celle en précision limitée sont respectivement en moyenne 540 et 120 fois supérieurs à ceux obtenus en virgule flottante.

Afin de diminuer le temps de simulation par rapport à la méthode présentée ci-dessus un nouveau type *pFix* a été proposé [61]. Il a pour but d'améliorer le temps de calcul des opérations en virgule fixe en utilisant au mieux, le chemin de données de l'unité de calcul en virgule flottante. Pour cela, le type *pFix* utilise la mantisse des données en virgule flottante pour stocker les données en virgule fixe. Ce type permet d'améliorer nettement le temps de simulation de l'algorithme, mais il est limité par la taille maximale de la mantisse (53 bits dans le cas d'un *double*). Le temps de simulation en virgule fixe d'un filtre IIR d'ordre 4 avec le type *pFix* est 7,5 fois supérieur à celui obtenu en virgule flottante [61].

La méthode proposée par Coster dans [32] utilise les différents types entiers présents sur la machine afin de coder plus efficacement les données de l'algorithme en vue de diminuer le temps d'exécution des simulations. La méthode minimise un coût reflétant le temps d'exécution de la simulation composé du coût lié à l'alignement des données avant les opérations arithmétiques et du coût lié à la réalisation des mécanismes de quantification et de dépassement. Le problème d'optimisation obtenu n'est pas linéaire. De nouvelles approches sont donc nécessaires pour diminuer l'espace de recherche. Les temps de simulation avec la méthode implantée dans HYBRIS (simulateur associé à l'outil FRIDGE [58]) sont 3,6 fois supérieurs à ceux obtenus en virgule flottante [30]. Néanmoins, les temps d'optimisation ne sont pas pris en compte dans ce calcul. Ceux-ci peuvent réduire de manière importante les gains ainsi obtenus.

Vérification directe du critère de qualité de l'application

L'utilisation de techniques de simulation en virgule fixe permet de vérifier directement les critères de qualité associés à l'application. La dégradation des performances liées à l'utilisation de l'arithmétique virgule fixe est mesurée en simulant l'algorithme en virgule fixe et en virgule flottante tel qu'illustré à la figure 1.12. La virgule flottante est considérée comme la référence car l'erreur liée à l'utilisation de cet arithmétique est négligeable. Cependant cette approche atteint rapidement ses limites. Pour vérifier les critères de qualité associés à l'application, de nombreux échantillons doivent être utilisés en entrée

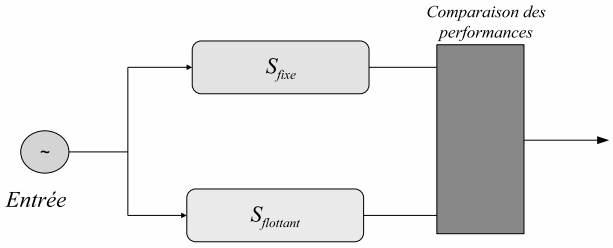

FIG. 1.12 – *Méthode d'évaluation directe des performances de l'application par simulation*

Dans les systèmes de communications numériques, la mesure d'un taux d'erreur binaire faible nécessite de tester un nombre de symboles élevé. Les expérimentations menées sur un récepteur WCDMA montrent que la simulation d'une spécification virgule fixe sous Matlab nécessite plusieurs heures [111]. Les mêmes conclusions ont été obtenues sur un codeur-décodeur MP3 [97]. Ainsi, cette technique est inutilisable pour explorer l'espace de conception en virgule fixe.

Evaluation de la puissance du bruit par simulation

L'évaluation de la précision peut s'effectuer à travers une métrique comme le RSBQ. Dans ce cas, la puissance du bruit en sortie du système (figure 1.13). L'objectif est de réduire le nombre de simulations en virgule fixe pour réduire les temps de simulation. Le bruit de sortie b_y est déterminé en effectuant la différence entre la sortie obtenue en précision infinie et en précision finie.

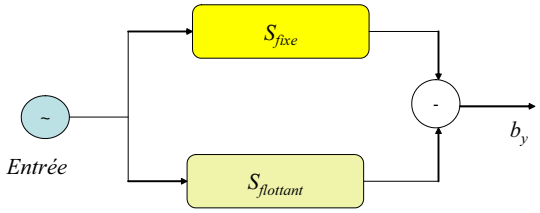

FIG. 1.13 – *Méthode d'évaluation de la puissance du bruit par simulation*

Dans [73], la qualité de l'estimation de la puissance du bruit est analysée en fonction du nombre d'échantillons N_{ech} utilisés en entrée du système. Pour cela, l'erreur relative entre la puissance estimée à l'aide de N_{ech} échantillons et la puissance réelle obtenue avec un nombre de points nettement plus important est mesurée. La figure 1.14 illustre l'erreur obtenue pour des simulations effectuées sur un nombre de points allant de 1 à 1000. L'application testée est un rake receiver. Pour 1000 points l'erreur est inférieure à 8%. Pour obtenir une erreur inférieure à 5%, 5000 échan-

tillons sont nécessaires. Pour avoir des résultats corrects, le nombre de points de simulations doit être relativement important.

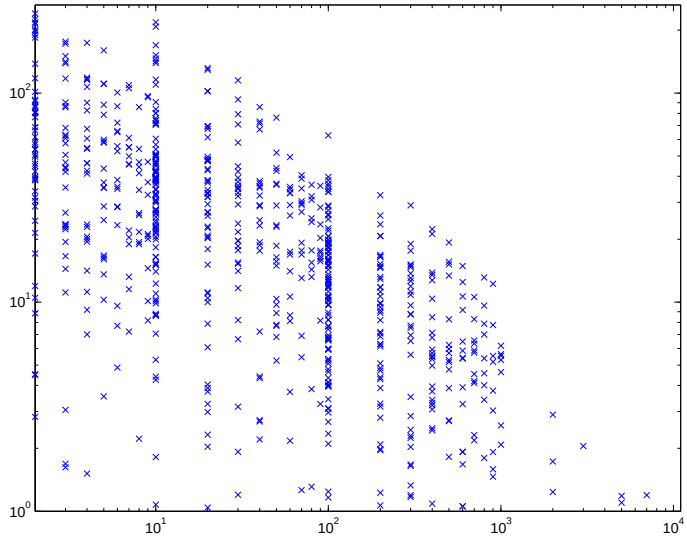

FIG. 1.14 – *Calcul de l'erreur relative entre la puissance du bruit estimée par simulation et sa valeur réelle en fonction du nombre de points de simulations*

L'utilisation d'autres métriques permet de réduire de manière importante ce nombre d'échantillons d'entrée. Pour cela, des hypothèses sur la nature du bruit en sortie doivent être posées comme pour la méthode présentée dans la partie suivante.

Evaluation du nombre de bits significatifs

La méthode présentée dans [22] a pour objectif de déterminer le nombre de bits significatifs. Cette méthode est basée sur l'arithmétique stochastique [23]. L'arithmétique stochastique consiste à modifier de manière aléatoire le dernier bit de chaque donnée. Le programme est exécuté un nombre N de fois en utilisant les mêmes données en entrée de l'application. A partir de ces N échantillons, la valeur de la sortie $\overline{R}$ est estimée en moyennant les échantillons R_i, fournis par les différentes simulations.

$$\overline{R} = \frac{\sum_{i=1}^{N} R_i}{N} \tag{1.41}$$

Avec ce modèle, le nombre de bits significatifs correspond au nombre de bits communs entre la valeur réelle R et celle estimée $\overline{R}$. La variable aléatoire $N(\frac{\overline{R}-R}{s})$ suit une loi de Student à $N-1$ degrés de liberté, où s est défini par

$$s = \sqrt{\frac{\sum_{i=1}^{N}(R_i - \overline{R})^2}{N-1}} \tag{1.42}$$

Le nombre de bits significatifs est déterminé par la relation suivante :

$$C_{\overline{R}} = log_2\left(\frac{\sqrt{N}|\overline{R}|}{s\tau_\beta}\right) \tag{1.43}$$

De plus, la variable τ_β du test de Student pour $N-1$ degrés de liberté et de probabilité β permet de valider cette méthode. Ceci permet aussi de réduire N de façon drastique. En pratique

le nombre d'exécution N peut être réduit à 2 ou 3. De ce fait, l'avantage de ce modèle est de faire appel à un nombre très faible de simulations. Par contre, cette méthode suppose que le bruit de quantification en sortie du système soit gaussien. Cette hypothèse n'est pas valable dans le cas où un seul bruit de quantification prédomine. De plus, la moyenne de la sortie est supposée nulle alors que dans le cas d'une quantification par troncature, la moyenne du bruit n'est pas nulle comme présenté dans la partie 1.2. Cette méthode donne le nombre de bits significatifs pour un échantillon en entrée du système mais ne donne pas d'indications sur le comportement moyen au cours du temps du système en virgule fixe.

Méthode basée sur les bornes de l'erreur

Dans [82], les auteurs utilisent la Théorie des Valeurs Extrêmes pour déterminer les bornes de l'erreur de quantification en sortie du système. Cette théorie se base sur le fait que les valeurs extrêmes d'une suite d'échantillons suivent une distribution bien précise correspondant à la distribution de Gumbel exprimée par la relation suivante :

$$G(x) = e^{-e^{\left(\frac{x-\mu}{\sigma}\right)}} \tag{1.44}$$

où $\sigma = \frac{s\sqrt{6}}{\pi}$ et $\mu = \overline{x} - \sigma\lambda$. Les termes s et $\overline{x}$ désignent l'écart-type et la moyenne des échantillons, et $\lambda = 0.5772$ la constante d'Euler. La méthodologie est basée sur une simulation de N échantillons fournissant N valeurs x_i en sortie du système. A partir de ces valeurs, les termes s et $\overline{x}$ sont déterminés par la relation suivante :

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.45}$$

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N}} \tag{1.46}$$

La probabilité de dépassement f est fixée. La valeur maximale obtenue a est alors donnée par la relation suivante :

$$a = \mu - \sigma ln\left(ln(\frac{1}{1-f})\right) \tag{1.47}$$

Cette valeur a est la valeur maximale de la sortie du système avec une probabilité de dépassement de f. Cette méthode peut s'adapter à l'évaluation de la précision. En effet, les échantillons de bruit sont injectés en entrée. Sa valeur maximale est alors obtenue en sortie. Les résultats montrent qu'avec des tailles d'échantillons variant entre 150 et 600 selon les applications, les résultats obtenus sont optimaux. La méthode permet de diminuer le nombre de simulations de manière importante. Cependant, bien que choisis très faibles, les risques de dépassement ne sont pas nuls. De plus, les temps de simulations ne sont pas indiqués.

Conclusion

Les méthodes par simulations exigent des temps d'exécution importants. Les principales optimisations concernent la réduction du temps de simulation en virgule fixe et la réduction du nombre de simulations en virgule fixe en utilisant des approches semi-analytiques. Les méthodes analytiques, présentées dans la partie suivante, ont le principal avantage de réduire de façon drastique les temps d'optimisation.

1.3.3 Méthodes analytiques

Une approche alternative pour déterminer la précision d'un système consiste à utiliser des méthodes analytiques. Cette méthodologie permet de s'affranchir des simulations nécessitant des temps d'exécution assez longs. L'idée consiste à utiliser des modèles mathématiques fournissant une connaissance sur le bruit en sortie du système. Plusieurs types de métriques peuvent être prises en compte comme la puissance du bruit, le nombre de bits significatifs ou encore les bornes de l'erreur de quantification.

Détermination des bornes de l'erreur de façon analytique

Arithmétique d'intervalles Une méthode d'évaluation analytique des bornes de l'erreur utilise l'arithmétique d'intervalle [85] définissant les bornes des données traitées. Celle-ci est basée sur la propagation des intervalles dans lesquels se trouvent les données. Les règles de propagation sont les suivantes. Tout d'abord, les intervalles de définition de nos entrées sont fixés. Un intervalle $\mathcal{D}_x$ est associé à chaque données x

$$\mathcal{D}_x = [x_{min},x_{max}] \tag{1.48}$$

Cet intervalle $\mathcal{D}_x$ fixe les valeurs minimale et maximale pouvant être prises par la donnée x. De même, pour une donnée y, son intervalle de définition $\mathcal{D}_y$ est donné par $\mathcal{D}_y = [y_{min},y_{max}]$. La propagation de ces intervalles lors du passage dans différentes opérations telles que l'addition, la multiplication de signaux ou la multiplication par une constante, est fournie dans les équations suivantes :

$$\begin{aligned}
\mathcal{D}_{x+y} &= [x_{min} + y_{min},x_{max} + y_{max}] \\
\mathcal{D}_{xy} &= [min(x_{min}y_{min},x_{min}y_{max},x_{max}y_{min},x_{max}y_{max}) \\
&\quad , \quad max(x_{min}y_{min},x_{min}y_{max},x_{max}y_{min},x_{max}y_{max})] \\
\mathcal{D}_{ax} &= [ax_{min},ax_{max}] \text{ si } a > 0 \\
&= [ax_{max},ax_{min}] \text{ si } a < 0
\end{aligned} \tag{1.49}$$

L'un des défauts majeurs de cette méthode vient de l'estimation pessimiste faite de l'intervalle de définition des données. Pour illustrer ce phénomène, considérons la fonction f suivante :

$$f(e) = e(e-2) \tag{1.50}$$

et e est une erreur définie dans l'intervalle $\mathcal{D}_x = [0,q]$. En appliquant les règles de propagation des intervalles, il vient

$$\mathcal{D}_{f(e)} = [-2q,0] \tag{1.51}$$

Or en réécrivant $f(e)$ sous la forme

$$f(e) = (e-1)^2 - 1 \tag{1.52}$$

son intervalle de définition devient

$$\mathcal{D}_{f(x)} = [-q,0] \tag{1.53}$$

Ainsi, selon l'expression utilisée, l'intervalle sera plus ou moins large. Avec l'exemple précédent, la seconde forme de $f(e)$ permet de réduire de moitié la taille de l'intervalle. Ceci est dû au fait que l'arithmétique d'intervalles fait une estimation pessimiste de l'intervalle de définition et ne prend pas en compte les corrélations entre les données. De ce fait, les résultats sont obtenus pour le pire cas. De plus, cette méthode connaît également ses limites avec les systèmes récursifs pour lesquels l'intervalle de définition de la sortie du système ne peut être déterminé. Soit l'application

$$y(n) = ay(n-1) + e(n) \tag{1.54}$$

où $[y_{min}(n-1),y_{max}(n-1)]$ désigne l'intervalle de $y(n-1)$ et $[0,q]$ celui de e. La valeur maximale de l'intervalle de $y(n)$ est donnée par la relation suivante

$$y_{max}(n) = ay_{max}(n-1) + q \tag{1.55}$$

La valeur y_{max} dépend d'elle-même. Cela signifie que le graphe flot de signal modélisant l'application contient des circuits. Ainsi le parcours de graphe permettant de déterminer les différents intervalles des données est infini. De ce fait, cette méthode peut s'appliquer uniquement aux systèmes dont la sortie ne dépend que des valeurs d'entrée.

Arithmétique affine Pour résoudre une partie des problèmes de l'arithmétique d'intervalle concernant la non prise en compte des corrélations entre les données, l'arithmétique affine a été introduite [34]. Une donnée x est définie par l'expression linéaire suivante :

$$x = x_0 + \epsilon_1 x_1 + ... + \epsilon_n x_n \tag{1.56}$$

avec $\epsilon_i \in [-1,1]$. Chaque symbole ϵ_i est indépendant des autres et représente la valeur prise par x selon la composante x_i. Ces derniers étant les bornes de x. Ainsi x_0 représente la moyenne des valeurs que x peut prendre. Cette méthode permet de prendre en compte la corrélation spatiale entre les différentes données. Pour propager les valeurs de l'intervalle de définition, la forme affine est développée en fonction des ϵ_i. Soit deux données x et y définies par

$$x = x_0 + \epsilon_1 x_1 + ... + \epsilon_n x_n \tag{1.57}$$
$$y = y_0 + \epsilon_1 y_1 + ... + \epsilon_n y_n \tag{1.58}$$

La somme de ces deux données est une fonction affine des ϵ_i, soit

$$x + y = x_0 + y_0 + \epsilon_1(x_1 + y_1) + ... + \epsilon_n(x_n + y_n) \tag{1.59}$$

Pour la multiplication, le résultat est plus complexe. En effet, cette opération n'est plus une opération affine puisque la multiplication de x par y donne

$$x + y = x_0 y_0 + \sum_{i=1}^{n}(x_0 y_i + y_0 x_i)\epsilon_i + \sum_{i=1}^{n} x_i \epsilon_i \sum_{j=1}^{n} y_j \epsilon_j \tag{1.60}$$

Cette forme n'étant plus affine, il convent alors de majorer le dernier terme de cette expression en introduisant par exemple z_{n+1} tel que

$$|z_{n+1}| > \sum_{i=1}^{n} x_i \epsilon_i \sum_{j=1}^{n} y_j \epsilon_j \tag{1.61}$$

soit par exemple

$$z_{n+1} > \sum_{i=1}^{n} |x_i| \sum_{j=1}^{n} |y_j| \tag{1.62}$$

et si ϵ_{n+1} représente le facteur associé à z_{n+1}, il vient

$$x + y = x_0 y_0 + \sum_{i=1}^{n}(x_0 y_i + y_0 x_i)\epsilon_i + z_{n+1}\epsilon_{n+1} \tag{1.63}$$

L'exemple traité précédemment est repris, à savoir $f(e) = e(e-2)$ et $\mathcal{D}_x = [0,q]$. L'expression affine de e est alors

$$e = \frac{q}{2} + \frac{q}{2}\epsilon_1 \tag{1.64}$$

Donc

$$e - 2 = \frac{q}{2} - 2 + \frac{q}{2}\epsilon_1 \tag{1.65}$$

De ce fait, $f(e)$ s'écrit

$$f(e) = \frac{q^2}{4} - q + (\frac{q^2}{2} - q)\epsilon_1 + z_2\epsilon_2 \tag{1.66}$$

avec

$$z_2 = \frac{q}{2}.\frac{q}{2} = \frac{q^2}{4} \tag{1.67}$$

d'où

$$f(x) = \frac{q^2}{4} - q + (\frac{q^2}{2} - q)\epsilon_1 + z_2\epsilon_2 + \frac{q^2}{4}\epsilon_2 \tag{1.68}$$

Cette expression fournit comme bornes l'intervalle $[\frac{q^2}{2} - 2q,0]$. Cet intervalle est meilleur que celui obtenu pour l'arithmétique d'intervalle ($[-2q,0]$) mais est encore un peu éloigné de l'intervalle réel $[-q,0]$ puisque $q < 1$. Cette méthode peut être appliquée pour les systèmes en virgule flottante [43] ou en virgule fixe [44]. Néanmoins, bien que l'arithmétique affine utilise les corrélations spatiales entre les données permettant une amélioration de l'évaluation de l'intervalle de définition des données, la corrélation temporelle n'est pas prise en compte. En effet, dans le domaine du traitement du signal, la plupart des données sont réutilisées dans le temps. De ce fait, des échantillons $x(n)$ apparaissent en même temps que leurs versions antérieures $x(n-1)$,$x(n-2)$ dans différents systèmes comme par exemple les filtres FIR. Pour optimiser l'evaluation des intervalles de définition des différentes données, il est donc nécessaire de prendre en compte les corrélations temporelles existantes entre les données limitant l'utilisation de l'arithmétique affine dans le domaine du Traitement Numérique du Signal.

Méthode basée dur la théorie de la perturbation La méthode proposée par Wadekar dans [114] permet de déterminer les bornes de l'erreur en sortie de l'application. L'evaluation de la précision au sein de cette méthodologie est effectuée au moyen d'un développement de Taylor au second ordre. Soit $f(u,v)$ la sortie d'un opérateur contenant deux entrées u et v et δ_u et δ_v les erreurs au pire cas sur chacune des deux variables. L'erreur en sortie, déterminée par un développement de Taylor à l'ordre deux, est égale à

$$\delta_f = \frac{\partial f}{\partial u}(\delta_u) + \frac{\partial f}{\partial v}(\delta_v) + \frac{1}{2!}\left[\frac{\partial^2 f}{\partial u^2}(\delta_u)^2 + 2\frac{\partial^2 f}{\partial u \partial v}(\delta_u\delta_v) + \frac{\partial^2 f}{\partial v^2}(\delta_v)^2\right] \tag{1.69}$$

L'erreur au pire cas en sortie de l'opérateur est ainsi obtenue et propagée pour déterminer l'erreur en sortie du système. Ce modèle possède l'avantage de fonctionner pour les systèmes LTI et non-LTI. Cependant, les systèmes récursifs ne peuvent être traités car cette technique se base sur la propagation des bruits au sein du système.

Méthodes analytiques de calcul de la puissance du bruit

Méthodes basées sur la théorie de la perturbation Dans [26], une approche analytique pour évaluer la précision d'un système est proposée. Celle-ci est basée sur l'analyse de la perturbation présentée dans [114] . La quantification d'un signal génère un bruit de quantification pouvant être considéré comme une perturbation sur le signal. De ce fait, chaque perturbation sur les entrées d'un opérateur va générer une perturbation en sortie de celui-ci. La valeur de cette

perturbation en sortie de l'opérateur est déterminée de manière analytique. La sortie Y d'un opérateur est fonction de ses entrées $X1$,...,Xn

$$Y = f(X1,X2,...,Xn) \tag{1.70}$$

A chaque donnée d'entrée Xi est associée une perturbation xi conduisant à une autre perturbation y en sortie. Cette valeur y est calculée en effectuant un développement de Taylor au 1er ordre donnant l'expression suivante :

$$y = x1\frac{df}{dX1} + ... + xn\frac{df}{dXn} \tag{1.71}$$

Cette méthode permet de linéariser les différents opérateurs de façon à se ramener à un système LTI et reprendre la méthode présentée dans [29]. La précision du système est alors étudiée à partir d'un graphe flot de données (GFD). Le GFD initial est modifié en introduisant les différentes sources de bruit et leur propagation selon la méthode de la perturbation. Une fois le GDF final défini, le bruit en sortie est obtenu. Cette méthode est donc semi-analytique puisqu'elle ne permet pas d'avoir une expression global du bruit. Seule une expression au niveau de chaque opérateur est obtenue et donc des simulations sont nécessaires. Celles-ci augmentent le temps de calcul.

La méthode d'évaluation de la précision de C. Shi dans [103] est basée sur la théorie de la perturbation [103],[100]. Soient x_1,...,x_n les n entrées d'un opérateur auxquelles sont associés les bruits ϵ_1,...,ϵ_n. La sortie y_{FP} de cette opérateur en précision finie est donc une fonction ϕ_t dépendant de ces entrées à l'instant t . Un développement de Taylor à l'ordre 2 de la sortie du système en précision finie en fonction des ϵ_i est alors mise en œuvre et conduit à

$$\phi_t(x_1,....,x_n,\epsilon_1,...,\epsilon_n) = \phi_t(x_1,....,x_n,0,...,0) \tag{1.72}$$

$$+ \sum_{i=1}^{n} \frac{\partial \phi_t}{\partial \epsilon_i}\epsilon_i + \sum_{i,j=1}^{n} \frac{\partial \phi_t}{\partial \epsilon_i \epsilon_j}\epsilon_i\epsilon_j \tag{1.73}$$

Cette méthode est ensuite insérée dans un modèle plus général de conversion de virgule flottante en virgule fixe [101] et [102]. La puissance du bruit est alors donnée par la relation suivante :

$$Pb = \overline{\mu}^t B \overline{\mu} + \sum_{i \in \text{chemin de données}} C_i 2^{-2n_i} \tag{1.74}$$

où n_i désigne le nombre de bits de la partie fractionnaire des données et C_i des constantes multiplicatrices. Le terme $\overline{\mu}$ est un vecteur contenant l'ensemble des moyennes des différents bruits de quantification générés. Le terme B est une matrice carrée de taille N_e, où N_e désigne le nombre de bruits générés. Les termes B et C_i sont alors déterminés par des simulations. Le nombre de simulations nécessaires est alors de l'ordre de N_e^2. Le problème posé par cette méthode est le nombre de simulations effectuées. En effet, le nombre de simulations augmente avec l'accroissement du nombre de sources de bruit. De ce fait, le temps requis pour l'évaluation de la précision augmente en conséquence.

Méthodes basées sur les modèles de bruits Dans [112, 67], une méthode de calcul analytique du bruit de quantification est proposée. Cette méthode est basée sur la propagation de modèles de bruit au sein de la description de l'algorithme. Chaque opérateur est modélisé par une source de bruit propagé et une source de bruit généré. Afin d'obtenir des expressions des bruits générés et propagés indépendantes de la statistique des signaux utilisés en entrée des opérateurs, l'auteur a posé plusieurs hypothèses. Les variables en entrée de chaque opérateur sont considérées comme indépendantes et centrées. Les bruits générés lors d'un changement de format sont assimilés à des variables aléatoires centrées. Ainsi, ce modèle n'est valide que pour les opérateurs utilisant une loi de quantification par arrondi. Chaque variable est codée en virgule fixe cadrée

à gauche et afin d'être indépendant de la puissance de chaque donnée, celle-ci est bornée par la valeur 1. Ces différentes simplifications permettent d'obtenir une expression de la puissance du bruit en sortie, proportionnelle au carré du pas de quantification q.
L'application est modélisée par un graphe flot de données et celui-ci est parcouru des entrées vers la sortie et pour chaque nœud le bruit est calculé à partir des prédécesseurs de ce nœud. Cette méthode permet de calculer plus rapidement le bruit de quantification en sortie de l'algorithme par rapport aux méthodes basées sur la simulation. Cependant, l'utilisation d'une technique de propagation d'un modèle de bruit ne permet pas de traiter les graphes possédant des cycles. Ainsi, la méthode ne peut pas être appliquée aux systèmes récursifs.

Le modèle, présenté dans [78] et [77], permet de traiter les systèmes LTI. Cette approche utilise les fonctions de transfert définissant le système considéré. Soit un système LTI de sortie $y(n)$ composé de N_e entrées $x_j(n)$. Soit $H_j(z)$ la fonction de transfert entre l'entrée $x_j(n)$ et la sortie $y(n)$, et h_j la réponse impulsionnelle associée à $H_j(z)$. Dans ces conditions, le système peut s'écrire sous la forme

$$y(n) = \sum_{j=0}^{N_e-1} h_j * x_j(n) \tag{1.75}$$

Dans le cas d'une implémentation en virgule fixe, à chaque échantillon du signal d'entrée $x_j(n)$ est associé un bruit de quantification $b_j(n)$. De plus, au cours des recadrages internes au systèmes N_g sources de bruit b_{g_i} vont également apparaître. Soit $H_{g_i}(z)$ la fonction de transfert entre la source de bruit b_{g_i} et la sortie $y'(n)$ en virgule fixe. Le bruit en sortie du système $b_y(n)$ (figure 1.15) s'écrit alors

$$b_y(n) = \sum_{j=0}^{Ne-1} h_j * b_j(n) + \sum_{k=0}^{Ng-1} h_{g_k} * b_{g_k}(n) \tag{1.76}$$

En réindexant les termes, l'équation suivante est obtenue

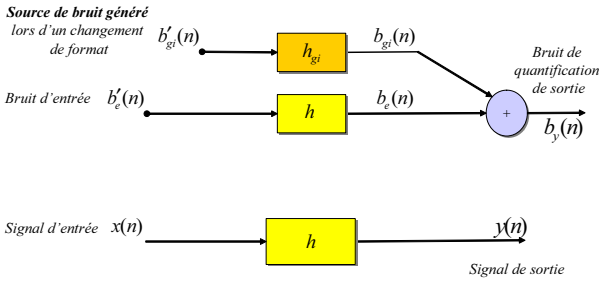

FIG. 1.15 – *Modélisation d'un système LTI*

$$b_y(n) = \sum_{i=0}^{Nq-1} h_i * b_{q_i}(n) \tag{1.77}$$

où

$$\begin{aligned} N_q &= N_e + N_g \\ h_i &= h_{g_{i-N_e}} \quad \text{pour } i \in]Ne,Nq] \\ b_{q_i}(n) &= b_{g_{i-N_e}}(n) \quad \text{pour } i \in]Ne,Nq] \\ b_{q_i}(n) &= b_i(n) \quad \text{pour } i \in [0,Ne] \end{aligned} \tag{1.78}$$

Soit $m_{b_{q_i}}$ la moyenne du bruit d'entrée b_{q_i} et $\sigma^2_{b_{q_i}}$ sa variance. La puissance du bruit est alors donnée par [80]

$$P_{b_y} = \sum_{i=0}^{N_q} \sigma^2_{b_{q_i}} \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_i(e^{j\Omega})|^2 d\Omega + \left(m_{b_{q_i}} H_i(1)\right)^2 \tag{1.79}$$

Cette méthode permet de fournir la puissance du bruit dans le cas des systèmes LTI. Cette approche se base sur la détermination de la fonction de transfert liant la source de bruit à la sortie du système. Les différentes fonctions de transfert sont déterminées automatiquement à partir de l'analyse du graphe flot de signal de l'application. Cependant, cette méthode ne peut pas s'adapter aux systèmes non-LTI pour lesquelles la notion de fonction de transfert n'est pas définie.

Pour traiter les systèmes non-LTI non récursifs, un modèle de calcul de la puissance du bruit a été présenté dans [76]. Ce modèle se base sur la propagation des bruits. Cette méthodologie a été développée pour des systèmes utilisant des opérateurs classiques du type addition, multiplication et division. Chaque source de bruit b_{q_i} est propagée à travers K_i opérations υ_{K_i}, et conduit à un bruit b'_{q_i} en sortie du système. Ce bruit est le produit de chaque source de bruit b_{q_i} et des différents signaux α_k associés aux opérations υ_{K_i} incluses dans la propagation des sources de bruit b_{q_i} comme le montre la figure 1.16.

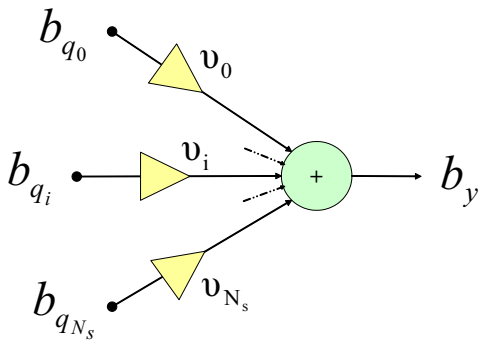

FIG. 1.16 – *Modélisation d'un système non-LTI non-récursif*

$$b'_{q_i} = b_{q_i} \prod_{k=1}^{K_i} \alpha_k = b_{q_i} \upsilon_i \qquad \text{avec} \quad \upsilon_i = \prod_{k=1}^{K_i} \alpha_k \tag{1.80}$$

Le bruit de sortie b_y peut s'exprimer comme suit

$$b_y = \sum_{i=0}^{N_s-1} b'_{q_i} = \sum_{i=0}^{N_s-1} b_{q_i} \upsilon_i \tag{1.81}$$

La puissance du bruit, présentée dans [76], est donnée par l'expression suivante :

$$P_{b_y} = \sum_{i=0}^{N_s} E(b_{q_i}^2)E(\upsilon_i^2) + 2\sum_{i=0}^{N_s}\sum_{\substack{j=0 \\ j>i}}^{N_s} E(b_{q_i})E(b_{q_j})E(\upsilon_i\upsilon_j) \tag{1.82}$$

Cette expression permet de déterminer la puissance du bruit en sortie d'un système non-LTI non-récursif. Par contre, cette méthode ne peut pas traiter les systèmes non-LTI possédant une récursion comme les filtres adaptatifs.

1.3.4 Bilan

Les deux types d'approche permettant d'évaluer la précision d'une implantation en virgule fixe ont été détaillés. Les méthodes statistiques basées sur la simulation de l'algorithme en virgule fixe permettent de traiter tous les types d'algorithme de traitement numérique du signal. Cependant, ces techniques conduisent à des temps de simulation élevés et plus particulièrement si elles sont intégrées au sein d'un processus d'optimisation du format des données où de multiples simulations sont nécessaires. En effet, l'émulation des mécanismes de l'arithmétique virgule fixe augmente le temps de simulation par rapport à la virgule flottante. De plus, il est nécessaire d'utiliser de nombreux échantillons d'entrée afin d'obtenir des paramètres statistiques réalistes.

La seconde approche correspond aux méthodes analytiques permettant d'obtenir l'expression de la métrique définie. Cette expression fournit plus d'information sur le comportement du bruit au sein de l'algorithme par rapport à une méthode basée sur la simulation. En effet, cette approche permet d'analyser l'influence de telle ou telle opération sur la précision globale des calculs. La détermination de l'expression de la métrique est réalisée une seule fois et ensuite le temps de calcul de celle-ci correspond au temps d'évaluation de l'expression analytique. Ce temps de calcul est définitivement plus faible que le temps de simulation de l'application en virgule fixe. Sur la figure 1.17, le temps d'optimisation en fonction du nombre d'évaluations de la métrique de précision est représenté dans les cas des méthodes par simulation ou des méthodes analytiques. Le gain en temps des méthodes analytiques est très important sauf pour un nombre de simulation très faible comme dans [22].

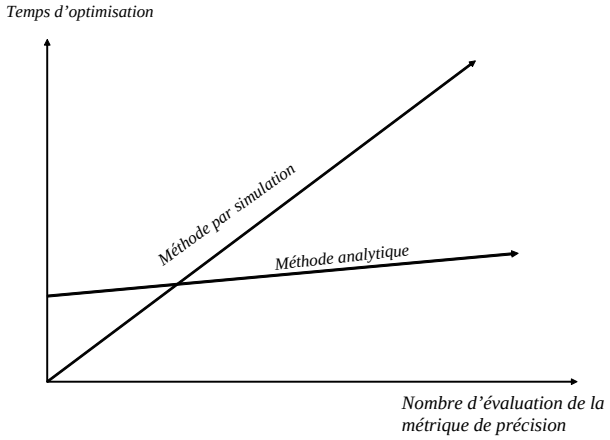

FIG. 1.17 – *Comparaison en temps d'optimisation des méthodes analytiques et par simulation.*

Néanmoins cette approche est très restrictive au niveau des caractéristiques du bruit de sortie. La méthode analytique est donc privilégiée. Ces approches analytiques nécessitent un temps de mise en place initial non négligeable correspondant au calcul des différents termes statistiques présents dans l'expression. Ensuite, à chaque nouvelle évaluation effectuée, une expression mathématique relativement faible en termes de temps d'évaluation est appliquée. Ainsi le temps requis

pour l'évaluation analytique est quasiment constant par rapport au nombre d'itérations du processus d'optimisation.

1.4 Conclusion

Cette première partie a permis de définir l'arithmétique virgule fixe. Celle-ci se traduit par la présence de bruits appelés bruits de quantification dont les modèles ont été présentés. Les bruits de quantification se propagent au sein du système pour aboutir à une erreur globale en sortie de celui-ci. Cette erreur modifie la précision de l'application et nécessite son évaluation. Les différentes méthodes d'évaluation de la précision ont été présentées. Deux types d'approche existent. Tout d'abord les méthodes d'évaluation de la précision par simulation. La qualité de l'application est évaluée en simulant l'application en virgule fixe et en virgule flottante. Cependant, cette approche est très contraignante en terme de temps de simulation. La deuxième approche concerne les méthodes analytiques d'évaluation de la précision. La métrique d'évaluation de la précision est déterminée par une expression mathématique. Cette approche permet de réduire le temps d'optimisation par rapport aux méthodes basées sur la simulation. De ce fait, le modèle développé par la suite est analytique et la métrique utilisée est le RSBQ. Cette métrique est celle représentant le mieux l'influence du bruit dans l'application. Or, les méthodes analytiques sont définies pour des types d'applications particulières comme les systèmes LTI ou bien les systèmes non-LTI non-récursifs. Ces méthodes ne permettent pas de traiter des systèmes non-LTI possédant une récursion comme les filtres adaptatifs. Dans un premier temps, l'objectif est de développer des modèles pour des applications plus complexes comme les filtres adaptatifs. Cette partie est présentée dans le chapitre 2. Dans un second temps, dans un souci de généralisation, le but est de développer un modèle valide pour tout type de système. Ce modèle est étudié dans le chapitre 3.

Chapitre 2

Evaluation de la précision des filtres adaptatifs

Sommaire

L'objectif de ce chapitre est d'analyser les filtres adaptatifs en virgule fixe et de proposer un modèle dédié pour les différents filtres basés sur l'algorithme du gradient. Dans notre cas, la précision est évaluée au moyen du Rapport Signal à Bruit de Quantification (RSBQ). La méthode proposée est une approche analytique pour laquelle aucune simulation virgule fixe n'est requise. L'objectif est de définir une expression analytique de la puissance du bruit en sortie du système considéré. L'expression du RSBQ diffère selon le type de système considéré. Plusieurs méthodes ont été présentées dans le chapitre précédent permettant de traiter les systèmes LTI et des systèmes non-LTI non récursifs. Cependant, les systèmes non-LTI présentant une récursion tels que les filtres adaptatifs, ne peuvent être traités par cette approche. En effet, ce type de système est le plus complexe à étudier. Ainsi, dans ce chapitre, l'expression du bruit en sortie des filtres adaptatifs est présentée. Dans une première partie, une présentation des filtres adaptatifs est développée. Leur implantation en virgule fixe est présentée ainsi que les différentes approches existantes pour étudier leur comportement en précision finie. Dans une seconde partie, l'expression de la puissance du bruit en sortie des filtres adaptatifs est présentée. Pour cette partie, la puissance du bruit en sortie des algorithmes LMS, NLMS, Leaky-LMS ou encore APA est exprimée analytiquement. Finalement, diverses expérimentations évaluent la qualité des modèles proposés. Les termes scalaires sont notés en minuscule $x(n)$, les vecteurs en majuscule $X(n)$ et les matrices en majuscule gras $\mathbf{X}(n)$.

2.1 Modèles existants pour les filtres adaptatifs

Dans cette partie, les filtres adaptatifs sont présentés. Ensuite, l'implantation des différents algorithmes existants en virgule fixe est répertoriée pour déterminer les différents modèles définis. Les filtres adaptatifs [52] sont utilisés dans de nombreux domaines tels que l'annulation d'échos, l'égalisation, la prédiction linéaire et l'évaluation de canal. L'objectif des filtres adaptatifs est d'estimer une séquence désirée $y(n)$ à partir de données observées $X(n)$. Pour ce faire, les données observées sont filtrées par des coefficients $W(n)$ de façon à converger vers la séquence désirée. Cette convergence s'effectue en minimisant l'erreur quadratique $J(n)$ commise entre les données d'entrée filtrées $\widehat{y}(n)$ et la séquence désirée $y(n)$. Cette erreur quadratique $J(n)$ diffère selon le type d'algorithme choisi. Pour les algorithmes du gradient, l'erreur quadratique correspond à la puissance de l'erreur $e(n)$ entre la séquence désirée $y(n)$ et les données d'observation filtrées $W^t(n)X(n)$:

$$J(n) = E[e^2(n)] \tag{2.1}$$

Pour les algorithmes des moindres carrés, l'erreur quadratique s'écrit

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} e^2(k) \tag{2.2}$$

où λ désigne le facteur d'oubli. La mise à jour des coefficients $W(n)$ se fait alors selon l'expression suivante :

$$W(n+1) = W(n) - \frac{\mu}{2}\frac{\partial J(n)}{\partial W(n)} \tag{2.3}$$

où μ désigne le pas d'adaptation. Ces types d'algorithme sont définis dans les parties suivantes.

2.1.1 Algorithmes du gradient

Dans cette partie les algorithmes du gradient ainsi que leur implantation en précision finie est présentée. Pour ce faire, les différents algorithmes dérivant de cette méthode sont étudiés.

Algorithme LMS

L'algorithme Least Mean Square (LMS) est défini dans [52] et [117]. Pour les algorithmes du gradient, la mise à jour des coefficients s'effectue selon l'expression suivante :

$$\begin{aligned} W(n+1) &= W(n) - \frac{\mu}{2}\frac{\partial J(n)}{\partial W(n)} \\ &= W(n) - \frac{\mu}{2}\frac{\partial e^2(n)}{\partial W(n)} \\ &= W(n) - \mu e(n)\frac{\partial e(n)}{\partial W(n)} \\ &= W(n) - \mu e(n)\frac{\partial \left(y(n) - W^t(n)X(n)\right)}{\partial W(n)} \\ &= W(n) + \mu e(n)X(n) \end{aligned} \tag{2.4}$$

La figure 3.22 illustre le graphe de cet algorithme. Les équations du système sont les suivantes :

$$W(n+1) = W(n) + \mu e(n)X(n) \tag{2.5}$$

$$e(n) = y(n) - \widehat{y}(n) \tag{2.6}$$

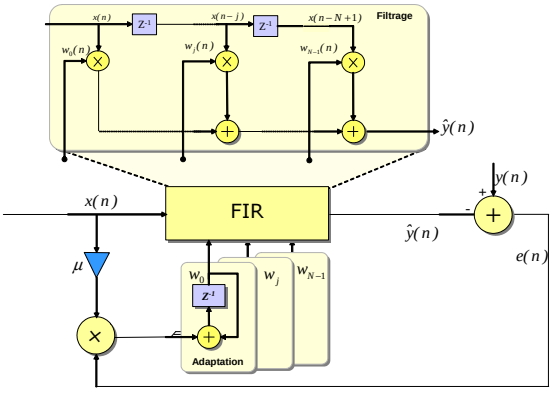

FIG. 2.1 – *Schéma du filtre LMS en précision infinie*

$$\widehat{y}(n) = W^t(n)X(n) \tag{2.7}$$

Les termes $W(n) = [w_0(n)...w_{N-1}(n)]^t$ et $X(n) = [x(n)....x(n-N+1)]^t$ sont des vecteurs de taille N représentant les coefficients du système et le vecteur de données d'entrée. Les termes $e(n)$ et $y(n)$ sont des scalaires. Le fonctionnement de ce système peut être divisé en deux phases. Tout d'abord une phase de convergence pendant laquelle les coefficients $W(n)$ convergent vers la solution optimale W_{opt} définie par

$$W_{opt} = \mathbf{R}_{xx}^{-1} R_{xy} \tag{2.8}$$

où $\mathbf{R}_{xx}$ désigne la matrice d'autocorrélation des données d'entrée $\mathbf{R}_{xx} = E(X(n)X^t(n))$ et R_{xy} représente l'intercorrélation entre les données d'entrée et la valeur désirée $R_{xy} = E(X(n)y(n))$. Le temps de convergence vers ces coefficients optimaux dépend de la valeur du pas d'adaptation μ. Celui-ci doit être compris dans l'intervalle

$$0 < \mu < \frac{2}{Tr(\mathbf{R}_{xx})} \tag{2.9}$$

où Tr désigne l'opérateur $Trace$. La vitesse de convergence est donc liée à la corrélation des données d'entrée $X(n)$. Une étude de cette convergence pour des données d'entrée gaussiennes et non-corrélées est donnée dans [45]. Une autre étude est présentée dans [37] dans laquelle les données d'entrée sont considérées corrélées. Dans ces deux études, les auteurs présentent de nouvelles bornes de μ pour assurer une convergence des coefficients. Pour étudier la convergence, Slock dans [104] utilise un modèle de décomposition des données en les écrivant selon la formule

$$X(n) = s||X||\upsilon \tag{2.10}$$

où s désigne le signe, $||X||$ la norme et υ un vecteur défini par

$$Prob(\upsilon = \nu_i) = \frac{\lambda_i}{Tr(\mathbf{R}_{xx})} \text{ pour } i \in [1:N] \tag{2.11}$$

où ν_i désigne le vecteur propre associé à la valeur propre λ_i de la matrice d'autocorrélation $\mathbf{R}_{xx}$ des données d'entrée. Cette modélisation permet d'étudier la vitesse de convergence de l'algorithme.
La seconde phase correspond à l'état stable du système, une fois la convergence terminée. Lors de cette phase, les coefficients $W(n)$ oscillent autour de leur valeur optimale W_{opt}. Bershad dans [15] montre que la distribution des coefficients $w_i(n)$ est gaussienne centrée sur le coefficient optimal w_{opt_i}. L'écart entre les coefficients et leur valeur optimale W_{opt} est étudié dans [106] pour la variance. Un autre intérêt de l'état stable est d'analyser le comportement de l'erreur quadratique.

Celle-ci est fonction de l'erreur quadratique minimale J_{min} [52]. Dans [107], l'auteur étudie plus en détails sa variance.

Algorithme LMS en précision finie L'implémentation en précision finie de l'algorithme LMS est représentée sur la figure 2.2. Soit $X'(n) = (x'(n)x'(n-1)...x'(n-N+1))^t$ le signal d'entrée quantifié. Il correspond à un vecteur colonne de taille N (où N désigne la taille du filtre) et $x'(n)$ l'échantillon d'entrée à l'instant n. Soit $y'(n)$ le signal désiré quantifié et $\hat{y}'(n)$ le signal quantifié en sortie du filtre. Le terme $W'(n)$ représente le vecteur colonne de taille N des coefficients en précision finie et μ désigne le pas de quantification et est supposé être une puissance de deux.

Les bruits de quantification sont au nombre de quatre. Soit $\alpha(n)$ le bruit de quantification associé au signal d'entrée, $\beta(n)$ celui associé au signal désiré, $\eta(n)$ le bruit à l'intérieur du FIR et $\gamma(n)$ l'ensemble des bruits au niveau de la multiplication. De plus, le terme $\rho(n)$ représentant l'écart entre $W'(n)$ et $W(n)$ est introduit. Il ne peut être considéré comme un bruit de quantification. Ces bruits sont donc définis par les relations suivantes :

$$\begin{aligned} X'(n) &= X(n) + \alpha(n) \\ y'(n) &= y(n) + \beta(n) \\ W'(n) &= W(n) + \rho(n) \end{aligned} \tag{2.12}$$

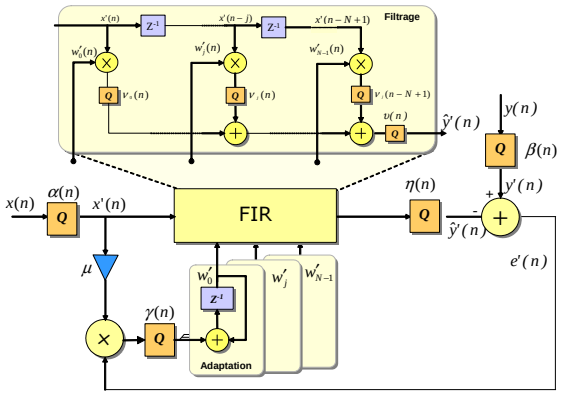

FIG. 2.2 – *Schéma du filtre LMS en précision finie*

En précision finie, les équations régissant le système, et introduisant les bruits $\gamma(n)$ et $\eta(n)$ sont définies par :

$$W'(n+1) = W'(n) + \mu e'(n)X'(n) + \gamma(n) \tag{2.13}$$

$$e'(n) = y'(n) - \widehat{y'}(n) \tag{2.14}$$

$$\widehat{y'}(n) = W'^t(n)X'(n) + \eta(n) \tag{2.15}$$

Convergence de l'algorithme L'implémentation de l'algorithme doit permettre d'assurer la convergence du filtre. Pour ce faire, il est nécessaire d'allouer un nombre de bits suffisant au codage des coefficients. Pour que la convergence du filtre soit assurée, la mise à jour de ceux-ci doit être satisfaite. Le produit $\mu e'(n)X'(n)$ assurant la mise à jour des coefficients ne doit pas être nul lors de la phase de convergence. Il doit être supérieur au pas de quantification du format appliqué aux coefficients [10], [19] et [51]. Le signal sortant de la conversion en sortie de la multiplication (lié

au bruit $\gamma(n)$) ne doit donc pas être nul. L'amplitude de ce signal doit être supérieure au pas de quantification q lié à la conversion.

$$\mu X'(n)e'(n) > q \tag{2.16}$$

Or $X'(n)$ et $e'(n)$ peuvent être approchés par leur valeur en précision infinie soit :

$$\mu X(n)e(n) > q \tag{2.17}$$

Puis, en supposant l'indépendance entre le signal d'entrée $X(n)$ et l'erreur $e(n)$, et en passant à la puissance, il vient :

$$\mu^2\sigma_x^2\xi > q^2 \tag{2.18}$$

où σ_x^2 représente la puissance du signal d'entrée et ξ, l'erreur quadratique moyenne. Cette équation permet de déterminer la valeur maximale du pas de quantification q assurant une bonne convergence des coefficients. Cette valeur du pas de quantification q fournit le nombre de bits minimal du format des coefficients assurant une convergence correcte de l'algorithme.

Calcul de l'erreur quadratique Une fois la convergence de l'algorithme assurée, un critère de qualité utilisé est l'erreur quadratique. Elle caractérise l'écart entre les coefficients et leur valeur optimale. Dans cette partie, l'erreur quadratique moyenne en précision finie est considérée. De nombreuses études ont été effectuées [21, 99, 64, 35, 52, 47, 46, 49, 3] mais trois en particulier ont retenu notre attention. La première est celle développée par Caraiscos dans [19]. La seconde est celle proposée par Bellanger dans [10]. Bien que la façon d'approcher le problème soit différente, les résultats obtenus sont très proches. Ces modèles sont présentés dans le cadre d'une loi de quantification par arrondi. Cependant, comme dans les deux modèles les moyennes des bruits de quantification sont considérées comme nulles, les modèles sont en réalité proposés pour le cas de l'arrondi convergent. Le dernier modèle est celui proposé par Bershad dans [16]. Il utilise une décomposition du bruit en série de Fourier puis étudie l'erreur quadratique à partir de cette décomposition.

Modèle proposé par C.Caraiscos dans [19] Ce modèle présenté dans [19], étudie l'influence de la virgule fixe sur l'erreur quadratique moyenne de l'algorithme LMS en précision finie. Les bruits de quantification générés par la virgule fixe entraînent une différence sur l'erreur quadratique moyenne par rapport à la précision infinie. L'objectif est de déterminer cette nouvelle erreur quadratique. L'étude est faite uniquement dans le cadre d'une quantification par arrondi convergent car la moyenne de tous les bruits est nulle. Pour simplifier, les notations précédentes au niveau du schéma 2.2 sont reprises. L'erreur réelle en sortie du filtre, à savoir l'écart entre le signal désiré et le signal en sortie du filtre quantifiée est donnée par la relation suivante :

$$e'(n) = y(n) - \hat{y}'(n) \tag{2.19}$$

Or en injectant les équations (2.15), et (2.12) dans l'expression précédente et en négligeant le terme croisé de bruits $\rho^t(n)\alpha(n)$, il vient:

$$e'(n) = y(n) - W^t(n)X(n) - \rho^t(n)X(n) - \alpha^t(n)W(n) - \eta(n) \tag{2.20}$$

Le terme $y(n) - W^t(n)X(n)$ correspond à l'erreur $e(n)$ en précision infinie. Ainsi l'équation (2.20) peut être réécrite de la manière suivante :

$$e'(n) = e(n) - \rho^t(n)X(n) - \alpha^t(n)W(n) - \eta(n) \tag{2.21}$$

Soient $\xi' = E(e'(n)^2)$ et $\xi = E(e(n)^2)$ les erreurs quadratiques moyennes en précision finie et en précision infinie. Les termes croisés sont nuls puisque les moyennes de $\alpha(n)$, $\rho(n)$ et de

$\eta(n)$ sont considérées nulles, et ces termes sont non-corrélés. L'erreur quadratique moyenne totale conduit à l'expression suivante :

$$\xi' = \xi + E(\rho^t(n)X(n))^2 + E(\alpha^t(n)W(n))^2 + E(\eta(n))^2 \tag{2.22}$$

Soit $\mathbf{R}_{xx}$ la matrice d'autocorrélation du signal d'entrée et J_{min} l'erreur quadratique minimale obtenue avec les coefficients optimaux. D'après [72], l'erreur quadratique en précision infinie est égale à :

$$\xi = \frac{2J_{min}}{2 - \mu Tr(\mathbf{R}_{xx})} \tag{2.23}$$

L'erreur quadratique moyenne totale est alors la somme de l'erreur quadratique moyenne en précision infinie et de trois termes correspondant aux trois termes de bruits précédents. L'étude proposée dans [19] analyse ensuite chacun des trois termes. Pour le premier terme, le calcul est le suivant :

$$E(W^t(n)\alpha(n))^2 = E(W^t(n)W(n))\sigma_\alpha^2 = \left(E(W_{opt}^t W_{opt}) + E(\phi^t(n)\phi(n))\right)\sigma_\alpha^2 \tag{2.24}$$

où $\phi(n)$ désigne l'écart entre les coefficients $W(n)$ et leur valeur optimale W_{opt}. L'auteur fournit dans [117] l'égalité suivante :

$$E(\phi^t(n)\phi(n)) = \frac{N\mu J_{min}}{2} \tag{2.25}$$

Finalement, avec la notation suivante $\|W\|^2 = E(W^tW)$, le calcul du premier bruit donne l'expression suivante :

$$E(\alpha^t(n)W(n))^2 = (\|W_{opt}\|^2 + \frac{1}{2}\mu J_{min}N)\sigma_\alpha^2 \tag{2.26}$$

Pour le terme $E(\alpha^t(n)W(n))^2$, le développement proposé dans [19] conduit à l'équation suivante :

$$\rho(n+1) = \mathbf{F}(n)\rho(n) + B(n) \tag{2.27}$$

où

$$\mathbf{F}(n) = \mathbf{I_N} - \mu X(n)X^t(n) \tag{2.28}$$

$$B(n) - \mu X(n)W^t(n)\alpha(n) + \mu X(n)(\beta(n) - \eta(n)) + \mu\alpha(n)e(n) + \gamma(n) \tag{2.29}$$

En introduisant $\mathbf{P}(n) = E(\rho(n)\rho^t(n))$, les termes croisés n'apparaissent plus puisque la quantification est par arrondi convergent (les moyennes des bruits sont nulles). Il vient alors :

$$\mathbf{P}(n+1) = E(\mathbf{F}(n)\mathbf{P}(n)\mathbf{F}(n)) + \mathbf{Q}(n) \tag{2.30}$$

avec $\mathbf{Q}(n) = \sigma_\gamma^2\mathbf{I_N} + \mu^2\|W_{opt}\|^2\sigma_\alpha^2\mathbf{R}_{xx} + \mu^2\sigma_\alpha^2\xi_{\min}\mathbf{I_N} + \mu^2(\sigma_\beta^2 + \sigma_\eta^2)\mathbf{R}_{xx}$.

Ensuite, le développement de l'expression $E(\mathbf{F}(n)\mathbf{P}(n)\mathbf{F}(n))$ utilise les propriétés présentées dans [56] pour aboutir à l'équation suivante :

$$\mathbf{P}(n+1) = \mathbf{P}(n) - \mu(\mathbf{R}_{xx}\mathbf{P}(n) + \mathbf{P}(n)\mathbf{R}_{xx}) + \mu^2\mathbf{R}_{xx}Tr(\mathbf{R}_{xx}\mathbf{P}(n)) + \mathbf{Q}(n) \tag{2.31}$$

Finalement, le développement conduit à l'équation :

$$E(\rho^t(n)X(n))^2 = \frac{N\sigma_\gamma^2 + \mu^2(\|W_{opt}\|^2\sigma_\alpha^2 + \sigma_\beta^2 + \sigma_\eta^2)Tr(\mathbf{R}_{xx}) + N\mu^2\sigma_\alpha^2 J_{min}}{2\mu - \mu^2 Tr(\mathbf{R}_{xx})} \tag{2.32}$$

La puissance du dernier terme $\eta(n)$ est la suivante :

$$E(\eta^2(n)) = \sigma_\eta^2 \tag{2.33}$$

Le résultat global de l'expression de l'erreur quadratique moyenne en précision finie ξ' est obtenue en sommant les expressions (2.26), (2.32) et (2.33).

$$\begin{aligned}\xi' = J_{min} &+ \frac{\mu J_{min} Tr(\mathbf{R}_{xx})}{2 - \mu Tr(\mathbf{R}_{xx})} + (\|W_{opt}\|^2 + \frac{1}{2}\mu J_{min} N)\sigma_\alpha^2 \\ &+ \frac{N\sigma_\gamma^2 + \mu^2(\|W_{opt}\|^2\sigma_\alpha^2 + \sigma_\beta^2 + \sigma_\eta^2)Tr(\mathbf{R}_{xx}) + N\mu^2\sigma_\alpha^2 J_{min}}{2\mu - \mu^2 Tr(\mathbf{R}_{xx})} + \sigma_\eta^2\end{aligned} \tag{2.34}$$

Ainsi, comme l'erreur quadratique moyenne est la somme de l'erreur quadratique en précision infinie et de la puissance du bruit, alors la puissance totale du bruit est égale à

$$\begin{aligned}Pb = &(\|W_{opt}\|^2 + \frac{1}{2}\mu J_{min} N)\sigma_\alpha^2 + \sigma_\eta^2 \\ &+ \frac{N\sigma_\gamma^2 + \mu^2(\|W_{opt}\|^2\sigma_\alpha^2 + \sigma_\beta^2 + \sigma_\eta^2)Tr(\mathbf{R}_{xx}) + N\mu^2\sigma_\alpha^2 J_{min}}{2\mu - \mu^2 Tr(\mathbf{R}_{xx})}\end{aligned} \tag{2.35}$$

Cette expression de la puissance du bruit en sortie du système est assez complexe. L'erreur quadratique minimale J_{min} doit par exemple être connue. De plus, certains termes contenus dans cette expression pourraient être négligés sans réduire la qualité de l'évaluation.

Modèle proposé par M.Bellanger dans [10]

Ce modèle proposé dans [10] utilise une approche différente. Comme précédemment, la loi utilisée est l'arrondi convergent. Cependant, le bruit sur le signal d'entrée n'est pas pris en compte ($\alpha(n) = 0\ \forall n$). Les mêmes notations sont reprises dans un souci de clarté. La matrice $\mathbf{R}_{xx}$ est la matrice d'autocorrélation du signal d'entrée. Elle est symétrique donc diagonalisable. Il existe une matrice $\mathbf{M}$ et une matrice $\mathbf{D}$ diagonale telle que : $\mathbf{R}_{xx} = \mathbf{MDM^t}$. Soit $\theta(n)$ l'écart entre les coefficients et les coefficients optimaux dans l'espace transformé en précision infinie.

$$\theta(n) = \mathbf{M}[W_{opt} - W(n)] \tag{2.36}$$

D'après [10] les équations suivantes sont obtenues :

$$\theta(n+1) = \theta(n) - \mu\mathbf{M}X(n)e(n) \tag{2.37}$$

$$E\left(\theta(n+1)\theta^t(n+1)\right) = [\mathbf{I_N} - 2\mu D]E\left(\theta(n)\theta^t(n)\right) + \mu^2 E(e^2(n))\mathbf{D} \tag{2.38}$$

$$E(e^2(n)) = J_{min} + \Lambda^t E(\theta^2(n)) \tag{2.39}$$

où Λ désigne le vecteur colonne des valeurs propres λ_i de $\mathbf{D}$ et $E(\theta^2(n)) = [\theta_1^2(n),\theta_2^2(n),...,\theta_N^2(n)]^t$, le vecteur colonne contenant les éléments de $\theta(n)$ au carré.

Dans le cadre d'une implémentation en virgule fixe apparaissent alors deux bruits de quantification au sein du système : $\eta(n)$ et $\gamma(n)$ tel que représenté sur la figure 2.2. Les bruits $\alpha(n)$ et $\beta(n)$ n'existent pas car ils ne sont pas pris en compte. Alors les équations (2.37), (2.38) et (2.39) deviennent :

$$\theta'(n+1) = \theta'(n) - \mu \mathbf{M} X(n) e'(n) + \gamma(n) \tag{2.40}$$

$$E\left(\theta'(n+1)\theta'^{t}(n+1)\right) = [\mathbf{I_N} - 2\mu\mathbf{D}]E\left(\theta'(n)\theta'^{t}(n)\right) + \mu^2 E(e'^2(n))\mathbf{D} + \sigma_\gamma^2 \mathbf{I_N} \tag{2.41}$$

$$E(e'^2(n)) = J_{min} + \Lambda^t E(\theta'^2(n)) + \sigma_\eta^2 \tag{2.42}$$

A l'état stable, en notant $\xi' = E(e'^2(\infty))$, les équations (2.41) et (2.42) s'écrivent :

$$2\mu\mathbf{D}E(\theta'_\infty \theta'^t_\infty) = \mu^2 \xi' \mathbf{D} + \sigma_\gamma^2 \mathbf{I_N} \tag{2.43}$$

$$\xi' = J_{min} + \Lambda^t E(\theta'^2_\infty) + \sigma_\eta^2 \tag{2.44}$$

Les éléments composant θ sont non-corrélés et ont même puissance. En prenant la trace de l'expression (2.43), et en notant $\|\theta'^2_\infty\|$ la puissance d'un élément du vecteur $E(\theta'^2_\infty)$, il vient :

$$2\mu\|\theta'^2_\infty\| Tr(\mathbf{R}_{xx}) = \mu^2 \xi' Tr(\mathbf{R}_{xx}) + N\sigma_\gamma^2 \tag{2.45}$$

D'autre part, l'équation (2.44) peut être réécrite sous la forme

$$\xi' = J_{min} + \|\theta'^2_\infty\| Tr(\mathbf{R}_{xx}) + \sigma_\eta^2 \tag{2.46}$$

Puis en l'injectant l'équation (2.45) dans (2.46), cela mène finalement à l'expression suivante :

$$\xi' = \frac{1}{1 - \frac{\mu}{2} Tr(\mathbf{R}_{xx})} (J_{min} + \frac{N\sigma_\gamma^2}{2\mu} + \sigma_\eta^2) \tag{2.47}$$

L'équation correspond à la nouvelle valeur de l'erreur quadratique moyenne comme dans l'étude précédente. La première partie correspond à l'erreur quadratique en précision infinie et les autres termes correspondent à la puissance du bruit à l'intérieur du système, d'où :

$$Pb = \frac{1}{1 - \frac{\mu}{2} Tr(\mathbf{R}_{xx})} \left(\frac{N\sigma_\gamma^2}{2\mu} + \sigma_\eta^2 \right) \tag{2.48}$$

L'équation obtenue semble différente de celle obtenue par la méthode présentée dans [19]. En fait, les résultats sont identiques. L'équation (2.34) est reprise en y enlevant tous les termes en α et β. Il vient alors :

$$\xi' = J_{min} + \frac{\mu J_{min} Tr(\mathbf{R}_{xx})}{2 - \mu Tr(\mathbf{R}_{xx})} + \frac{N\sigma_\gamma^2 + \mu^2 \sigma_\eta^2 Tr(\mathbf{R}_{xx})}{2\mu - \mu^2 Tr(\mathbf{R}_{xx})} + \sigma_\eta^2 \tag{2.49}$$

Puis en remettant au même dénominateur et en simplifiant par 2μ cela aboutit à :

$$\xi' = \frac{1}{1 - \frac{\mu}{2} Tr(\mathbf{R}_{xx})} (J_{min} + \frac{N\sigma_\gamma^2}{2\mu} + \sigma_\eta^2) \tag{2.50}$$

Malgré des approches différentes, ces deux méthodes conduisent donc à des résultats identiques.

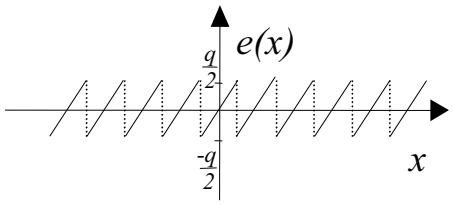

FIG. 2.3 – *Représentation du bruit en fonction du signal*

Modèle basé sur une décomposition en série de Fourier [16] Ce modèle utilise une décomposition en série de Fourier du bruit. En effet, le bruit e est périodique en fonction du signal x auquel il est associé comme le montre la figure 2.3. A partir de là, une décomposition du bruit en série de Fourier est faite. Puis une étude sur le désajustement (écart normalisé entre les coefficients optimaux et les coefficients en précision finie) est effectuée. Celle-ci est réalisée pour le cas d'un pas d'adaptation quelconque dans [17] ou une puissance de 2 dans [16]. Elle est également développée dans le cas de données d'entrée gaussiennes [18]. Bien que l'approche soit innovante, les équations obtenues sont très complexes surtout par rapport à celles développées par les deux modèles précédents.

Les modèles présentés précédemment ont le principal désavantage de n'être élaborés que pour la loi d'arrondi convergent. Leur utilisation est donc limitée. Néanmoins, la seconde approche fournit une expression plus simple à exploiter.

Algorithme NLMS

L'algorithme NLMS (Normalized Least Mean Square) est dérivé du LMS. Cet algorithme présenté dans [52] normalise les données d'entrées si bien que la convergence est assurée pour une valeur du pas d'adaptation comprise entre 0 et 2

$$0 < \mu < 2 \tag{2.51}$$

L'équation de mise à jour des coefficients diffère du LMS et est égale à

$$W(n+1) = W(n) + \mu e(n) \frac{X(n)}{X^t(n)X(n)} \tag{2.52}$$

Le comportement de cet algorithme est étudié dans [14] avec la restriction de données d'entrée gaussiennes. De la même façon, Slock dans [104] utilise sa modélisation dans le cadre du NLMS pour étudier sa convergence. Dans [110], les auteurs démontrent que la convergence du NLMS est plus rapide que celle du LMS pour le pas d'adaptation μ optimal dans les deux cas. Par contre, le désajustement de l'erreur quadratique est plus important dans le cas du NLMS.

L'implantation de l'algorithme NLMS en précision finie est présentée sur la figure 2.8.

La mise à jour des coefficients est la suivante :

$$W'(n+1) = W'(n) + \mu e'(n) \frac{X'(n)}{X'^t(n)X'(n)} + \gamma(n) \tag{2.53}$$

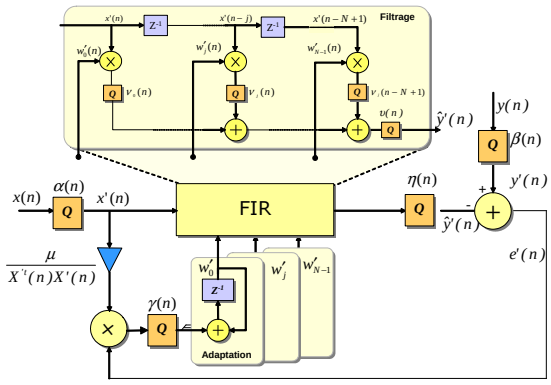

FIG. 2.4 – *Schéma du filtre NLMS en virgule fixe*

Convergence de l'algorithme NLMS Comme dans le cas de l'algorithme LMS, la convergence est assurée si le terme de mise à jour des coefficients est supérieur au pas de quantification du codage lié aux coefficients [20]. L'équation assurant une convergence correcte de l'algorithme est la suivante :

$$\mu e(n)\frac{x(n)}{X^t(n)X(n)} > q \tag{2.54}$$

En passant à la puissance, en utilisant l'approximation $E(\frac{1}{X^t(n)X(n)}) = \frac{1}{Tr(\mathbf{R}_{xx})}$ fournie par Slock dans [104] et en considérant l'indépendance entre le signal d'entrée $X(n)$ et l'erreur $e(n)$, il vient :

$$\mu^2\xi\frac{\sigma_x^2}{Tr(R)} > q^2 \tag{2.55}$$

Comme dans le cas du LMS, le nombre de bits minimum assurant une convergence correcte de l'algorithme est ainsi déterminée.

Erreur quadratique moyenne l'algorithme NLMS en virgule fixe Un modèle est proposé par Chang dans [20]. Ce modèle s'intéresse à l'erreur quadratique moyenne de l'algorithme NLMS en précision finie. Pour cela, une méthodologie identique à celle présentée dans [19] est utilisée. L'erreur quadratique en virgule fixe ξ' est dans ce cas la somme de l'erreur quadratique en précision infinie et de la puissance du bruit.

$$\begin{aligned}\xi' = & \frac{J_{min}}{1-\mu\sum_{i=1}^{N}\frac{\lambda_i}{2Tr(\mathbf{R}_{xx})-\mu\lambda_i}} \\ & + (\|W_{opt}\|^2 + \frac{\mu J_{min}}{1-\mu\sum_{i=1}^{N}\frac{\lambda_i}{2Tr(R)-\mu\lambda_i}}\sum_{i=1}^{N}\frac{1}{2Tr(\mathbf{R}_{xx})-\mu\lambda_i})\sigma_\alpha^2 \\ & + \frac{N\sigma_\gamma^2+\mu^2E(f(X(n))^2)(\|W_{opt}\|^2\sigma_\alpha^2+\sigma_\beta^2+\sigma_\eta^2)Tr(\mathbf{R}_{xx})+N\mu^2E(f(X(n))^2)\sigma_\alpha^2J_{min}}{2\mu E(f(X(n))^2)-\mu^2E(f(X(n))^2)Tr(\mathbf{R}_{xx})} + \sigma_\eta^2\end{aligned} \tag{2.56}$$

où $f(X(n)) = \frac{1}{X^t(n)X(n)}$ correspond à la fonction de normalisation. Cette equation est complexe à mettre en œuvre mais donne des résultats satisfaisants en terme de qualité de l'estimation de la puissance du bruit [20].

Autres algorithmes transversaux du gradient

Pour prévenir les problèmes de stabilité occasionnés par la virgule fixe, un nouveau type d'algorithme a été créé correspondant à l'algorithme Leaky-LMS [71]. La seule différence avec l'algorithme LMS vient de la mise à jour des coefficients effectuée à l'aide de l'équation suivante :

$$W(n+1) = \lambda W(n) + \mu e(n) X(n) \tag{2.57}$$

où λ est une constante comprise entre 0 et 1. Son implementation est étudiée en [38] et l'influence de la virgule fixe est vue en [120]. Ce modèle se base sur la conservation de l'énergie. Il fournit une étude de l'erreur quadratique a posteriori. Cette étude généralisable à tous les algorithmes du gradient reste néanmoins complexe en termes mathématiques.

Un autre algorithme utilisé est l'algorithme du signe [13], pour lequel la mise à jour des coefficients est faite en prenant le signe de l'erreur. Cette variante de l'algorithme LMS est plus simple à mettre en œuvre mais la vitesse de convergence est plus importante. Deux études sur son comportement sont faites en [69] et [41]. Une autre est effectuée en [31] où une modélisation non-linéaire de l'opérateur signe est utilisée. Une étude de l'influence en précision finie est présentée en [101] dans laquelle une analyse de l'opérateur 'signe' est effectuée. Cet algorithme présente l'intérêt de supprimer la multiplication au niveau de la mise à jour des coefficients mais la convergence est beaucoup plus lente. Une autre variante appelé sign-sign algorithm existe [39],[40]. Pour cet algorithme, la mise à jour est faite en prenant également le signe des données d'entrées simplifiant encore son implémentation. La complexité est encore réduite mais le temps de convergence en est rallongé.

Les signaux non-stationnaires rendent les algorithmes précédents moins robustes. Pour résoudre ce problème, l'algorithme Least-Mean-Fourth (LMF), dans lequel la fonction à minimiser est le moment d'ordre 4 de l'erreur, a été créé [119].

$$J(n) = e^4(n) \tag{2.58}$$

Cet algorithme est beaucoup plus intéressant pour les signaux non-stationnaires en terme de convergence. Néanmoins les études analytiques concernant cet algorithme sont difficiles à mettre en œuvre.

Un autre algorithme est le Constant Modulus Algorithm (CMA) [48] où la fonction de coût est différente de celle rencontrée dans les algorithmes LMS ou NLMS. Celui-ci est introduit car il est plus robuste que le LMS dans le cadre d'une égalisation aveugle. Son critère d'erreur est égal à $[\gamma - |y(n)^2|]$. L'implantation de cet algorithme est étudié dans [53].

Filtres en treillis

Ce type de filtres basés sur la récurrence d'ordre sont définis en [52]. Ces filtres peuvent être basés soit sur la méthode du gradient soit sur celle des moindres carrés. Pour les algorithmes basés sur la méthode du gradient, une étude de leur comportement en précision infinie est donnée dans [105] et [55]. Leur implémentation est souvent utilisée dans des domaines tels que l'annulation de bruit ou l'égalisation [94]. Leur implementation en précision finie est étudiée dans [70] et [95]. Ces études montrent que l'erreur quadratique en précision finie est inversement proportionnelle au paramètre de convergence définissant la mise à jour du coefficient entre deux états. De plus, les signaux corrélés introduisent plus de bruits que les signaux faiblement corrélés.

2.1.2 Algorithmes des moindres carrés

Dans cette partie, l'étude des algorithmes des moindres carrés est effectuée. L'erreur quadratique à minimiser est donnée par l'expression suivante :

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} e^2(k) \tag{2.59}$$

L'algorithme le plus utilisé est l'algorithme RLS (Recursive Least Square) étudié dans la partie suivante.

Algorithme RLS

L'algorithme RLS utilise une inversion de matrice implémentée de manière récursive, d'où son nom. Il est défini dans [52]. Son implantation en précision finie est représentée sur la figure 2.5.

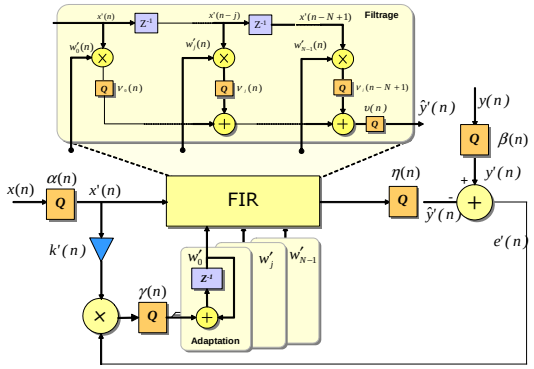

FIG. 2.5 – *Schéma du filtre RLS en virgule fixe*

Ses équations en précision finie sont les suivantes :

$$\begin{aligned} W'(n+1) &= W'(n) + k'(n)e'(n)X'(n) + \delta(n)e'(n) + \gamma(n) \\ e'(n) &= y'(n) - \widehat{y'}(n) \\ \widehat{y'}(n) &= W'^{t}(n)X'(n) + \eta(n) \end{aligned} \tag{2.60}$$

avec $\delta(n)$ le bruit généré par la multiplication entre les données d'entrée $X'(n)$ et le vecteur gain $k'(n)$, décomposé de la façon suivante :

$$\begin{aligned} k'(n) &= \frac{\lambda^{-1}\mathbf{P}'(n-1)}{1 + \lambda^{-1}X'^{t}(n)\mathbf{P}'(n-1)X'(n)} \\ \mathbf{P}'(n) &= \lambda^{-1}\mathbf{P}'(n-1) - \lambda^{-1}k'(n)X'^{t}(n)\mathbf{P}'(n-1) \end{aligned} \tag{2.61}$$

Le terme $\mathbf{P'(n)}$, calculé récursivement, désigne l'inverse de la matrice d'autocorrélation des données d'entrée et λ désigne la facteur d'oubli de valeur positive inférieure ou égale à 1.

Ainsi, il est clair que la principale difficulté vient de la détermination du vecteur $k'(n)$ plus complexe que dans le cas des algorithmes du gradient. Dans [65], l'auteur étudie l'accumulation des erreurs d'arrondi non-linéaires. Cette analyse permet de fournir les bornes de l'erreur et déterminent les conditions sur celles-ci assurant la stabilité du système.

Convergence de l'algorithme RLS Pour assurer le bon fonctionnement de l'algorithme, il est nécessaire d'avoir une mise à jour correcte des coefficients. D'après [4], une évolution correcte des coefficients est obtenue si :

$$k'(n)e'(n)x'(n) > q \tag{2.62}$$

où q est le pas de quantification du codage lié aux coefficients. En passant à la puissance, et en supposant l'indépendance entre l'erreur $e(n)$ et le gain $k(n)$, le résultat suivant est obtenu :

$$E(k(n)^2)\xi\sigma_x^2 > q^2 \tag{2.63}$$

Cette équation permet de déterminer le nombre minimal de bits assurant une mise à jour des coefficients.

Erreur quadratique moyenne de l'algorithme RLS Le calcul de l'erreur quadratique moyenne dans le cadre d'une implémentation en précision finie est proposé dans [4], [1] et [2]. Ce modèle ne prend pas en compte le bruit sur le signal d'entrée ($X'(n) = X(n)$) et tous les bruits sont considérés comme étant blancs et centrés. Soit $\phi(n)$ l'écart entre la valeur courante des coefficients $W'(n)$ et leur valeur optimale W_{opt}.

$$\phi(n) = W'(n) - W_{opt} \tag{2.64}$$

Les relations de récurrence suivantes sont obtenues :

$$\begin{aligned} \phi(n) &= (\mathbf{I_N} - k'(n)X(n)X^t(n))\phi(n-1) + \psi(n) \\ \psi(n) &= -k'(n)X(n)\eta(n) + \gamma(n) \\ e'(n) &= X^t(n)\phi(n) - \eta(n) \end{aligned} \tag{2.65}$$

Ainsi, l'erreur quadratique moyenne ξ' est donnée par

$$\begin{aligned} \xi' &= E(e(n)^{'2}) \\ &= E\left(X^t(n)\phi(n)\phi^t(n)X(n)\right) + \sigma_\eta^2 \\ &= Tr(\mathbf{R}_{xx}\mathbf{R}_\phi) + \sigma_\eta^2 \end{aligned} \tag{2.66}$$

où $\mathbf{R}_\phi = E(\phi(n)\phi^t(n))$. Si le signal d'entrée est blanc, en notant $Tr(\mathbf{R}_\phi) = E(\|\phi(n)\|^2)$ alors cela aboutit à l'expression suivante :

$$\xi' = \sigma_x^2 E(\|\phi(n)\|^2) + \sigma_\eta^2 \tag{2.67}$$

Ensuite, une étude est faite pour déterminer $E(\|\phi(n)\|^2)$ et conduit à l'expression suivante :

$$E(\|\phi(n)\|^2) = \frac{\sigma_\delta^2}{2(1-\lambda)}N + \frac{1}{2}N(1-\lambda)\frac{\sigma_\eta^2}{\sigma_x^2} \tag{2.68}$$

où σ_δ^2 désigne la variance du bruit associé à $k'(n)$. La difficulté est de déterminer cette valeur. Pour tenter d'y répondre, une approche est proposée dans [52]. Ce modèle permet de donner l'expression de l'erreur quadratique moyenne de l'algorithme RLS en précision finie mais valable uniquement pour la loi de quantification par arrondi convergent.

Filtres en treillis

Les filtres en treillis des moindres carrés sont définis en [52] et [98]. De plus, il existe de nombreuses variantes de ce type de filtre [42]. Ceux-ci ont une structure identique aux filtres du gradient mais l'algorithme est différent. Une étude de leur implémentation en précision finie est proposée dans [92]. Elle étudie la propagation des bruits dans le système ainsi que l'influence de chacun d'eux. Elle illustre le fait que le bruit augmente quand le facteur d'oubli tend vers 1. Néanmoins, le cas où le facteur d'oubli est égal à 1 n'est pas pris en compte.

Autres filtres

D'autres études existent pour les algorithmes particuliers des moindres carrés. Pour accélérer la convergence de ces algorithmes, des versions rapides ont été introduites [25]. Dans [12], l'auteur étudie la propagation des erreurs dans le système et leur influence sur sa stabilité. Une structure de l'architecture en est déduite dans [121] assurant l'absence de débordement. Dans [11], l'auteur développe une réalisation complète de la version rapide QR-RLS. Cette version se base sur une décomposition QR utilisant des rotations. Son implantation en précision finie est présentée dans [36]. Néanmoins, les calculs sont très complexes et le papier est difficilement accessible.

Une autre version rapide est le Scaled Tangent Rotations RLS (STAR-RLS). Dans [84], l'influence de la précision finie et de la précision infinie sur cette variante est étudiée. Cependant, les calculs sont difficilement accessibles.

2.1.3 Conclusion

Dans cette partie ont été présentés les différents algorithmes adaptatifs existants ainsi que leurs études en précision finie. D'une manière plus générale, dans [24], l'auteur regarde l'influence de la précision finie sur les filtres adaptatifs de manière générale notamment en termes de stabilité. Des contraintes sont fixées pour assurer la stabilité des systèmes. L'influence de la précision finie est très importante sur les systèmes et des études sur la convergence et l'erreur quadratique sont faites pour la plupart des filtres adaptatifs. Cependant, les différents modèles existants ne prennent pas en compte la moyenne des bruits générés. Or lors d'une quantification par troncature, cette moyenne n'est pas nulle. L'utilisation de ces modèles est donc limitée.

2.2 Proposition de nouveaux modèles pour les algorithmes du gradient

Dans cette partie, l'expression analytique de la puissance du bruit en sortie des filtres adaptatifs est présentée. Celle-ci est définie pour les algorithmes du gradient introduits dans la partie précédente.

2.2.1 Algorithme LMS

Dans cette section, le calcul de la puissance du bruit en sortie de l'algorithme LMS est présenté. Ainsi, pouvoir déterminer de manière analytique la puissance du bruit de quantification en sortie du système permet d'analyser le comportement de l'algorithme implémenté en virgule fixe [88]. Le filtre LMS est présenté sur la figure 2.6. Le bruit en sortie du filtre défini par l'écart entre la sortie du filtre en précision finie et celle en précision infinie est étudié. La loi de quantification est quelconque (troncature ou arrondi) afin d'étudier le cas le plus général. Une étude de l'influence de la précision finie sur la convergence a été présentée dans la partie précédente. Ces approches ne sont valables que pour une loi de quantification par arrondi convergent. L'étude développée ci-dessous est valide pour toutes les lois de quantification. Cette étude est effectuée à l'état stable du filtre. En effet, la sortie du filtre est exploitable une fois la convergence des coefficients effectuée.

Le bruit de quantification $b_y(n)$ en sortie du système correspond à la différence entre la sortie du système $\hat{y}'(n)$ en précision finie et $\hat{y}(n)$ celle en précision infinie :

$$b_y(n) = \hat{y}'(n) - \hat{y}(n) \tag{2.69}$$

Soit $X(n) = [x(n)x(n-1)...x(n-N+1)]^t$ le vecteur de taille N des données d'entrée et $\alpha(n)$ son vecteur bruit associé. Le terme $W(n) = [w_0(n)....w_{N-1}(n)]^t$ désigne le vecteur de coefficients et le vecteur $\rho(n)$ représente l'écart entre les coefficients $W'(n)$ en précision finie et $W(n)$ ceux en précision infinie. Le bruit $\eta(n)$ fait référence à l'ensemble des bruits générés dans le FIR. Il est lié

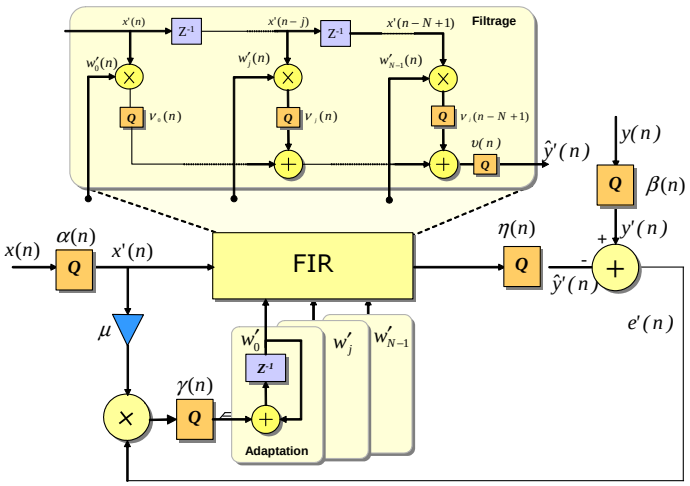

FIG. 2.6 – *Filtre LMS en précision finie*

à tous les bruits générés dans le filtre. En développant de la même façon que dans la partie 2.1.1, le bruit de sortie $b_y(n)$ s'écrit alors sous la forme :

$$b_y(n) = \rho^t(n)X(n) + \alpha^t(n)W(n) + \eta(n) \tag{2.70}$$

L'objectif étant de déterminer la puissance du bruit, le moment d'ordre deux du bruit b_y est calculé. Les termes croisés, correspondant au produit de deux termes de bruits, peuvent être négligés car très faibles devant les autres termes. L'expression suivante est obtenue.

$$E(b_y^2(n)) = E(\alpha^t(n)W(n))^2 + E(\rho^t(n)X(n))^2 + E(\eta(n))^2 \tag{2.71}$$

Le bruit total est alors la somme de trois bruits dont les expressions sont déterminées dans les parties suivantes.

Calcul du terme $E(\alpha^t(n)W(n))^2$

Le premier terme $E(\alpha^t(n)W(n))^2$ correspond au filtrage du bruit de quantification associé à l'entrée. Soit $\mathbf{W}(n) = E(W(n)W^t(n))$ et $\mathbf{\Gamma}(n) = E(\alpha(n)\alpha^t(n))$ deux matrices carrées de taille N représentant les autocorrélations des coefficients $W(n)$ et du bruit d'entrée $\alpha(n)$. En notant $w_i(n)$ le $i^{\text{ème}}$ coefficient, la trace du produit de ces deux matrices aboutit au terme cherché comme le montre l'équation (2.72).

$$\begin{aligned} Tr(\mathbf{W}(n)\mathbf{\Gamma}(n)) &= \sum_{i=1}^{N}(\mathbf{W}(n)\mathbf{\Gamma}(n))_{ii} \\ &= E\left(\sum_{i=1}^{N}\sum_{k=1}^{N}(w_i(n)w_k(n)\alpha_i(n)\alpha_k(n))\right) \\ &= E\left(\sum_{i=1}^{N}(w_i(n)\alpha_i(n))\sum_{k=1}^{N}(w_k(n)\alpha_k(n))\right) \\ &= E\left(\sum_{i=1}^{N}(w_i(n)\alpha_i(n))^2\right) \\ &= E(\alpha^t(n)W(n))^2 \end{aligned} \tag{2.72}$$

Pour déterminer la valeur du terme $E(\alpha^t(n)W(n))^2$, le produit des deux matrices doit être calculé. Or, $\alpha(n)$ est le bruit généré par la quantification du signal d'entrée. Celui-ci est considéré comme un bruit blanc. Sa matrice d'autocorrélation $\mathbf{\Gamma}(n)$ s'écrit donc :

$$\mathbf{\Gamma}(n) = \sigma_\alpha^2 \mathbf{I_N} + m_\alpha^2 \mathbf{1_N} \tag{2.73}$$

où $\mathbf{I_N}$ est la matrice identité de taille N et $\mathbf{1_N}$ la matrice unitaire de taille N. Le terme m_α désigne la moyenne du bruit $\alpha(n)$ et σ_α^2 sa variance. Ainsi, l'équation (2.72) devient :

$$E(\alpha^t(n)W(n))^2 = \sigma_\alpha^2 Tr\big(E(W(n)W^t(n))\big) + m_\alpha^2 Tr\big(E(W(n)W^t(n))\mathbf{1_N}\big) \tag{2.74}$$

D'autre part, les termes $Tr\big(E(W(n)W^t(n))\big)$ et $Tr\big(E(W(n)W^t(n))\mathbf{1_N}\big)$ s'écrivent :

$$\begin{aligned} Tr\big(E(W(n)W^t(n))\big) &= \sum_{i=1}^{N} E(w_i^2(n)) \\ Tr\big(E(W(n)W^t(n))\mathbf{1_N}\big) &= [\sum_{i=1}^{N} E(w_i(n))]^2 \end{aligned} \tag{2.75}$$

Après convergence, les coefficients $W(n)$ sont très proches des coefficients optimaux W_{opt}. Ainsi, l'approximation suivante $W(n) \approx W_{opt}$ peut être faite à l'état stable. L'expression analytique de ce bruit est alors égale à :

$$Tr(\mathbf{W}(n)\mathbf{\Gamma}(n)) = \sigma_\alpha^2 \sum_{i=1}^{N} w_{opt_i}^2 + m_\alpha^2 (\sum_{i=1}^{N} w_{opt_i})^2 \tag{2.76}$$

avec w_{opt_i} le $i^{\text{ème}}$ coefficient optimal du vecteur W_{opt}. En introduisant $\|W_{opt}\|^2$ la puissance des coefficients optimaux définie par $\|W_{opt}\|^2 = \sum_{i=1}^{N} w_{opt_i}^2$, la puissance du premier terme de bruit est égale à :

$$E(W(n)^t\alpha(n))^2 = \|W_{opt}\|^2 \sigma_\alpha^2 + m_\alpha^2 (\sum_{i=1}^{N} w_{opt_i})^2 \tag{2.77}$$

Ce bruit correspond au bruit d'entrée filtré par les coefficients optimaux.

Calcul du terme $E(\eta(n))^2$

Ce bruit représente le bruit présent en sortie du filtre lié aux bruits générés au sein du filtre. Pour déterminer ce bruit, considérons l'implémentation du filtre représentée à la figure 2.7. Ce terme est la somme des bruits en sortie de chaque multiplication $\nu_i(n)$ et du bruit de cadrage en sortie $\upsilon(n)$.

$$\eta(n) = \sum_{i=0}^{N-1} \nu_i(n) + \upsilon(n) \tag{2.78}$$

Les termes $\nu_i(n)$ dépendent du nombre de bits b_{mult} alloués à la sortie de la multiplication. Si l'implémentation est faite en double précision ($b_{mult} = b_{coef} + b_{in}$), aucun bruit n'est généré. Le terme $\upsilon(n)$ dépend du nombre de bits b_{add} en sortie du FIR. Si l'implémentation est faite en simple précision $b_{add} = b_{mult}$, ce bruit n'est pas présent. Dans le cas le plus général, $\eta(n)$ est la somme de ces différents bruits non-corrélés entre eux.

La moyenne m_η de $\eta(n)$ est donnée par la relation suivante :

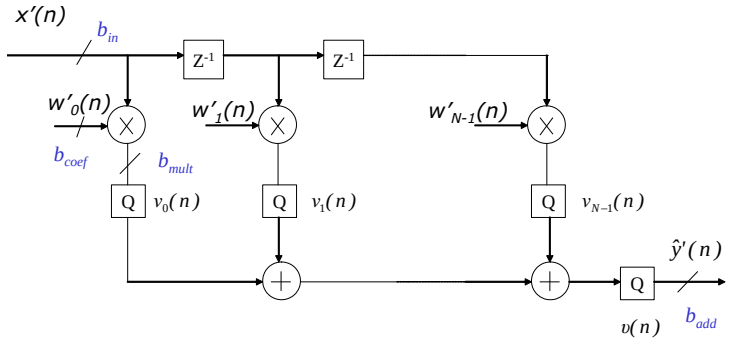

FIG. 2.7 – *Schéma du FIR du LMS en précision finie*

$$m_\eta = N m_\nu + m_\upsilon \tag{2.79}$$

où m_ν et m_υ désignent les moyennes des bruits $\nu(n)$ et $\upsilon(n)$. En notant σ_ν^2 et σ_υ^2 les variances des bruits $\nu(n)$ et $\upsilon(n)$, la variance σ_η^2 du bruit $\eta(n)$ est donnée par l'expression suivante :

$$\sigma_\eta^2 = N\sigma_\nu^2 + \sigma_\upsilon^2 \tag{2.80}$$

Alors, la puissance du bruit au niveau du filtre correspond à la somme de sa moyenne et de sa variance comme indiquée dans l'expression (2.81).

$$E(\eta(n)^2) = m_\eta^2 + \sigma_\eta^2 \tag{2.81}$$

Calcul du terme $E(\rho^t(n)X(n))^2$

Le terme $E(\rho^t(n)X(n))^2$ correspond au bruit sur les coefficients. La principale difficulté pour le calcul de ce bruit vient du fait que le terme $\rho(n)$ ne correspond pas à un bruit de quantification. Le point de départ utilisé est la récurrence de ce terme $\rho(n)$. Celle-ci est explicitée ci-dessous. Les termes $e'(n)$, $y(n)$ et $\beta(n)$ désignent respectivement l'erreur en précision finie, le signal désiré en précision infinie et son bruit associé. L'équation (2.14) est développée et aboutit à l'expression suivante :

$$e'(n) = -W'^t(n)X'(n) - \eta(n) + y(n) + \beta(n) \tag{2.82}$$

Puis, en développant $W'^t(n)X'(n)$ avec les termes de bruits et en négligeant le produit des deux bruits $\alpha(n)$ et $\rho(n)$, cela mène à :

$$e'(n) = \underbrace{-W^t(n)X(n) + y(n)}_{e(n)} - \alpha^t(n)W(n) - \rho^t(n)X(n) - \eta(n) + \beta(n) \tag{2.83}$$

En soustrayant les équations de mise à jour des coefficients en précision finie et en précision infinie, en intégrant l'équation (2.83) et en négligeant les termes d'ordre deux, l'expression de $\rho(n)$ s'exprime par la récurrence suivante :

$$\rho(n+1) = \mathbf{F}(n)\rho(n) + B(n) \tag{2.84}$$

où

$$\mathbf{F}(n) = \mathbf{I_N} - \mu X(n)X^t(n) \tag{2.85}$$

$$B(n) = -\mu X(n)W^t(n)\alpha(n) + \mu X(n)\left(\beta(n) - \eta(n)\right) + \mu\alpha(n)e(n) + \gamma(n) \tag{2.86}$$

Ainsi, une récurrence sur le terme $\rho(n)$ est obtenue. Or, pour calculer ce deuxième bruit, $\rho(n)$ doit être mis au carré en introduisant sa matrice d'autocorrélation $\mathbf{P}(n) = E(\rho(n)\rho^t(n))$.

L'équation (2.84) est transformée en introduisant $\mathbf{P}(n)$ conduisant à l'équation suivante :

$$\mathbf{P}(n+1) = E(B(n)B^t(n)) + E(\mathbf{F}(n)\rho(n)B^t(n)) + E(B(n)\rho^t(n)\mathbf{F}^t(n)) + E(\mathbf{F}(n)\rho(n)\rho^t(n)\mathbf{F}^t(n)) \tag{2.87}$$

Les signaux étant réels et stationnaires, le terme $\mathbf{F}(n)$ est égal à son terme transposé. Cette équation est composée de quatre termes développés ci-dessous.

Le terme $B(n)$ s'écrit

$$B(n) = -\mu\left[X(n)W^t(n)\alpha(n) + X(n)\left(\beta(n) - \eta(n)\right) + \alpha(n)e(n)\right] + \gamma(n) \tag{2.88}$$

Hypothèse Hormis $\gamma(n)$, tous les termes correspondent au produit de μ par un terme de bruit. Les valeurs de μ étant en général inférieures à 1 ainsi que les puissances des signaux, l'hypothèse de prédominance de $\gamma(n)$ au sein de $B(n)$ est faite conduisant à l'approximation suivante :

$$E(B(n)B^t(n)) \approx E(\gamma(n)\gamma^t(n)) \tag{2.89}$$

Le deuxième terme $E(\mathbf{F}(n)\rho(n)B^t(n))$ de l'équation (2.87) se décompose de la façon suivante :

$$E(\mathbf{F}(n)\rho(n)B^t(n)) = E(\rho(n)B^t(n)) - \mu E\left(X(n)X^t(n)\rho(n)B^t(n)\right)$$

Or, comme précédemment, $B(n)$ peut être approximé par le bruit blanc $\gamma(n)$. Ce terme est donc indépendant de tous les autres termes contenus dans l'expression.

$$E(\mathbf{F}(n)\rho(n)B^t(n)) = E(\rho(n))E(\gamma^t(n)) - \mu E\left(X(n)X^t(n)\rho(n)\right)E(\gamma^t(n)) \tag{2.90}$$

En repartant de l'équation (2.84) et en prenant l'espérance, les deux équations suivantes peuvent être déduites :

$$\mu E(X(n)X^t(n)\rho(n)) = E(\gamma(n)) \tag{2.91}$$

$$E(\rho(n)) = \frac{\mathbf{R}_{xx}^{-1}E(\gamma(n))}{\mu} \tag{2.92}$$

où $\mathbf{R}_{xx} = E(X(n)X^t(n))$ est la matrice d'autocorrélation du signal d'entrée. Cette dernière expression est basée sur l'hypothèse d'indépendance entre les données d'entrée $X(n)$ et le vecteur de bruit $\rho(n)$.

En introduisant les équations (2.91) et (2.92), l'expression (2.90) donne :

$$E(\mathbf{F}(n)\rho(n)B^t(n)) = \frac{\mathbf{R}_{xx}^{-1}E(\gamma(n))E(\gamma^t(n))}{\mu} - E(\gamma(n))E(\gamma^t(n)) \tag{2.93}$$

Le terme $E(B(n)\rho^t(n)\mathbf{F}(n))$ est développé de la même façon que précédemment conduisant à l'expression suivante :

$$E(B(n)\rho^t(n)\mathbf{F}(n)) = \frac{E(\gamma(n))E(\gamma^t(n))\mathbf{R^{-1}}}{\mu} - E(\gamma(n))E(\gamma^t(n)) \tag{2.94}$$

Dans ce paragraphe, l'expression $E(\mathbf{F}(n)\rho(n)\rho^t(n)\mathbf{F}(n))$ est calculée. Le terme δ_{ij} désigne le symbole de Kronecker défini de la façon suivante :

$$\delta_{ij} = \begin{cases} 1 & \text{si} \quad i = j \\ 0 & \text{si} \quad i \neq j \end{cases} \tag{2.95}$$

Soit $(i,j) \in [1:N]^2$. En utilisant le fait que $\mathbf{F}(n) = \mathbf{I_N} - \mu X(n)X^t(n)$, l'expression suivante est développée.

$$\begin{aligned} E(\mathbf{F}(n)\rho(n)\rho^t(n)\mathbf{F}(n))_{ij} &= E\left(\sum_{k=1}^{N}\sum_{l=1}^{N} F_{ik}(n)\rho_k(n)\rho_l(n)F_{lj}(n)\right) \\ &= E\left(\sum_{k=1}^{N}\sum_{l=1}^{N}(\delta_{ik} - \mu x_i(n)x_k(n))\rho_k(n)\rho_l(n)(\delta_{lj} - \mu x_l(n)x_j(n))\right) \\ &= E\left(\rho_i(n)\rho_j(n)\right) - \mu\sum_{l=1}^{N} E\left(\rho_i(n)\rho_l(n)x_l(n)x_j(n)\right) - \mu\sum_{k=1}^{N} E\left(x_i(n)x_k(n)\rho_k(n)\rho_j(n)\right) \\ &\quad + \mu^2\sum_{k=1}^{N}\sum_{l=1}^{N} E\left(\rho_k(n)\rho_l(n)x_i(n)x_k(n)x_l(n)x_j(n)\right) \end{aligned} \tag{2.96}$$

Le signal d'entrée $X(n)$ est supposé indépendant du bruit sur les coefficients $\rho(n)$. De plus, en négligeant le terme en μ^2, faible devant les autres termes, l'équation (2.97) est obtenue.

$$E\left(\mathbf{F}(n)\rho(n)\rho^t(n)\mathbf{F}(n)\right)_{ij} = \mathbf{P}_{ij}(n) - \mu(\mathbf{R}_{xx}\mathbf{P}(n))_{ij} - \mu(\mathbf{P}(n)\mathbf{R}_{xx})_{ij} \tag{2.97}$$

L'équation (2.87) est développée avec les équations (2.89), (2.93), (2.94) et (2.97).

$$\begin{aligned} \mathbf{P}(n+1) = \mathbf{P}(n) - \mu(\mathbf{R}_{xx}\mathbf{P}(n)) - \mu(\mathbf{P}(n)\mathbf{R}_{xx}) + \frac{E(\gamma(n))E(\gamma^t(n))\mathbf{R}_{xx}^{-1}}{\mu} \\ + \frac{\mathbf{R}_{xx}^{-1}E(\gamma(n))E(\gamma^t(n))}{\mu} - 2E(\gamma(n))E(\gamma^t(n)) + E(\gamma(n)\gamma^t(n)) \end{aligned} \tag{2.98}$$

Or à l'état stable, $\mathbf{P}(n+1) = \mathbf{P}(n)$, et en prenant la trace de cette égalité, l'expression suivante est obtenue :

$$2\mu Tr(\mathbf{R}_{xx}\mathbf{P}(n)) = 2\frac{Tr\left(E(\gamma(n))E(\gamma^t(n))\mathbf{R}_{xx}^{-1}\right)}{\mu} - 2Tr\left(E(\gamma(n))E(\gamma^t(n))\right) + Tr\left(E(\gamma(n)\gamma^t(n))\right) \tag{2.99}$$

Le premier terme $Tr(\mathbf{R}_{xx}\mathbf{P}(n))$ correspond au second terme de l'expression (2.71) :

$$Tr(\mathbf{R}_{xx}\mathbf{P}(n)) = E(\rho^t(n)X(n))^2 \tag{2.100}$$

Les trois autres termes sont obtenus à partir des équations (2.101), (2.102) et (2.103).

$$Tr\left(E(\gamma(n))E(\gamma^t(n))\mathbf{R}_{xx}^{-1}\right) = m_\gamma^2\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx_{ki}}^{-1}) \tag{2.101}$$

Pour calculer ce terme, tous les éléments de la matrice $\mathbf{R}_{xx}^{-1}$ sont sommés puis multipliés par m_γ^2 (moyenne au carré de $\gamma(n)$).

$$Tr\Big(E\big(\gamma(n)\big)E\big(\gamma^t(n)\big)\Big) = Nm_\gamma^2 \tag{2.102}$$

Ce terme correspond à N fois la moyenne au carré du bruit $\gamma(n)$.

$$Tr\Big(E\big(\gamma(n)\gamma^t(n)\big)\Big) = N(m_\gamma^2 + \sigma_\gamma^2) \tag{2.103}$$

Ce terme correspond à N fois la puissance du bruit $\gamma(n)$. Avec tous les termes développés, l'équation (2.99) aboutit à :

$$E(\rho^t(n)X(n))^2 = m_\gamma^2 \frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx_{ki}}^{-1})}{\mu^2} + \frac{N(\sigma_\gamma^2 - m_\gamma^2)}{2\mu} \tag{2.104}$$

Calcul global

La puissance du bruit global $b_y(n)$ est la somme des trois puissances de bruits calculées précédemment :

$$Pb = \|W_{opt}\|^2\sigma_\alpha^2 + m_\alpha^2(\sum_{i=1}^{N} w_{opt_i})^2 + m_\gamma^2 \frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx_{ki}}^{-1})}{\mu^2} + \frac{N(\sigma_\gamma^2 - m_\gamma^2)}{2\mu} + m_\eta^2 + \sigma_\eta^2 \tag{2.105}$$

Plusieurs termes sont nécessaires pour calculer l'expression. En premier lieu, trois bruits doivent être connus et tout particulièrement leurs moments d'ordre un et deux, à savoir le bruit $\alpha(n)$ sur le signal d'entrée, $\gamma(n)$ au niveau de la multiplication et enfin $\eta(n)$, le bruit à l'intérieur du filtre. En second lieu, les coefficients optimaux doivent être connus. Ils peuvent être estimés en prenant la moyenne des coefficients à l'état stable. En effet, ceux-ci oscillent autour de la valeur optimale [15]. Enfin, la matrice d'autocorrélation du signal d'entrée est nécessaire au calcul de la puissance du bruit. Si le bruit se rapproche d'un bruit blanc, la matrice peut être approchée par une matrice diagonale (significative des bruits blancs), simplifiant ainsi les calculs. A l'opposé, si le signal est très corrélé, la matrice peut être approximée par une matrice homogène dans un souci de simplification.

Cas de la quantification par arrondi Dans ce paragraphe, le cas d'une quantification par arrondi est plus spécifiquement étudié afin d'analyser les simplifications sur le calcul de la puissance du bruit. Le bruit $\alpha(n)$ sur le signal d'entrée aura une moyenne nulle dans ce cas. Par contre, $\eta(n)$, le bruit associé au FIR, $\gamma(n)$, le bruit dû à un cadrage et, a fortiori, $\rho(n)$ n'auront pas forcément une moyenne nulle. La moyenne de ces trois termes doit toujours être prise en compte. La puissance globale du bruit est définie par l'expression suivante :

$$Pb = \|W_{opt}\|^2\sigma_\alpha^2 + m_\gamma^2 \frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx_{ki}}^{-1})}{\mu^2} + \frac{N(\sigma_\gamma^2 - m_\gamma^2)}{2\mu} + m_\eta^2 + \sigma_\eta^2 \tag{2.106}$$

Dans le cas de la quantification par arrondi convergent. Les moyennes de tous les bruits de quantification sont nulles. L'expression de la puissance du bruit est la suivante :

$$Pb = \|W_{opt}\|^2\sigma_\alpha^2 + \frac{N\sigma_\gamma^2}{2\mu} + \sigma_\eta^2 \tag{2.107}$$

L'utilisation de l'arrondi convergent simplifie énormément l'expression. D'ailleurs, l'utilisation de cet arrondi permet de comparer avec les autres modèles déjà proposés et s'appliquant seulement dans ce cadre. Cette expression ne fait pas intervenir les moyennes des bruits. L'expression générale (2.105) de la puissance du bruit en sortie du système en prenant en compte les moyennes des bruits est bien plus complexe. Elle justifie sa mise en œuvre dans le cas d'une loi de quantification quelconque par rapport aux modèles développés dans la partie précédente et valable uniquement pour l'arrondi convergent. Le modèle proposé est plus général et permet de modéliser la puissance du bruit dans le cas d'une quantification par troncature notamment.

2.2.2 Algorithme Leaky LMS

Lors d'une implémentation en précision finie, des risques de dépassements existent au niveau des coefficients et de leur mise à jour. Pour éviter ce problème l'algorithme Leaky-LMS est introduit. La mise à jour des coefficients est faite selon l'équation :

$$W(n+1) = \lambda W(n) + \mu e(n) X(n) \tag{2.108}$$

où λ est une constante comprise entre 0 et 1 appelée coefficient de leakage. Le calcul de la puissance du bruit est fait pour cet algorithme. Par rapport au LMS, seule la mise à jour des coefficients diffère. De ce fait, seul le bruit lié à ceux-ci varie. L'équation de récurrence sur le bruit lié aux coefficients est donné par la relation suivante :

$$\rho(n+1) = \mathbf{F}(n)\rho(n) + B(n) \tag{2.109}$$

avec

$$\mathbf{F}(n) = \lambda \mathbf{I_N} - \mu X(n) X^t(n) \tag{2.110}$$

$$B(n) = -\mu X(n) W^t(n) \alpha(n) + \mu X(n)(\beta(n) - \eta(n)) + \mu \alpha(n) e(n) + \gamma(n) + \phi(n) \tag{2.111}$$

où $\phi(n)$ représente les bruits généré par la multiplication des coefficients par le coefficient de leakage λ. Le terme $\gamma(n)$ ne prédomine pas forcément par rapport à $\phi(n)$. L'hypothèse de prédominance de $\gamma(n)$ n'est pas prise en compte ici. La matrice $\mathbf{P}(n)$ d'autocorrélation du bruit $\rho(n)$ est égale à :

$$\begin{aligned}\mathbf{P}(n+1) = \lambda^2 \mathbf{P}(n) - \lambda\mu(\mathbf{R}_{xx}\mathbf{P}(n)) - \lambda\mu(\mathbf{P}(n)\mathbf{R}_{xx}) + \frac{E(B(n))E(B^t(n))\mathbf{R}_{xx}^{-1}}{\mu} \\ + \frac{\mathbf{R}_{xx}^{-1}E(B(n))E(B(n))}{\mu} - 2E(B(n))E(B(n)) + E(B(n)B(n))\end{aligned} \tag{2.112}$$

Avec le même développement que pour le LMS, la puissance du bruit lié aux coefficients est donnée par l'expression suivante :

$$\begin{aligned}E(\rho^t(n)X(n))^2 = Tr\Bigg(\Big((1-\lambda^2)\mathbf{I_N} + 2\mu\lambda\mathbf{R}_{xx}\Big)^{-1} \\ \Big[N\sigma_b^2\mathbf{R}_{xx} + m_b^2\Big(\big(1-\lambda\big)\mathbf{I_N} + \mu\mathbf{R}_{xx}\Big)^{-1}\Big(\big(1+\lambda\big)\mathbf{I_N} - \mu\mathbf{R}_{xx}\Big)\Big(\mathbf{1_N}\mathbf{R}_{xx}\Big)\Big]\Bigg)\end{aligned} \tag{2.113}$$

Cette expression permet d'établir la puissance du bruit en sortie du Leaky-LMS. En prenant $\lambda = 1$, l'expression de la puissance du bruit lié aux coefficients pour le LMS est retrouvée.

2.2.3 Algorithme NLMS

Dans cette partie, un modèle d'évaluation de la précision pour l'algorithme NLMS est développé. La puissance du bruit généré en sortie du système est calculée comme pour l'algorithme LMS. Le schéma du système est présenté sur la figure 2.8. Par rapport au LMS, la seule différence est au niveau du gain. Pour le LMS, celui-ci est défini par le pas d'adaptation μ alors que pour le NLMS, le gain en précision finie $k'(n)$ est défini par la relation suivante :

$$k'(n) = \frac{\mu}{X'^t(n)X'(n)} \tag{2.114}$$

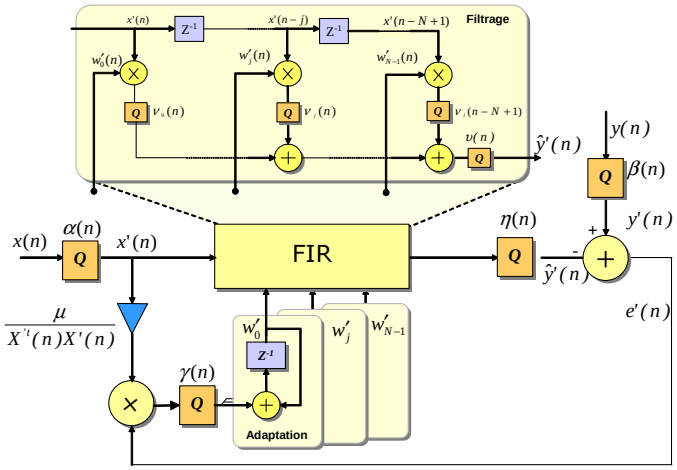

FIG. 2.8 – *Filtre NLMS en précision finie*

La propagation des bruits au sein de ce gain $k'(n)$ est étudiée en [76]. Le bruit $\nu(n)$ en sortie du gain $k'(n)$ est généré par la propagation du bruit d'entrée $\alpha(n)$ au sein de celui-ci. Le bruit en sortie $\nu(n)$ est défini par la relation suivante :

$$\nu(n) = 2\frac{\mu\alpha^t(n)X(n)}{(X'^t(n)X'(n))^2} \tag{2.115}$$

En pratique, pour simplifier l'implantation et éviter une division, la normalisation est effectuée en approchant le terme de normalisation par la puissance de deux la plus proche. Cette approche ne génère pas de nouveau bruit mais assure la propagation du bruit $\alpha(n)$ en son sein. Ensuite, la méthode appliquée est la même que pour le LMS. Seuls deux éléments diffèrent. Tout d'abord, un nouveau bruit $\nu(n)$ apparaît dans le système. Ensuite, le gain n'est plus μ mais $\frac{\mu}{X'^t(n)X'(n)}$. Ainsi tous les termes μ sont modifiés dans l'expression de la puissance du bruit. L'équation (2.84) développée pour le LMS devient dans le cas du NLMS.

$$\rho(n+1) = \mathbf{F}(n)\rho(n) + B(n) \tag{2.116}$$

où :
$$\mathbf{F}(n) = \mathbf{I_N} - \mu\frac{X(n)X^t(n)}{X^t(n)X(n)}$$

$$\begin{aligned} B(n) &= -\frac{\mu}{X^t(n)X(n)}X(n)W^t(n)\alpha(n) + \frac{\mu}{X^t(n)X(n)}X(n)(\beta(n)-\eta(n)) \\ &+ \frac{\mu}{X^t(n)X(n)}\alpha(n)e(n) + \nu(n)e(n) + \gamma(n) \end{aligned} \tag{2.117}$$

Slock dans [104] fournit l'égalité suivante $E(\frac{X(n)X^t(n)}{X(n)X^t(n)}) = \frac{\mathbf{R}_{xx}}{Tr(\mathbf{R}_{xx})}$. Le terme $B(n)$ contient un nouveau terme lié à la normalisation. Ce nouveau terme est pris en compte dans le modèle. En introduisant le terme ξ désignant l'erreur quadratique ainsi que la relation précédente, l'expression finale de la puissance du bruit est la suivante :

$$\begin{aligned} Pb = \|W_{opt}\|^2\sigma_\alpha^2 + m_\alpha^2(\sum_{i=1}^{N} w_{opt_i})^2 + m_\gamma^2 Tr(\mathbf{R}_{xx})^2\frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}^{-1}_{xx_{ki}})}{\mu^2} \\ + Tr(\mathbf{R}_{xx})\frac{N(\sigma_\gamma^2 - m_\gamma^2) + \xi Tr\Big(E(\nu(n)\nu^t(n))\Big)}{2\mu} + m_\eta^2 + \sigma_\eta^2 \end{aligned} \tag{2.118}$$

Le terme $Tr\Big(E((\nu(n)\nu^t(n)))\Big)$ est déterminé dans [76]. La principale différence entre le LMS et le NLMS vient de la normalisation assurant la convergence de l'algorithme pour $\mu \in$ [0,2]. Pour ce faire, le pas d'adaptation est divisé par la trace de $\mathbf{R}_{xx}$. Ceci est repris dans l'équation précédente. En effet, celle-ci est identique par rapport à celle du LMS excepté que le pas μ est normalisé par $Tr(\mathbf{R}_{xx})$.

2.2.4 Algorithmes de Projection Affine (APA)

Dans cette partie, un autre type d'algorithme correspondant aux Algorithmes de Projection Affine (APA) est étudié [81]. Ces algorithmes sont dérivés du NLMS pour avoir une convergence plus rapide. Pour un pas d'adaptation $\mu = 1$, l'algorithme NLMS annule une erreur a posteriori.

$$y(n) - W^t(n+1)X(n) = 0 \tag{2.119}$$

Les algorithmes APA permettent d'annuler K erreurs a posteriori pour $\mu = 1$.

$$y(n-i+1) - W^t(n+1)X(n-i+1) = 0 \text{ pour } i \in [1:K] \tag{2.120}$$

Les équations régissant cet algorithme sont les suivantes :

$$\begin{aligned} \zeta(n) &= Y(n) - \mathbf{X}^t(n)W(n) \\ W(n+1) &= W(n) + \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n) + \delta\mathbf{I_K}]^{-1}\zeta(n) \end{aligned} \tag{2.121}$$

où $X(n)$ représente le vecteur de taille N des données d'entrée [$x(n)$,$x(n-1)$,...$x(n-N+1)$]^t et $\mathbf{X}(n)$ est la matrice de taille NxK des K derniers vecteurs d'observation $\mathbf{X}(n)$ = [$X(n)$,$X(n-1)$,...$X(n-K+1)$]. Ainsi, le signal d'entrée n'est plus seulement le vecteur $X(n)$ comme précédemment mais la matrice $\mathbf{X}(n)$ contenant les K derniers vecteurs d'observation. Les termes $Y(n)$ et $\zeta(n)$ représentent respectivement, le signal désiré et l'erreur commise et sont des vecteurs de taille K. La principale difficulté correspond à l'inversion de la matrice. Une version appelée NLMS-OCF de ce type d'algorithmes est développée pour résoudre ce problème. Dans cette version, les vecteurs d'entrée sont traités de façon à être orthogonaux entre eux. La matrice $\mathbf{X}^t(n)\mathbf{X}(n)$ est alors diagonale et l'inversion de celle-ci n'est plus nécessaire.

Soit $V(n)$ le vecteur d'écart des coefficients par rapport au coefficients optimaux.

$$V(n) = W(n) - W_{opt} \tag{2.122}$$

Alors le signal désiré peut s'écrire :

$$Y(n) = \mathbf{X}^t(n)W_{opt} + U(n) \tag{2.123}$$

où $U(n)$ est un vecteur bruit. Alors, la mise à jour du vecteur $V(n)$ donne :

$$V(n+1) = \mathbf{G}(n)V(n) + \mathbf{J}(n)U(n) \tag{2.124}$$

avec $\mathbf{G}(n) = \mathbf{I_N} - \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\mathbf{X}^t(n)$ et $\mathbf{J}(n) = \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}$. Cette mise à jour équivaut à projeter le vecteur d'écart $V(n)$ sur un espace affine (défini par les vecteurs colonnes de $\mathbf{G}(n)$) orthogonal à l'espace constitué par les vecteurs colonnes de la matrice $\mathbf{X}(n)$ pour $\mu = 1$. Ainsi, ces algorithmes sont appelés algorithmes de projection affine. Cette projection permet d'avoir un temps de convergence plus rapide. Pour mieux comprendre, une interprétation géométrique peut être faite [79]. Soit l'hyperplan $\Pi_n = \{W \in \mathbb{R}^N / y(n) - W^t X(n) = 0\}$. L'algorithme NLMS permet de construire une suite de vecteurs $\{W(n), n > 0\}$ convergeant vers le filtre optimal W_{opt}. Par définition, ce filtre optimal appartient à l'intersection de tous les hyperplans $\{\Pi(n), n > 0\}$. Ainsi, le NLMS projette le vecteur $W(n-1)$ sur $\Pi(n)$ colinéairement à $X(n)$ pour obtenir $W(n)$ vérifiant $||V(n+1)|| < ||V(n)||$. La vitesse de convergence est liée à l'angle entre

les hyperplans $\Pi(n-1)$ et $\Pi(n)$ correspondant à la corrélation entre $X(n-1)$ et $X(n)$. Dans le cas de l'APA la projection est faite en direction de l'intersection des K hyperplans $\bigcap_{i=0}^{K-1} \Pi(n-i)$. Cette projection permet de réduire la norme de $V(n)$ et donc d'accroître la vitesse de convergence de l'algorithme. Des études plus détaillées sont présentées dans [93] et [96]. Celles-ci étudient son temps de convergence en particulier pour la version NLMS-OCF.

Algorithme APA en précision finie

Dans cette partie, l'étude de l'APA en précision finie est faite et un modèle pour l'évaluation de sa précision est proposé [89]. En effet, à notre connaissance, aucun modèle n'a été proposé pour étudier le comportement des algorithmes APA en virgule fixe. Le modèle de l'APA en précision finie est le suivant :

$$\begin{aligned} \zeta'(n) &= Y'(n) - \mathbf{X}'^{t}(n)W'(n) - \eta(n) \\ W'(n+1) &= W'(n) + \mu\mathbf{X}'(n)[\mathbf{X}'^{t}(n)\mathbf{X}'(n) + \delta\mathbf{I_K}]^{-1}\zeta'(n) + \gamma(n) \end{aligned} \tag{2.125}$$

où $X'(n)$ représente le vecteur de taille N des données d'entrée quantifiées $[x'(n)$,$x'(n-1)$,...$x'(n-N+1)]^t$. Le terme $\mathbf{X}'(n)$ désigne la matrice de taille NxK des K derniers vecteurs d'observation $X'(n) = [x'(n)$,$x'(n-1)$,...$x'(n-K+1)]$. Les termes $Y'(n)$ et $\zeta'(n)$ représentent respectivement, le signal désiré quantifié et l'erreur commise quantifiée et sont des vecteurs de taille K. La constante δ permet de rendre régulière la matrice $X(n)'^{t}X'(n)$ et est supposée être une somme de puissance de 2. Le terme $\gamma(n)$ est un vecteur bruit de taille N dû au calcul de $X'(n)[\mathbf{X}'^{t}(n)\mathbf{X}'(n)+\delta\mathbf{I_K}]^{-1}\zeta'(n)$. Le terme $\eta(n)$ est un vecteur bruit de taille K généré par le filtre. Les autres bruits de quantification sont exprimés à l'aide des équations suivantes :

$$\begin{aligned} \mathbf{X}'(n) &= \mathbf{X}(n) + \boldsymbol{\alpha}(n) \\ Y'(n) &= Y(n) + \beta(n) \\ [\mathbf{X}'^{t}(n)\mathbf{X}'(n) + \delta\mathbf{I_K}]^{-1} &= [\mathbf{X}^{t}(n)\mathbf{X}(n) + \delta\mathbf{I_K}]^{-1} + \nu(n) \\ W'(n) &= W(n) + \rho(n) \end{aligned} \tag{2.126}$$

Le bruit $\boldsymbol{\alpha}(n)$ est une matrice de taille NxK.

Puissance du bruit

L'objectif de cette partie est d'estimer analytiquement la puissance du bruit généré dans le système. Celle-ci est déterminée en suivant la même démarche que dans le cas du NLMS et du LMS et en négligeant les produits des termes de bruit.

$$\begin{aligned} Tr[E(B_y(n)B_y^t(n))] &= Tr\Big(E\Big[\Big(\mathbf{X}'^{t}(n)W'(n) + \eta(n) - \mathbf{X}^{t}(n)W(n)\Big) \\ &\quad . \quad \Big(X(n)'^{t}W'(n) + \eta(n) - \mathbf{X}^{t}(n)W(n)\Big)^{t}\Big]\Big) \\ &= Tr\left[E\left((\boldsymbol{\alpha}^{t}(n)W(n))(\boldsymbol{\alpha}^{t}(n)W(n))^{t}\right)\right] \\ &+ Tr\left[E\left((\mathbf{X}^{t}(n)\rho(n))(\mathbf{X}^{t}(n)\rho(n))^{t}\right)\right] + Tr\left[E\left(\eta(n)\eta^{t}(n)\right)\right] \end{aligned} \tag{2.127}$$

La puissance du bruit global est donc la somme de trois termes développés dans la suite.

Bruit d'entrée Le premier terme de l'équation (2.127) est donné par la propagation du bruit d'entrée $\alpha(n)$ à travers le système selon l'expression suivante :

$$B_x(n) = \boldsymbol{\alpha}^t(n)W(n) \tag{2.128}$$

Or, l'égalité suivante est vérifiée comme dans le cas du LMS :

$$Tr(E(B_x(n)B_x^t(n))) = Tr\Big(E\big(\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^t(n)\big)E\big(W(n)W^t(n)\big)\Big) \tag{2.129}$$

En approximant à l'état stable les coefficients $W(n)$ par les coefficients optimaux W_{opt}, et sachant que $\boldsymbol{\alpha}(n)$ est un bruit blanc, la puissance du premier terme est égale à :

$$Tr\left(E\left(B_x(n)B_x^t(n)\right)\right) = K\sigma_\alpha^2 \sum_{i=0}^{N-1} w_{opt_i}^2 + Km_\alpha^2 \left(\sum_{i=0}^{N-1} w_{opt_i}\right)^2 \tag{2.130}$$

Bruit lié aux coefficients Le bruit $B_w(n)$ lié aux coefficients est le deuxième terme de l'équation (2.127) et est égal à :

$$B_w(n) = \mathbf{X}^t(n)\rho(n) \tag{2.131}$$

En écrivant $\mathbf{P}(n) = E(\rho(n)\rho^t(n))$ et en supposant la non-corrélation entre le vecteur bruit $\rho(n)$ et les données d'entrée $\mathbf{X}(n)$ comme dans le cas du LMS, la puissance de ce terme est donnée par la relation suivante :

$$Tr(E(B_w(n)B_w^t(n))) = Tr\Big(E\big(\mathbf{X}(n)\mathbf{X}^t(n)\big)\mathbf{P}(n)\Big) \tag{2.132}$$

Or, depuis les équations (2.125), (2.126) et l'equation de mise à jour des coefficients en précision infinie, la récursion suivante peut être écrite :

$$\rho(n+1) = \mathbf{F}(n)\rho(n) + B(n) \tag{2.133}$$

avec

$$\mathbf{F}(n) = \mathbf{I_N} - \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\mathbf{X}^t(n) \tag{2.134}$$

$$\begin{aligned} B(n) &= \gamma(n) + \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}(\beta(n) - \eta(n)) + \mu\boldsymbol{\alpha}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\zeta(n) \\ &+ \mu\mathbf{X}(n)\nu(n)\zeta(n) - \mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\alpha^t(n)W(n) \end{aligned} \tag{2.135}$$

En supposant qu'à l'état stable, $\mathbf{P}(n+1) = \mathbf{P}(n)$, et avec $\mathbf{B}(n) = E(B(n)B^t(n))$, l'équation (2.133) conduit à :

$$\begin{aligned} Tr\Big(E\big(\mathbf{X}(n)\mathbf{X}^t(n)\big)\mathbf{P}(n)\Big) &= Tr\Big(E\big(\mathbf{X}(n)\mathbf{X}^t(n)\big)\big(\mathbf{I_N} - E(\mathbf{F}(n)\mathbf{F}(n)^t)\big)^{-1}.\big[\mathbf{B}(n) \\ &+ 2E(\mathbf{F}(n))(\mathbf{I_N} - E(\mathbf{F}(n)))^{-1}E(B(n))E(B^t(n))\big]\Big) \end{aligned} \tag{2.136}$$

Les termes $E(\mathbf{F}(n)\mathbf{F}(n)^t)$, $E(\mathbf{F}(n))$ et $E(\mathbf{X}(n)\mathbf{X}^t(n))$ peuvent être estimés facilement par une simulation en virgule flottante. Ces termes ne font pas intervenir de termes de bruits. Ainsi une seule simulation virgule flottante est nécessaire pour obtenir la valeur de ces termes. Les autres termes peuvent être développés en séparant les termes de bruit et les signaux tel que détaillé dans [89] et repris dans l'expression suivante :

$$\begin{aligned} B(n) &= \underbrace{\gamma(n)}_{B_1(n)} + \underbrace{\mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}(\beta(n)-\eta(n))}_{B_2(n)} \\ &+ \underbrace{\mu\boldsymbol{\alpha}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\zeta(n)}_{B_3(n)} + \underbrace{\mu\mathbf{X}(n)\nu(n)\zeta(n)}_{B_4(n)} \\ &- \underbrace{\mu\mathbf{X}(n)[\mathbf{X}^t(n)\mathbf{X}(n)]^{-1}\boldsymbol{\alpha}^t(n)W(n)}_{B_5(n)} \end{aligned} \tag{2.137}$$

Bruit généré par le filtre Le bruit $\eta(n)$ généré par le filtre est un vecteur de taille K. En notant m_η sa moyenne, σ_η^2 sa variance, la trace de sa matrice d'autocorrélation est égale à :

$$Tr(E(\eta(n)\eta(n)^t)) = K(m_\eta^2 + \sigma_\eta^2) \tag{2.138}$$

Puissance du bruit En utilisant les équations (2.130), (2.136) et (2.138), la puissance du bruit global en sortie du système est donnée par l'expression suivante :

$$\begin{aligned} Tr[E(B_y(n)B_y^t(n))] = K\Bigg(&\sigma_\alpha^2\sum_{i=0}^{N-1} w_{opt_i}^2 + m_\alpha^2(\sum_{i=0}^{N-1} w_{opt_i})^2 + m_\eta^2 + \sigma_\eta^2\Bigg) \\ &+ Tr\Bigg(E\Big(\mathbf{X}(n)\mathbf{X}^t(n)\Big)\Big(\mathbf{I_N} - E(\mathbf{F}(n)\mathbf{F}^t(n))\Big)^{-1} \\ &\Big[\mathbf{B}(n) + 2E(\mathbf{F}(n))\Big(\mathbf{I_N} - E(\mathbf{F}(n))\Big)^{-1}E(B(n))E(B^t(n))\Big]\Bigg) \end{aligned} \tag{2.139}$$

Un modèle analytique de la puissance du bruit en sortie du système a donc été présenté. Cette expression définit la puissance du bruit en sortie de l'application. Le bruit de sortie est un vecteur de taille K. Or, en pratique, seul le premier terme de ce vecteur a un intérêt. En effet, ce terme de bruit correspond à l'échantillon de sortie $\widehat{y'}(n)$ du filtre à l'instant n. Cet échantillon de sortie permet d'approcher au mieux la séquence désirée $y'(n)$. Les autres échantillons du vecteur de sortie du filtre servent uniquement dans la mise à jour des coefficients. De ce fait, en pratique, la puissance du bruit en sortie de l'application se résume à la puissance du bruit sur le premier échantillon du vecteur de sortie $\widehat{Y'}(n)$. La puissance utile du bruit en sortie se retrouve alors dans l'expression précédente. Elle correspond au premier terme de la matrice $E(B_y(n)B_y^t(n))$. La puissance utile P_u est alors définie par la relation suivante :

$$\begin{aligned} P_u = &\sigma_\alpha^2\sum_{i=0}^{N-1} w_{opt_i}^2 + m_\alpha^2(\sum_{i=0}^{N-1} w_{opt_i})^2 + m_\eta^2 + \sigma_\eta^2 \\ &+ \Bigg(E\Big(\mathbf{X}(n)\mathbf{X}^t(n)\Big)\Big(\mathbf{I_N} - E(\mathbf{F}(n)\mathbf{F}^t(n))\Big)^{-1} \\ &\Big[\mathbf{B}(n) + 2E(\mathbf{F}(n))\big(\mathbf{I_N} - E(\mathbf{F}(n))\big)^{-1}E(B(n))E(B^t(n))\Big]\Bigg)_{(1,1)} \end{aligned} \tag{2.140}$$

où le terme (1,1) désigne le premier terme de la matrice. Les deux premiers termes de cette expression sont divisés par K pour établir l'expression de la puissance utile du bruit en sortie. Dans la prochaine partie, l'expression de la puissance du bruit de sortie est développée dans le cas de l'algorithme NLMS-OCF.

Algorithme NLMS-OCF

L'objectif de cette partie est d'appliquer le modèle précédent à l'algorithme NLMS-OCF. Cet algorithme est un algorithme de projection affine dans lequel les K dernières observations sont orthogonales. Dans un souci de simplification, $X(n)$ est supposé centré et σ_x^2 désigne la variance d'un échantillon.

Le terme $E\Big(\mathbf{X}(n)\big(\mathbf{X}^t(n)\mathbf{X}(n)\big)^{-1}\mathbf{X}^t(n)\Big)$ peut être approché par :

$$E\Big(\mathbf{X}(n)\big(\mathbf{X}^t(n)\mathbf{X}(n)\big)^{-1}\mathbf{X}^t(n)\Big) \approx \frac{K}{N}\mathbf{I_N} \tag{2.141}$$

Bruit lié aux coefficients dans le NLMS-OCF Chaque terme de l'équation (2.136) peut être simplifié de la façon suivante :

$$\Big(\mathbf{I_N} - E\big(\mathbf{F}(n)\mathbf{F}^t(n)\big)\Big) \approx (2\mu - \mu^2)\frac{K}{N}\mathbf{I_N} \tag{2.142}$$

$$E\big(\mathbf{X}(n)\mathbf{X}^t(n)\big) \approx K\sigma_x^2\mathbf{I_N} \tag{2.143}$$

$$(\mathbf{I_N} - E(\mathbf{F}(n))) \approx \mu\frac{K}{N}\mathbf{I_N} \tag{2.144}$$

En utilisant l'équation (2.136) et en remplaçant par les relations précédentes, le bruit dû aux coefficients est égal à :

$$\begin{aligned} &Tr\Big(E\big(\mathbf{X}(n)\mathbf{X}^t(n)\big)\mathbf{P}(n)\Big) \\ &= \tfrac{N\sigma_x^2}{(2\mu-\mu^2)}\Big[Tr\big(\mathbf{B}(n)\big) + 2\tfrac{N(1-\mu\frac{K}{N})}{K\mu}Tr\big(E(B(n))E(B^t(n))\big)\Big] \end{aligned} \tag{2.145}$$

Les termes $Tr(\mathbf{B}(n))$ et $Tr\big(E(B(n))E(B^t(n))\big)$ doivent être calculés pour déterminer complètement le bruit dû aux coefficients. Soit $\mathbf{R}_{B_i}$ la matrice d'autocorrélation de chacun des cinq termes $B_i(n)$ de l'expression (2.137). Ces cinq termes de bruits sont non-corrélés car ils dépendent de sources de bruit différentes donc indépendantes. Le terme $Tr(\mathbf{B}(n))$ est exprimé en fonction de ces bruits dans l'équation suivante :

$$Tr(\mathbf{B}(n)) = \sum_{i=1}^{5} Tr(\mathbf{R}_{B_i}) \tag{2.146}$$

Par la suite, chaque terme $\mathbf{R_{B_i}}$ est développé.

$$Tr(\mathbf{R}_{B_1}) = Tr(E(\gamma(n)\gamma^t(n))) = K^2Nm_\gamma^2 + KN\sigma_\gamma^2 \tag{2.147}$$

Ici, $B_1(n)$ peut être considéré comme la somme de K bruits non-corrélés avec la même densité de probabilité.

$$\begin{aligned} Tr(\mathbf{R}_{B_2}) = \mu^2 Tr\Bigg[&\Big((\sigma_\beta^2 + \sigma_\eta^2)\mathbf{I_N} + (m_\beta^2 + m_\eta^2)\mathbf{1_N}\Big) \\ &\Big(E\Big(\mathbf{X}(n)\big[\mathbf{X}^t(n)\mathbf{X}(n)\big]^{-2}\mathbf{X}^t(n)\Big)\Big)\Bigg] \\ = K\mu^2(E(\beta^2) + E(\eta^2))&\frac{1}{N\phi_x + N(N-1)\sigma_x^4} \end{aligned} \tag{2.148}$$

où ϕ_x est le kurtosis du signal d'entrée $x(n)$. Le terme $B_2(n)$ peut être exprimé comme la somme de K bruits.

$$
\begin{aligned}
Tr(\mathbf{R}_{B_3}) = \mu^2 Tr\Bigg[&\Big(\sigma_\alpha^2 \mathbf{I_N} + m_\alpha^2 \mathbf{1_N}\Big) \\
&\Big(E\Big(\big[\mathbf{X}^t(n)\mathbf{X}(n)\big]^{-1}\zeta(n)\zeta(n)^t\big[\mathbf{X}^t(n)\mathbf{X}(n)\big]^{-1}\Big)\Big)\Bigg] \\
&= K\mu^2(\sigma_\alpha^2 + m_\alpha^2)\frac{\xi}{N\phi_x + N(N-1)\sigma_x^4}
\end{aligned}
\tag{2.149}
$$

Le terme $B_3(n)$ est également la somme de K bruits de moyenne nulle car $\zeta(n)$ est centré. Le terme ξ désigne l'erreur quadratique. Les deux derniers termes $B_4(n)$ et $B_5(n)$ peuvent être développés de la même façon. Ensuite, $Tr(E(B(n))E(B(n)^t))$ doit être développé. Le terme $E(B(n))$ est égal à :

$$E(B(n)) = [Km_\gamma, Km_\gamma, \dots Km_\gamma]^t \tag{2.150}$$

Donc $Tr(E(B(n))E(B(n)^t))$ est défini par la relation suivante :

$$Tr(E(B(n))E(B(n)^t)) = NK^2 m_\gamma^2 \tag{2.151}$$

Ce terme peut être interprété comme la somme de K sources de bruit. Ainsi, chaque terme du NLMS-OCF (B_1, B_2, B_3, B_4 et B_5) est la somme de K bruits. Dans le cas où seule la puissance du bruit utile en sortie de l'application est considérée, le facteur K disparaît. La puissance utile du bruit en sortie de l'APA est alors égale à la puissance du bruit en sortie du NLMS.

Bruit lié aux données dans le NLMS-OCF Dans le NLMS-OCF, comme dans l'équation (2.130), le bruit lié aux données est égal à :

$$Tr(E(\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^t(n)W(n))) = K\sigma_\alpha^2 \sum_{i=0}^{N-1} w_{opt_i}^2 + Km_\alpha^2 (\sum_{i=0}^{N-1} w_{opt_i})^2 \tag{2.152}$$

La valeur de la puissance utile est représentée par l'expression précédente divisée par K.

Bruit lié au filtre dans le NLMS-OCF Comme dans l'équation (2.138), la puissance du bruit lié au calcul du filtre est égale à :

$$Tr(E(\eta(n)\eta(n)^t)) = K(m_\eta^2 + \sigma_\eta^2) \tag{2.153}$$

Dans cette partie, le bruit généré dans l'algorithme NLMS-OCF a été calculé. En analysant les résultats et en comparant au NLMS , la puissance du bruit généré dans le NLMS-OCF est égal à K fois la puissance du bruit généré dans le NLMS. Le NLMS-OCF peut être considéré comme un NLMS d'ordre K. Si la puissance utile est étudiée, la puissance du bruit est égale à celle du NLMS.

Conclusion

Dans cette partie le calcul de la puissance du bruit a été présenté pour les algorithmes de projection affine. Le calcul est plus complexe car les principales données à traiter sont matricielles et non plus vectorielles comme dans le cas du NLMS. Néanmoins, en ne prenant en compte que la puissance utile du bruit lié à la valeur en sortie du système, l'expression est plus simple. D'ailleurs, dans le cas du NLMS-OCF, la puissance utile est égale à la puissance du bruit en sortie du NLMS. Ces matrices sont de tailles NxK pour $K < N$. Dans le cas où $K = N$, le cas des algorithmes des moindres carrés est retrouvé. De ce fait, les algorithmes de projection affine sont définis comme faisant le lien entre le NLMS et les algorithmes des moindres carrés.

2.2.5 Généralisation du modèle

Les quatre algorithmes du gradient traités précédemment sont très proches les uns des autres. De ce fait, un modèle général de calcul de la puissance du bruit pour ceux-ci peut être établi. Elle s'exprime sous la forme suivante :

$$\begin{aligned} Tr(E(b_y(n)b_y^t(n))) &= \sigma_\alpha^2 \sum_{i=0}^{N-1} w_{opt_i}^2 + m_\alpha^2 (\sum_{i=0}^{N-1} w_{opt_i})^2 + m_\eta^2 + \sigma_\eta^2 \\ &+ Tr\Bigg(\Big((1-\lambda^2)\mathbf{I_N} + 2\mu f(X(n))\lambda \mathbf{R}_{xx} \Big)^{-1} . \Big[N\sigma_b^2 \mathbf{R}_{xx} + m_b^2 \Big(\big(1-\lambda\big)\mathbf{I_N} \\ &+ \mu f(X(n))\mathbf{R}_{xx} \Big)^{-1} \Big(\big(1+\lambda\big)\mathbf{I_N} - \mu f(X(n))\mathbf{R}_{xx} \Big) (\mathbf{1_N}\mathbf{R}_{xx}) \Big] \Bigg) \end{aligned} \quad (2.154)$$

Le terme $f(X(n))$ est une fonction des données d'entrée valant 1 pour le LMS et $\frac{1}{Tr(\mathbf{R}_{xx})}$ pour le NLMS. Le coefficient λ est le coefficient de leakage différent de 1 seulement pour leaky-LMS. Cette expression permet d'obtenir une expression générale valable pour l'ensemble des algorithmes du gradient et s'adaptant pour chacun d'eux.

2.3 Qualité des modèles proposés

Dans la partie précédente, des modèles dédiés d'évaluation de la puissance du bruit pour différents types de système ont été présentés. Dans cette partie, la qualité de ces modèles est analysée au moyen de diverses expérimentations. Tout d'abord, les conditions d'expérimentation sont définies. Ensuite, les modèles destinés aux filtres adaptatifs sont validés sur les algorithmes LMS, Leaky-LMS et APA.

2.3.1 Conditions d'expérimentations

Dans cette section, les diverses conditions mises en œuvre pour évaluer la qualité des modèles sont présentées. Dans un premier temps, la procédure de test est définie. Elle permet de présenter le mode opératoire utilisé pour estimer la précision de nos approches. La qualité des modèles est estimée en utilisant l'erreur relative. Dans un second temps, le signal d'entrée utilisé pour les expérimentations est explicité. Le choix du signal permet de tester les différentes configurations de signaux existants. Finalement, les conditions du domaine de filtrage choisi pour les filtres adaptatifs sont définies.

Procédure de test pour mesurer l'erreur relative

Dans cette section, la méthode d'évaluation de la qualité des modèles est définie. D'une part, le modèle défini dans les parties précédentes est appliqué et fournit la valeur de la puissance du bruit estimée en sortie du système. Cette valeur P_{est} correspond à la valeur obtenue analytiquement par notre approche. Pour valider ce résultat, celui-ci doit être comparé à la valeur réelle de la puissance du bruit en sortie du système P_{sim}. La valeur réelle de la puissance du bruit P_{sim} est déterminée par simulation selon la technique illustrée à la figure 2.9. Pour ce faire, l'application est simulée en virgule fixe et en virgule flottante sur l'outil Matlab/Simulink. La virgule flottante génère également des bruits comme l'arithmétique virgule fixe. Cependant, les bruits générés par l'arithmétique virgule flottante sont négligeables par rapport à ceux générés par l'arithmétique virgule fixe étant donnée la largeur des données en virgule fixe. De ce fait, la virgule flottante est considérée comme la référence. Ainsi, l'écart entre la sortie du système simulé en virgule fixe et celle du système simulé en virgule flottante représente le bruit réel en sortie du système. Ensuite, ses paramètres statistiques sont calculés pour pouvoir déterminer sa puissance réelle P_{sim}. La

puissance du bruit P_{est} estimée par notre modèle peut alors être comparée à la puissance réelle P_{sim} pour mesurer sa qualité. Le nombre d'échantillons pris en compte pour les calculs statistiques est 10000.

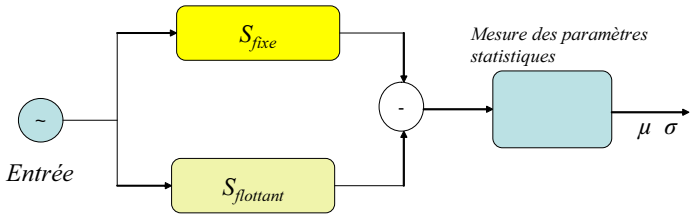

FIG. 2.9 – *Procédure d'évaluation de la qualité de l'estimation*

La qualité des résultats fournis par notre approche est mesurée au moyen de l'erreur relative. Celle-ci est définie par la relation suivante :

$$Err = \frac{|P_{sim} - P_{est}|}{P_{sim}} \tag{2.155}$$

Elle correspond à l'écart entre la puissance réelle P_{sim} et la puissance estimée par le modèle P_{est} relativement à la puissance réelle P_{sim}. Elle se représente sous la forme d'un pourcentage d'erreur commise sur l'estimation de la valeur réelle de la puissance. Les résultats obtenus doivent être interprétés selon d'autres mesures comme le nombre de bits significatifs ou l'écart de la puissance en dB. Le nombre de bits significatifs représente le nombre de bits associés aux formats des données n'étant pas affectés par le bruit. L'écart en dB définit la différence entre la puissance estimée par notre approche et la puissance réelle en dB comme l'exprime la relation suivante :

$$P_{dB} = 10log(P_{sim}) - 10log(P_{est}) \tag{2.156}$$

Par exemple, une erreur relative de 300 % semble énorme. En réalité, cela correspond à seulement 1 bit significatif supplémentaire estimé par notre approche. De la même façon, 1 bit significatif en moins représente 75 % d'erreur relative et un écart de 6 dB. Ainsi, les résultats obtenus avec l'erreur relative doivent être interprétés pour juger la qualité du modèle proposé.

Signal d'entrée

Dans cette section, le signal d'entrée choisi est présenté. Le signal d'entrée est du type AR(1) (Auto-Régressif d'ordre 1). Le principal avantage de ce type de signal est de pouvoir contrôler et faire varier sa corrélation et ainsi, de tester le modèle sur des bruits blancs mais aussi des signaux très corrélés. L'équation de ce signal est la suivante :

$$x(n+1) = \beta x(n) + u(n) \tag{2.157}$$

où $u(n)$ est un bruit blanc centré de puissance σ_u^2 et $\beta \in [0,1[$. Lorsque β tend vers 0, alors le signal $x(n)$ est un bruit blanc. A l'inverse, quand β tend vers 1 le signal est très corrélé. De ce fait, ce type de signal permet d'effectuer des simulations sur des signaux non-corrélés, moyennement corrélés où fortement corrélés. Différents termes statistiques de ce signal peuvent être calculés. Sa moyenne est nulle et sa puissance est définie par l'expression suivante :

$$E(x(n)^2) = \frac{\sigma_u^2}{1-\beta^2} \tag{2.158}$$

L'expression suivante en est déduite :

$$E(x(n)^2) = \begin{cases} \sigma_u^2 & \beta \longrightarrow 0 \\ \infty & \beta \longrightarrow 1 \end{cases} \tag{2.159}$$

Si β tend vers 0 la puissance du signal tend vers la puissance du bruit blanc $u(n)$. Si β tend vers 1, la puissance du signal tend vers l'infini. La puissance de ce signal est déterminée par la valeur de β. Sa fonction d'autocorrélation R_{xx} est définie dans l'expression suivante pour $k \in [0 : N-1]$.

$$\begin{aligned} R_{xx}(k) = E(x(n)x(n-k)) &= E(\beta x(n-k)x(n-1) + x(n-k)u(n-1)) \\ &= \beta E(x(n-1)x(n-k)) + E(x(n-k))\underbrace{E(u(n-1))}_{=0} \\ &= \beta R_{xx}(k-1) \end{aligned} \tag{2.160}$$

Cette formule de récurrence aboutit à l'expression suivante :

$$R_{xx}(k) = \frac{\beta^k \sigma_u^2}{1-\beta^2} \tag{2.161}$$

Si β tend vers 0, la fonction d'autocorrélation est diagonale comme celle du bruit blanc. Si β tend vers 1, la matrice est plus homogène démontrant la corrélation du signal.

Domaine de filtrage

Les différentes expérimentations sont effectuées à l'aide du signal d'entrée présenté précédemment et sont mises en œuvre sur des filtres adaptatifs. Ces derniers sont utilisés dans divers domaines comme l'annulation d'échos, l'égalisation ou encore l'identification de canal. Les tests présentés pour les filtres adaptatifs utilisent le domaine de l'égalisation. Pour ce faire, les données d'entrée sont filtrées par le filtre de référence H_{opt}. Puis, un bruit blanc $b(n)$ gaussien centré de puissance σ_b^2 est ajouté pour aboutir au signal désiré $y(n)$. Le schéma 2.10 illustre le système mis en œuvre.

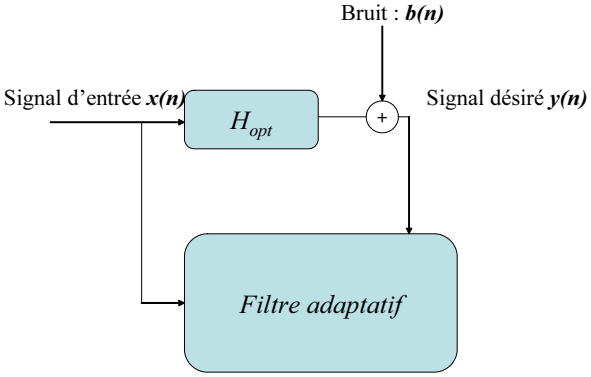

FIG. 2.10 – *Domaine d'utilisation des filtres adaptatifs*

Le filtre de référence choisi est un filtre IIR d'ordre 2 défini par la relation suivante :

$$H_{opt} = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{2.162}$$

Les coefficients du filtre adaptatif convergent donc vers ce filtre. Le choix des différents coefficients de ce filtre optimal permet de faire varier les valeurs des coefficients optimaux du filtre adaptatif.

2.3.2 Evaluation de la qualité du modèle dédié à l'algorithme LMS

Dans cette section, le modèle dédié à l'algorithme LMS est évalué au moyen de diverses expérimentations. Dans un premier temps, l'analyse de la convergence de l'algorithme en fonction du nombre de bits destiné au codage des coefficients est présentée. Dans un deuxième temps, l'étude de l'influence des différents bruits présents dans le système est effectuée. L'erreur relative commise par notre approche est évaluée dans un troisième temps. Finalement, une comparaison avec les autres méthodes existantes est mise en œuvre.

Analyse de la convergence de l'algorithme

La convergence du filtre nécessite une mise à jour correcte des coefficients. Cette dernière est assurée par la multiplication entre l'erreur courante et le signal d'entrée. Le signal sortant de la conversion en sortie de la multiplication (lié au bruit $\gamma(n)$) ne doit donc pas être nul. Ce signal doit être d'amplitude supérieure au pas de quantification comme l'expression (2.18), reprise par la suite, l'indique.

$$Nb = -\frac{1}{2} log_2 \left(\frac{\mu^2 E(x(n)^2) \xi}{4 w_{max}^2} \right) \tag{2.163}$$

avec ξ l'erreur quadratique moyenne et w_{max} la valeur maximale des coefficients. La valeur Nb obtenue définit le nombre de bits minimum associé au codage des coefficients assurant une convergence correcte de ceux-ci.

Pour vérifier la validité de cette égalité, un test a été effectué. Soit un filtre de taille $N = 1$ pour pouvoir visualiser la convergence de ce coefficient. Les autres constantes sont fixées aux valeurs $\beta = 0.85$, $\mu = 2^{-5}$ et la quantification est effectuée par arrondi. Avec l'expression précédente, le nombre de bits minimum pour le codage du coefficient assurant une convergence correcte est de $Nb = 5.8$. En théorie, si le nombre de bits pour la simple précision est inférieur ou égal à 5, le filtre ne converge pas correctement. La figure 2.11 représente l'évolution du coefficient en fonction du nombre de bits alloués à son codage. Dans le cas où 16 bits sont prévus pour son codage, son évolution est tout à fait correcte. Lorsque le nombre de bits passe à 6, le filtre converge toujours mais les mises à jour sont très visibles. Le nombre de bits étant faible, celles-ci s'effectuent lentement, mais le résultat est satisfaisant. Lorsque le nombre de bits est fixé à 5, la convergence n'est pas correcte car le coefficient ne converge pas. Sur cet exemple, l'expression du nombre de bits permet d'aboutir à des résultats corrects. Celle-ci n'a pas pour but de donner la valeur exacte du nombre minimal de bits mais plutôt un ordre d'idée permettant d'assurer de manière correcte la convergence du filtre. Par la suite, les cas traités assurent la convergence du filtre.

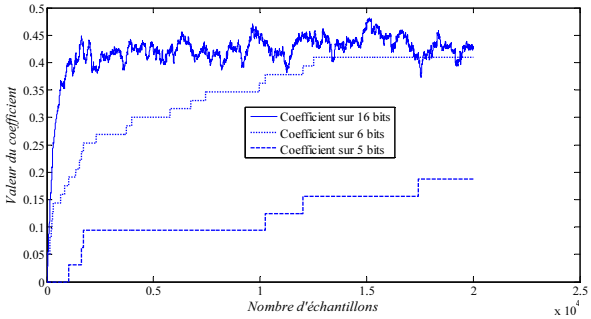

FIG. 2.11 – *Evolution du coefficient pour un nombre de bits variant entre 5, 6 et 16*

Influence des différents bruits

Dans cette section, une étude est faite sur l'influence des différents bruits présents au sein de l'application LMS. L'expression (2.71) page 49, montre que la puissance du bruit en sortie de l'algorithme LMS est composée de trois termes. Tout d'abord, le bruit $\alpha(n)$ lié aux données d'entrée. Ensuite, le bruit $\rho(n)$ sur les coefficients. Enfin le bruit $\eta(n)$ modélisant l'ensemble des bruits générés dans le filtre. Chacun de ces trois termes influe de façon plus ou moins importante sur la puissance du bruit en sortie du système. La figure 2.12, illustre la puissance de chacun de ces trois termes en sortie du système pour des lois de quantification par troncature et arrondi. Les résultats obtenus pour l'arrondi convergent sont du même ordre que l'arrondi classique. Pour cette expérimentation, la taille du filtre varie entre $N = 2$ et $N = 64$ pour une valeur de pas d'adaptation $\mu = \frac{\mu_{max}}{2}$. Le bruit dont la contribution est la plus importante correspond au bruit sur les coefficients dans le cas de la loi de quantification par troncature et arrondi. La puissance de ce bruit augmente avec la taille du filtre. Pour ce cas-ci, la puissance de ce bruit en sortie est de l'ordre de 10^{-5} pour la troncature et de 10^{-8} pour l'arrondi. La puissance des autres bruits est environ 100 à 1000 fois plus faible. Le bruit sur les coefficients prédomine donc au sein du système. De plus, la puissance des bruits diffère de manière importante selon la loi de quantification utilisée. Cette différence s'explique par la prise en compte de la moyenne des bruits dans l'expression de la puissance de ces termes. La prise en compte de la moyenne des termes de bruit est primordiale pour obtenir un modèle général et valable pour tous les cas. Pour les lois de quantification par arrondi et arrondi convergent, la puissance du bruit sur les coefficients prédomine également.

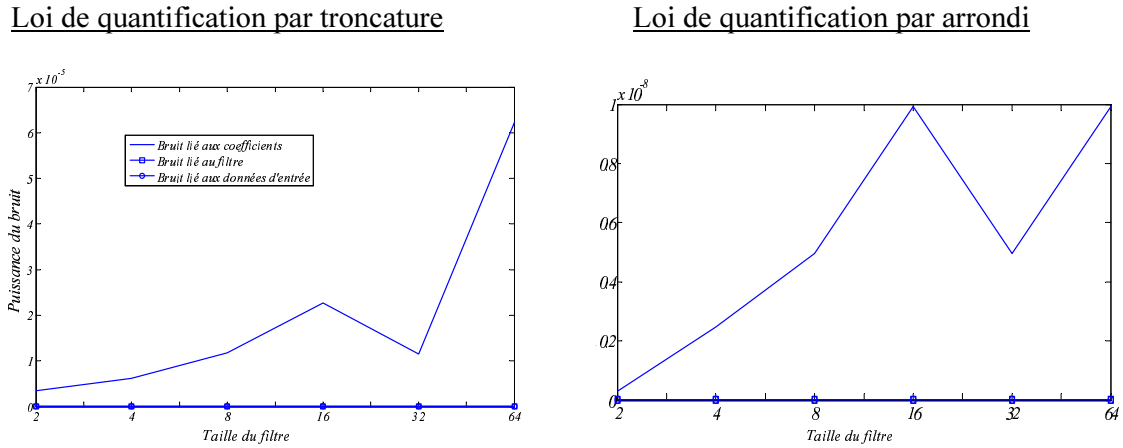

FIG. 2.12 – *Influence des trois termes de bruit*

Analyse de l'erreur relative d'estimation

Dans cette section, l'analyse de la qualité de notre modèle d'estimation est effectuée. Dans un premier temps, l'estimation est étudiée en fonction de la taille du filtre pour des lois de quantification par troncature et arrondi.

La figure 2.13 montre l'erreur relative associée à notre modèle pour une loi de quantification par arrondi avec une taille de filtre variant entre $N = 2$ et $N = 128$. Le signal d'entrée est soit très corrélé $\beta = 0.95$, soit moyennement corrélé $\beta = 0.5$, soit très peu corrélé $\beta = 0.05$. Les valeurs du pas d'adaptation μ sont fixées à $\mu = \frac{\mu_{max}}{2}$. Pour les 3 cas considérés, l'erreur relative commise se situe entre 8% et 25%. Ces résultats sont très corrects car ils correspondent à un écart de seulement 1 à 2 dB sur le RSBQ. De plus, la corrélation du signal n'influe pas sur la qualité de l'estimation.

La figure 2.14 illustre les mêmes expérimentations effectuées pour une loi de quantification par troncature. L'erreur relative est inférieure à 30%. Néanmoins, l'erreur commise pour des données d'entrée très peu corrélées $\beta = 0.05$ est plus faible que pour les deux autres cas. L'erreur relative

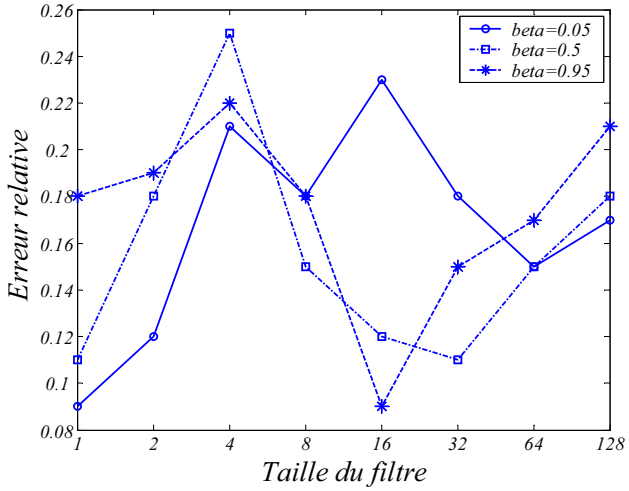

FIG. 2.13 – *Erreur relative pour l'algorithme LMS en fonction de N pour une loi de quantification par arrondi*

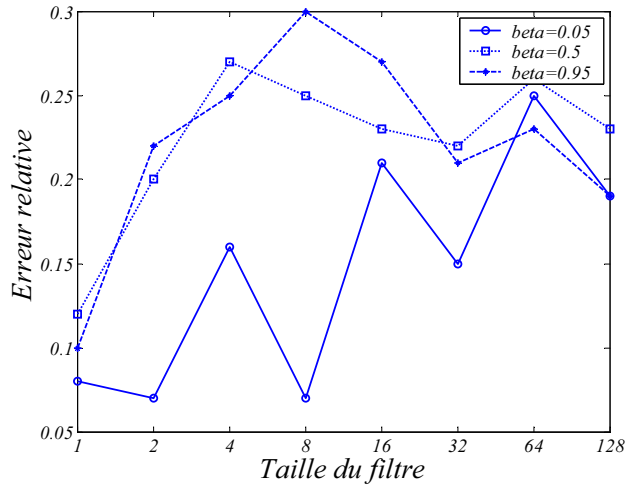

FIG. 2.14 – *Erreur relative pour l'algorithme LMS en fonction de N pour une loi de quantification par troncature*

obtenue permet de montrer que l'hypothèse de non-corrélation entre les données d'entrées $x(n)$ et le bruit sur les coefficients $\rho(n)$ utilisée dans notre modèle n'affecte pas la qualité de l'estimation. Cependant, moins les données d'entrée $x(n)$ sont corrélées, plus cette hypothèse se vérifie car l'erreur relative est moindre. Les expérimentations montrent que l'hypothèse peut être utilisée.

Le modèle a été validé pour différentes valeurs de N mais pour une valeur de μ fixe. Ensuite, l'objectif est de valider le modèle pour différentes valeurs de μ. Ainsi, la valeur de N est fixée à 16. Le terme μ est normalisé par rapport à sa valeur maximale μ_{max} et $\frac{\mu}{\mu_{max}}$ varie de 0.1 à 0.9. La figure 2.15 représente les résultats obtenus pour une loi de quantification par troncature.

L'erreur relative obtenue augmente en fonction de $\frac{\mu}{\mu_{max}}$ pour aboutir à des valeurs inférieures à 45%. L'augmentation de cette erreur est due au rapprochement de μ par rapport à sa valeur maximale μ_{max}. Pour une valeur supérieure à 0.9, les résultats obtenus divergent. Néanmoins et en pratique, les valeurs de μ utilisées restent très inférieures à μ_{max}. Ainsi, le modèle est correct pour ces valeurs de μ. La figure 2.16 représente les mêmes expérimentations effectuées pour une loi de quantification par arrondi. Les résultats obtenus sont du même ordre que ceux pour la troncature. L'erreur relative est inférieure à 50%. Pour des valeurs de μ inférieures à $\frac{\mu_{max}}{2}$, l'erreur est

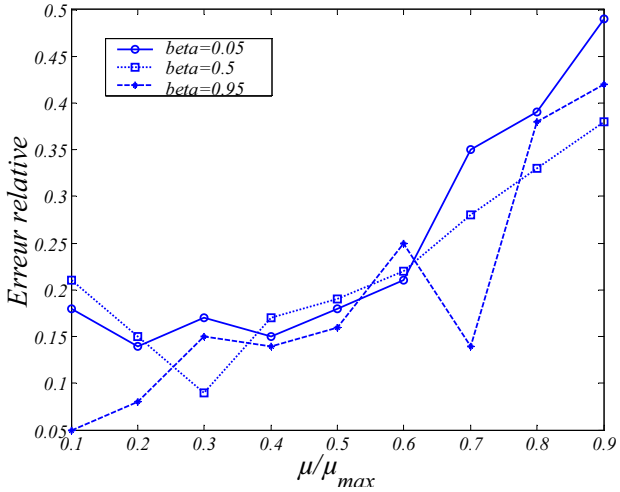

FIG. 2.15 – *Erreur relative pour l'algorithme LMS en fonction de μ pour une loi de quantification par troncature*

inférieure à 20%.

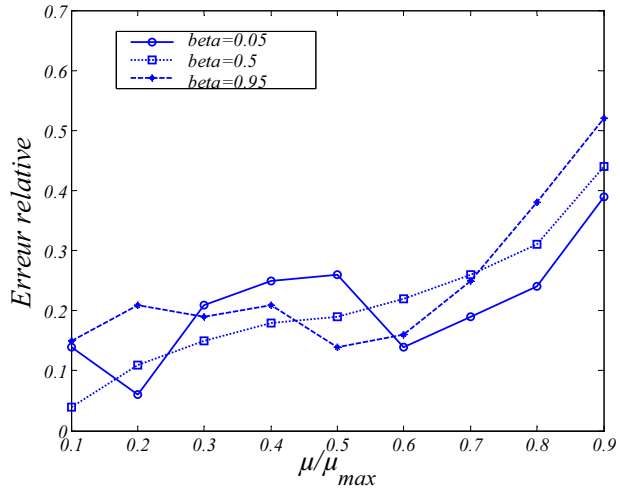

FIG. 2.16 – *Erreur relative pour l'algorithme LMS en fonction de μ pour une loi de quantification par arrondi*

Comparaison avec les modèles présentés dans [19] et [10]

Pour des valeurs de μ utilisées en pratique, l'erreur relative reste inférieure à 25%. Ce modèle fournit de bons résultats. Pour terminer, le modèle est comparé aux résultats obtenus avec les modèles développés par M.Bellanger [10] et C.Caraiscos [19]. L'expérimentation est effectuée avec un signal moyennement corrélée ($\beta = 0.5$) et la taille du filtre varie de 2 à 128. La loi de quantification choisie est l'arrondi convergent car ces deux modèles sont développés dans ce cadre. La figure 2.17 montre l'erreur relative obtenue par les trois modèles en fonction du pas d'adaptation. Les résultats sont très proches les uns des autres. Néanmoins, le modèle de M.Bellanger présente une erreur relative plus importante due à la non prise en compte des bruits $\alpha(n)$ sur les données d'entrée. Les écarts entre notre modèle et celui de C.Caraiscos sont très proches (de l'ordre de 1% en moyenne). Ces résultats justifient les simplifications effectuées par notre approche. Celles-ci permettent de simplifier l'expression sans diminuer la qualité de l'estimation.

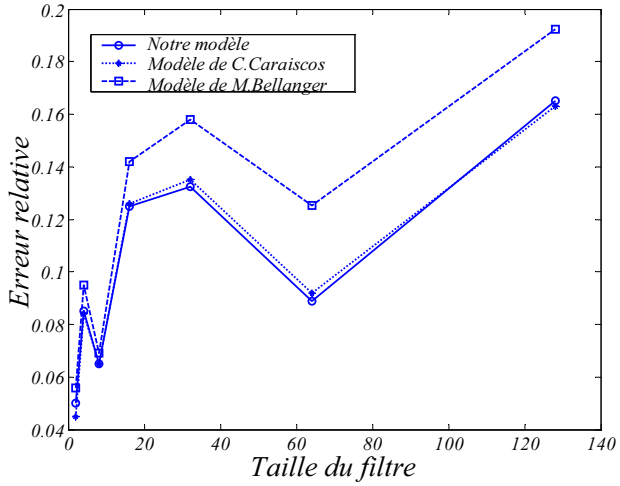

FIG. 2.17 – *Erreur relative sur la puissance du bruit pour les trois modèles*

Dans le cas où les bruits d'entrée ne sont pas pris en considération, les trois modèles fournissent des résultats équivalents. Dans ce cas, le modèle de Bellanger est plus adéquate que celui de Caraiscos car son expression est plus simple et fournit un résultat du même ordre.

2.3.3 Evaluation de la qualité du modèle dédié aux Algorithmes de Projection Affine

Cette partie présente les expérimentations effectuées pour valider le modèle de puissance de bruit pour les filtres adaptatifs basés sur les Algorithmes de Projection Affine (APA). Les conditions de simulations sont identiques à celles de la partie précédente. Les termes N et K désignent respectivement, la taille du filtre et l'ordre de projection. Dans un premier temps, les résultats sont présentés pour une loi de quantification par arrondi. La figure 2.18 est définie pour N entre 1 et 20, K entre 1 et $N-1$ et pour les données d'entrée orthogonales. De ce fait, l'algorithme simulé est le NLMS-OCF dans le cas d'une quantification par arrondi. L'erreur relative obtenue est inférieure à 38% et de valeur moyenne égale à 19 %, permettant ainsi de montrer que le modèle pour le NLMS-OCF permet d'obtenir une estimation précise.

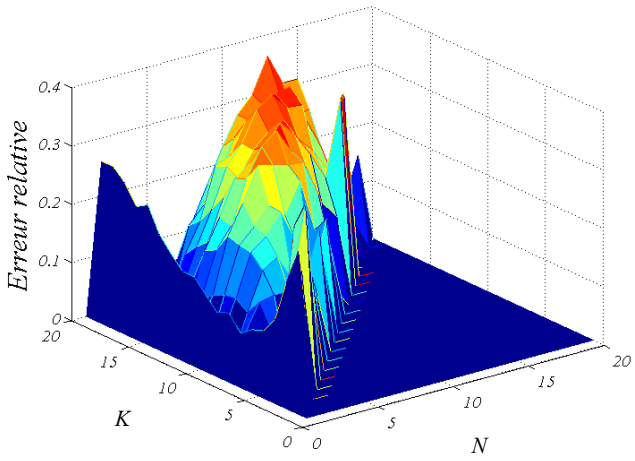

FIG. 2.18 – *Erreur relative pour le NLMS-OCF dans le cadre d'une quantification par arrondi*

	Arrondi	**Troncature**
Erreur moyenne	14.52%	16.77%
Erreur maximale	38%	36%

TAB. 2.1 – *Valeurs maximales et moyennes de l'erreur relative pour les algorithmes APA*

La figure 2.19 montre la qualité du modèle pour N entre 1 et 20, K entre 1 et $N-1$, pour les données d'entrée très corrélées ($\beta = 0.9$) dans le cadre d'une loi de quantification par troncature. L'erreur relative obtenue est inférieure à 36%. Le résultat est correct car cela représente une différence de seulement de 2 dB sur le RSBQ. Ainsi, ce nouveau modèle développé fournit une estimation précise pour les entrées très corrélées. Pour des entrées peu corrélées, une erreur relative inférieure à 30% est obtenue.

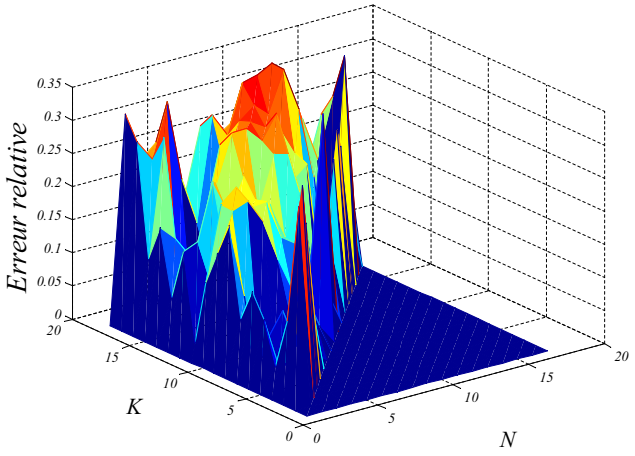

FIG. 2.19 – *Erreur relative pour les APA dans le cadre d'une quantification par troncature*

Le modèle pour déterminer analytiquement la puissance de bruit en sortie des APA est validé par des simulations pour la version générale des APA mais aussi pour l'algorithme NLMS-OCF. Le tableau 2.1 résume les valeurs moyennes et maximales de l'erreur relative ainsi obtenues.

2.3.4 Evaluation du modèle dédié à l'algorithme Leaky-LMS

Dans cette partie, la qualité du modèle développé pour l'algorithme Leaky-LMS est testée. Le coefficient de leakage est fixé à la valeur 0.999. La figure 2.20 illustre l'erreur relative obtenue pour une loi de quantification par arrondi en fonction de la taille N du filtre et de la valeur de μ. Les résultats obtenus montrent une erreur inférieure à 10%. La figure 2.21 représente l'erreur relative obtenue avec la taille du filtre variant de 2 à 128 pour une valeur de μ fixé à $\frac{\mu_{max}}{2}$ et une loi de quantification par troncature. Les résultats obtenus sont très bons car l'erreur relative est inférieure à 7%. Cette expérimentation permet de valider le modèle dédié à l'algorithme Leaky-LMS. Les valeurs moyennes et maximales de l'erreur relative sont résumées dans le tableau 2.2.

2.3.5 Conclusion

Dans cette section, les modèles dédiés aux filtres adaptatifs présentés dans ce chapitre ont été validés par diverses expérimentations. Les erreurs relatives obtenues sont très correctes car en général inférieures à 35%. Pour cette valeur, l'erreur représente un écart de seulement 2 dB. Les

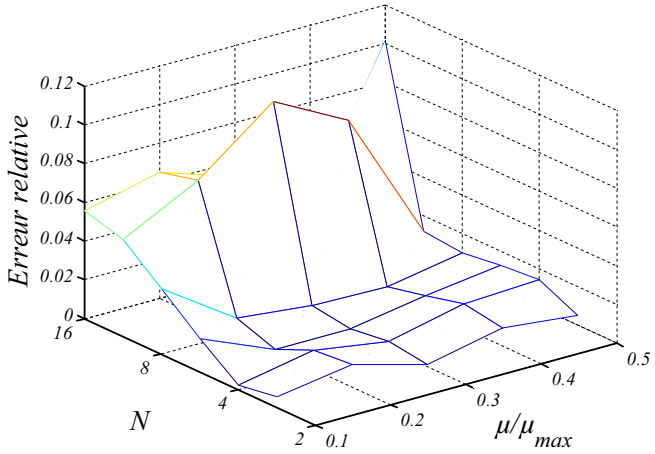

FIG. 2.20 – *Erreur relative pour l'algorithme Leaky-LMS en fonction de la taille du filtre et du pas d'adaptation pour une loi par arrondi*

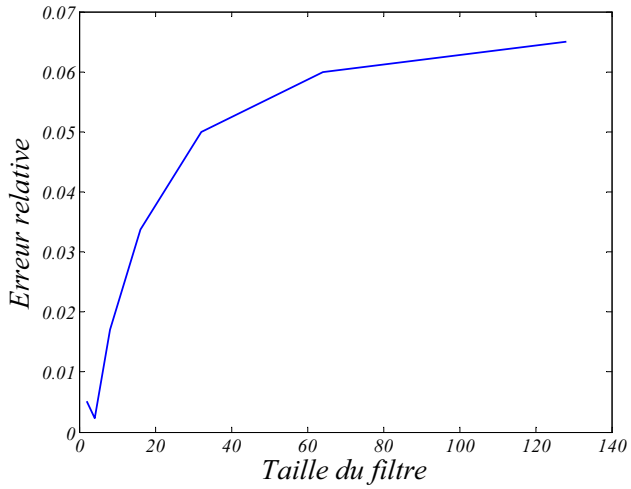

FIG. 2.21 – *Erreur relative pour l'algorithme Leaky-LMS en fonction de la taille du filtre pour une loi par troncature*

cas où l'erreur est supérieure à cette valeur proviennent des limites mêmes des algorithmes. Les expérimentations ont été effectuées pour les algorithmes LMS, APA et Leaky-LMS. Pour chacun des ces trois types d'algorithme, l'erreur relative a été mesurée pour les lois de quantification par arrondi et troncature. Les valeurs obtenues de l'erreur relative permettent donc de montrer la qualité de ces différents modèles.

2.4 Conclusion

Dans cette partie, des modèles analytiques d'évaluation de la précision en virgule fixe pour les systèmes non-LTI récursifs représentés par les filtres adaptatifs ont été présentés. Dans un premier temps les filtres adaptatifs ont été introduits et les différentes méthodes étudiant leur implantation en virgule fixe ont été explicitées. Dans un second temps, un modèle analytique de calcul de la puissance du bruit de quantification en sortie du système a été développé pour chaque type d'algorithme adaptatif. Ces modèles diffèrent les uns des autres car ils prennent en compte les caractéristiques propres des systèmes. Dans la dernière partie, la qualité des modèles proposés a été évaluée au moyen de différentes expérimentations.

	Arrondi	**Troncature**
Erreur moyenne	2.62%	3%
Erreur maximale	10%	6.5%

TAB. 2.2 – *Valeurs maximales et moyennes de l'erreur relative pour l'algorithme Leaky-LMS*

Dans le premier chapitre, des modèles ont été présentés pour les systèmes LTI et les systèmes non-LTI non-récursifs. Ce deuxième chapitre permet d'avoir à disposition des modèles analytiques dédiés pouvant être utilisés par exemple dans la génération de composants matériels dédiés. Cependant, ces approches spécifiques ne sont pas généralisables aux autres systèmes. Ainsi, dans le chapitre suivant, une approche analytique de l'évaluation de la précision, valide pour tout type de système, est mise en œuvre.

Chapitre 3

Méthode générale

Sommaire

La conversion de virgule flottante en virgule fixe doit satisfaire les contraintes de précision assurant l'intégrité de l'algorithme à convertir. Un outil de conversion en virgule fixe doit supporter tous les types de système. Les modèles présentés dans la partie précédente sont valides uniquement pour des algorithmes particuliers. La démarche suivie pour obtenir ces modèles est liée aux applications considérées. Ainsi, dans le but d'avoir un modèle général, une nouvelle approche doit être définie. Cette approche doit permettre par la suite de retrouver les résultats obtenus dans la partie précédente en appliquant le nouveau modèle aux systèmes particuliers.

Ainsi, dans cette partie, une méthodologie générale d'évaluation automatique de la précision des systèmes est élaborée. Les systèmes considérés sont de type quelconque et composés d'opérations arithmétiques (addition, soustraction, multiplication et division). La méthode adoptée est la suivante. La propagation de chaque source de bruit est modélisée sous la forme d'une pseudo fonction de transfert. Pour déterminer celle-ci, une modélisation du système est développée. Ensuite, le bruit de sortie est la somme de chacune des contributions de chaque source de bruit. A partir de cette formulation, la puissance du bruit de sortie et toutes les autres caractéristiques statistiques peuvent être calculées. D'autre part, l'algorithme à convertir peut n'être qu'une partie d'un système plus complet. Le modèle développé doit être capable de cascader plusieurs systèmes.

Ainsi le modèle général doit vérifier la contrainte suivante. Connaissant les caractéristiques statistiques des bruits d'entrée, il doit permettre de retrouver les caractéristiques de la contribution de ces bruits de sortie. Ainsi, ce modèle est capable de mettre en œuvre la composition de différents systèmes.

Le plan de ce chapitre est le suivant. Dans une première partie, la modélisation des bruits est définie. Elle permet de présenter les termes statistiques définissant les sources de bruit et leur modèle de propagation au sein des opérations arithmétiques. Dans une seconde partie, la modélisation des systèmes et leur composition sont présentées. Chaque bloc composant le système est modélisé analytiquement. Cette démarche permet de mettre en œuvre une méthodologie étudiant la composition des systèmes. Dans un troisième temps, le calcul de la puissance du bruit en sortie du système traité est développé. L'expression obtenue est analytique et dépend uniquement des statistiques des bruits d'entrée et du système considéré. Une approche par prédiction linéaire est présentée permettant de réduire la complexité des calculs des statistiques des bruits de sortie. Dans une quatrième partie, l'application LMS est traitée en détails. Finalement, dans un cinquième partie, la qualité de la méthode proposée est évaluée au moyen de diverses expérimentations sur différents types de système. Avant tout cela, les notations mises en place sont définies.

Pour l'étude de la précision suivante, le système à étudier est composé de N_e bruits d'entrée et de N_s bruits de sortie. Chaque bruit d'entrée est noté $b_i(n)$ où n est l'instant donné et $i \in [1 : N_e]$. Chaque terme de sortie est noté $b_{yu}(n)$ où $u \in [1 : N_s]$. De plus, les notations permettent de différencier les dimensions des données. Les scalaires sont notés en minuscules ($x(n)$), les vecteurs en majuscules ($X(n)$) et les matrices en gras majuscules ($\mathbf{X}(n)$).

3.1 Modélisation des bruits

Cette section permet de définir les termes statistiques des bruits et d'étudier leur mode de propagation à travers les différentes opérations rencontrées.

3.1.1 Caractéristiques statistiques des bruits

Dans cette partie, les caractéristiques statistiques des différents bruits sont définies. Dans un système quelconque, le bruit en entrée peut être un bruit généré ou un bruit provenant de la sortie d'un autre système. Ainsi ce bruit n'a plus les caractéristiques du bruit blanc. De ce fait, la connaissance des caractéristiques statistiques des bruits est nécessaire. Plusieurs sont définies (Fonction Densité de Probabilité (FDP), moyenne, autocorrélation....). Le bruit d'entrée considéré est noté $b_i(n)$ dans tous les cas bien qu'il puisse être scalaire, vectoriel ou matriciel. Ses dimensions sont notées $M_i \mathrm{x} Ni$.

Densité de probabilité

Un bruit $b_i(n)$ est déterminé par sa Fonction de Densité de Probabilité (FDP) notée $f_{b_i}(x)$ représentant sa répartition sur $\mathbb{R}$. Si b_i est un scalaire alors sa FDP est une fonction : $\mathbb{R} \rightarrow \mathbb{R}$. Sinon, sa FDP est une fonction multidimensionnelle : $\mathbb{R}^{M_i \mathrm{x} N_i} \rightarrow \mathbb{R}$. Dans le cas de bruits générés, leur FDP correspond à une loi uniforme. Dans le cas où le bruit n'est pas blanc, sa FDP est modifiée par le système traversé précédemment par le bruit.

Moyenne du bruit

Le bruit $b_i(n)$ est de taille $M_i \mathrm{x} N_i$ $\forall i \in [1 : N_e]$. Sa moyenne possède les mêmes dimensions et est notée $m_{b_i} = E(b_i(n))$. Elle correspond à la moyenne de chacun des termes composant $b_i(n)$. En effet, $b_i(n)$ est de la forme :

$$b_i(n) = \begin{pmatrix} b_{i_{11}}(n) & b_{i_{21}}(n) & \dots & b_{i_{1N_i}}(n) \\ b_{i_{11}}(n) & b_{i_{22}}(n) & \dots & b_{i_{2N_i}}(n) \\ \dots & \dots & \dots & \dots \\ b_{i_{M_i1}}(n) & b_{i_{M_i2}}(n) & \dots & b_{i_{M_iN_i}}(n) \end{pmatrix} \tag{3.1}$$

Donc m_{b_i} correspond à la moyenne de chacun des termes composant le bruit $b_i(n)$. Soit $\mu_{b_{i_{kj}}}$ la moyenne du terme $b_{i_{kj}}(n)$ de la matrice, alors m_{b_i} est défini par l'expression suivante :

$$m_{b_i} = \begin{pmatrix} \mu_{b_{i_{11}}} & \mu_{b_{i_{12}}} & \dots & \mu_{b_{i_{1N_i}}} \\ \mu_{b_{i_{21}}} & \mu_{b_{i_{22}}} & \dots & \mu_{b_{i_{2N_i}}} \\ \dots & \dots & \dots & \dots \\ \mu_{b_{i_{M_i1}}} & \mu_{b_{i_{M_i2}}} & \dots & \mu_{b_{i_{M_iN_i}}} \end{pmatrix} \tag{3.2}$$

De cette façon, m_{b_i} possède des dimensions identiques à celles de $b_i(n)$ alors que $\mu_{b_{i_{kj}}}$ est un scalaire. Si $b_i(n)$ est un scalaire alors l'égalité suivante est vérifiée $m_{b_i} = \mu_{b_i}$.

Auto-corrélation du bruit

Deux notions différentes sont associées à l'autocorrélation. Tout d'abord, si le bruit $b_i(n)$ est un vecteur, alors l'auto-corrélation est la corrélation entre les différentes valeurs de ce vecteur, ceci correspond à la corrélation spatiale. D'autre part, la notion de corrélation temporelle entre les données correspond à la relation entre les vecteurs pris à des instants différents. De ce fait, plusieurs matrices d'auto-corrélation notées $\mathbf{R}_{b_ib_i}(\theta)$ sont définies de la manière suivante :

$$\mathbf{R}_{b_ib_i}(\theta) = E(b_i(n)b_i^t(n-\theta)) \tag{3.3}$$

Ces matrices sont de taille M_ixM_i et $\theta \in \mathbb{N}$. Dans le cas où $\theta = 0$, la formulation de la corrélation spatiale est retrouvée.

Inter-corrélation des bruits

L'inter-corrélation entre deux bruits $b_i(n)$ et $b_j(n)$ différents ($i \neq j$) est la matrice définie par :

$$\mathbf{R}_{b_ib_j}(\theta) = E(b_i(n)b_j^t(n-\theta)) \tag{3.4}$$

D'autre part, si les deux bruits $b_i(n)$ et $b_j(n)$ considérés sont non-corrélés, leur matrice d'intercorrélation de taille M_ixM_j est obtenue à partir des moyennes m_{b_i} et m_{b_j} associées à chaque bruit.

$$\mathbf{R}_{b_ib_j}(\theta) = m_{b_i}m_{b_j}^t \tag{3.5}$$

Cette inter-corrélation est possible uniquement si les bruits ont des tailles compatibles. L'égalité $N_i = N_j$ doit être respectée et dans ce cas la matrice d'inter-corrélation a pour dimensions M_ixM_j.

Covariance du bruit

La matrice de covariance $\mathbf{C}_{b_ib_i}(\theta)$ d'un bruit $b_i(n)$ correspond à l'autocorrélation du bruit centré. Son expression est donnée par la relation suivante :

$$\mathbf{C}_{b_ib_i}(\theta) = E\left((b_i(n) - m_{b_i})(b_i^t(n-\theta) - m_{b_i}^t)\right) \tag{3.6}$$

L'autocorrélation et la covariance sont reliées par l'expression suivante :

$$\mathbf{R}_{b_ib_i}(\theta) = \mathbf{C}_{b_ib_i}(\theta) + m_{b_i}m_{b_i}^t \tag{3.7}$$

Dans le cas où $\theta = 0$, la covariance est simplement la variance du bruit. L'intercovariance de deux bruits $b_i(n)$ et $b_j(n)$ est égale à :

$$\begin{aligned}\mathbf{C}_{b_i b_j}(\theta) &= E\left((b_i(n) - m_{b_i})(b_j^t(n-\theta) - m_{b_j}^t)\right) \\ &= \mathbf{R}_{b_i b_j}(\theta) - m_{b_i} m_{b_j}^t\end{aligned} \tag{3.8}$$

Si les deux bruits b_i et b_j sont non-corrélés, alors leur intercovariance est nulle.

Cas d'un bruit blanc

Dans cette partie, le cas des bruits de quantification est traité. Ces bruits sont blancs et non-corrélés entre eux. Le bruit $b_i(n)$ est alors composé de termes ayant tous les mêmes caractéristiques statistiques. Le terme $\sigma_{b_i}^2$ désigne la variance des échantillons de $b_i(n)$ et μ_{b_i} leur moyenne. En notant $\mathbf{1_N}$ la matrice unité de taille N et $\mathbf{I_N}$ la matrice identité, les termes statistiques de $b_i(n)$ peuvent s'écrire sous la forme suivante :

$$\mathbf{C}_{b_i b_i}(\theta) = \sigma_{b_i}^2 \mathbf{I_N} \delta(\theta) \tag{3.9}$$

$$\mathbf{R}_{b_i b_i}(\theta) = \sigma_{b_i}^2 \mathbf{I_N} \delta(\theta) + \mu_{b_i}^2 \mathbf{1_N} \tag{3.10}$$

$$\mathbf{R}_{b_i b_j}(\theta) = \mu_{b_i} \mu_{b_j} \mathbf{1_N} \tag{3.11}$$

Conclusion

Les termes statistiques des bruits ont été définis. A partir des connaissances des termes en entrée du système, l'objectif est de déterminer ces mêmes caractéristiques pour les bruits en sortie du système. Pour cela, les termes statistiques des bruits d'entrée $b_i(n)$ sont supposés connus et le modèle à élaborer doit permettre de déterminer ces mêmes termes pour les bruits de sortie $b_{yu}(n)$. Ainsi, si le modèle vérifie cette condition, alors celui-ci permet l'étude de la composition de différents systèmes. De ce fait, le modèle est complet en s'adaptant à tout type de système et chaque bloc contenu dans le système peut être étudié indépendamment des autres.

3.1.2 Modèles de propagation des bruits

Les différentes sources de bruit dont les caractéristiques statistiques ont été définies dans la partie précédente se propagent au sein du système pour aboutir à la présence d'un bruit de sortie $b_{yu}(n)$. Le bruit se propage à travers les différents opérateurs arithmétiques (additionneur, soustracteur, multiplieur et diviseur) composant le système. Pour étudier en détails le bruit de sortie $b_{yu}(n)$, les modèles de propagation des bruits au niveau de chaque type d'opération doivent être définis. Dans cette partie, les différents modèles de propagation des bruits au sein des opérations arithmétiques sont présentés.

Soit γ une opération arithmétique composée de deux entrées $x(n)$ et $y(n)$ et d'une sortie $z(n)$. A chacune des deux entrées est associée un bruit $b_x(n)$ et $b_y(n)$. Soit $b_z(n)$ le bruit associé à la sortie $z(n)$. L'objectif est d'exprimer ce bruit de sortie $b_z(n)$ en fonction des données et des bruits d'entrée selon l'opération γ effectuée. Ce bruit $b_z(n)$ s'exprime en fonction des données d'entrée $x(n)$ et $y(n)$, mais aussi des bruits d'entrée $b_x(n)$ et $b_y(n)$ comme défini au travers de l'expression suivante :

$$b_z(n) = f_\gamma(x,y,b_x,b_y) \tag{3.12}$$

où la fonction f_γ dépend de l'opération γ en question. Cependant, les tailles des données $x(n)$ et $y(n)$ influent sur la modélisation de la propagation des bruits. En effet, si les données sont scalaires ou vectorielles, le traitement n'est pas le même et le résultat du modèle diffère. Ainsi, les cas scalaires et vectoriels/matriciels sont traités séparément.

Opérations scalaires

Dans cette section, un modèle de propagation des bruits au sein d'opérations effectuées sur des scalaires est proposé. Une étude est réalisée sur les différentes opérations arithmétiques puis une généralisation est faite pour toutes ces opérations.

Opération d'addition/soustraction Dans ce paragraphe, l'opération d'addition/soustraction est considérée. La somme de deux données bruitées peut se séparer en deux additions distinctes : l'une sur les données et l'autre sur les bruits comme le montre la figure 3.1. Le bruit de sortie $b_z(n)$ est défini par l'expression suivante :

$$b_z(n) = b_x(n) \pm b_y(n) \tag{3.13}$$

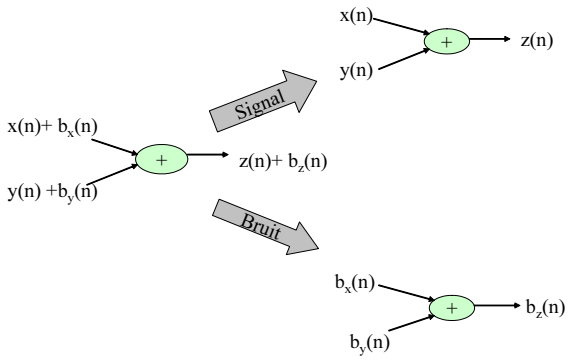

FIG. 3.1 – *Modélisation au niveau signal et bruit de l'opération d'addition*

Le bruit en sortie de l'opération est simplement la somme des deux bruits d'entrée.

Opération de multiplication Dans le cas d'une opération de multiplication, le graphe peut être divisé en deux graphes distincts. La valeur du bruit de sortie est représentée sur la figure 3.2 et exprimée par la relation suivante :

$$b_z(n) = b_x(n)y(n) + x(n)b_y(n) \tag{3.14}$$

Le terme croisé $b_x b_y$ apparaît initialement dans cette expression. Néanmoins, ce terme n'est pas pris en compte car il est négligeable par rapport aux autres termes présents dans l'équation. En effet, les modèles de bruits présentés dans la partie 1.2 sont basés sur une condition. La puissance du bruit doit être inférieure à celle du signal. De ce fait, les termes d'ordre 2 du bruit peuvent être négligés par rapport aux termes d'ordre 1. Le bruit en sortie de l'opération de multiplication est la somme de chaque source de bruit multipliée par l'autre terme de signal.

Opération de division La sortie bruitée d'une opération de division s'exprime sous la forme

$$z(n) + b_z(n) = \frac{x(n) + b_x(n)}{y(n) + b_y(n)} = \frac{x(n) + b_x(n)}{y(n)(1 + \frac{b_y(n)}{y(n)})} \tag{3.15}$$

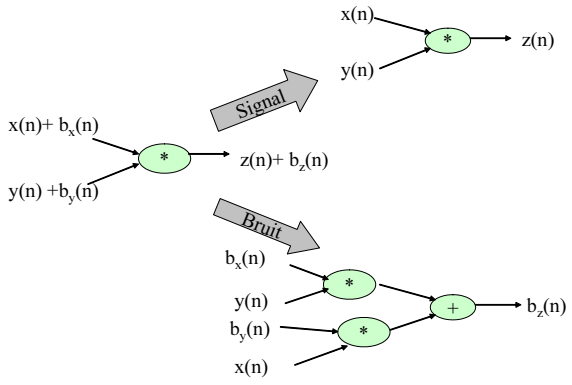

FIG. 3.2 – *Modélisation au niveau signal et bruit de l'opération de multiplication*

Un développement de Taylor à l'ordre 1 permet d'aboutir à l'expression suivante :

$$\frac{1}{1+\frac{b_y(n)}{y(n)}} = 1 - \frac{b_y(n)}{y(n)} \tag{3.16}$$

Ainsi, la sortie bruitée de l'opération s'écrit

$$\begin{aligned} z(n) + b_z(n) = & \quad \frac{(x(n)+b_x(n))(1-\frac{b_y(n)}{y(n)})}{y(n)} \\ = & \quad \frac{x(n)}{y(n)} + \frac{b_x(n)}{y(n)} - \frac{b_y(n)x(n)}{y(n)^2} - \underbrace{\frac{b_y(n)b_x(n)}{y(n)^2}}_{\text{négligé}} \end{aligned} \tag{3.17}$$

De ce fait, le bruit $b_z(n)$ en sortie s'exprime comme la différence de la sortie bruitée et du signal de sortie de l'opération de division (figure 3.3).

$$b_z = \frac{b_x(n)}{y(n)} - \frac{x(n)b_y(n)}{y(n)^2} \tag{3.18}$$

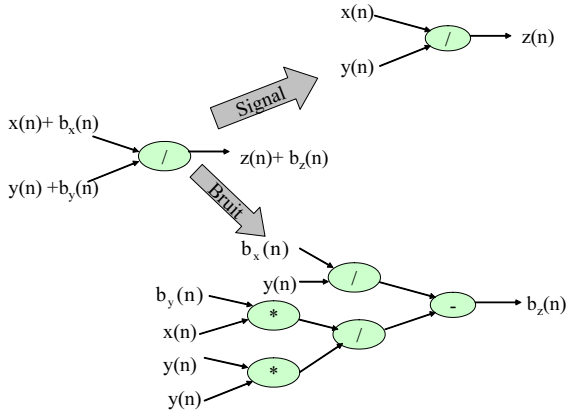

FIG. 3.3 – *Modélisation au niveau signal et bruit de l'opération de division*

Le bruit en sortie s'écrit comme la somme des deux bruits associés aux données d'entrée multipliés par des termes composés des données d'entrée.

Operation	Valeur de $\alpha^{(1)}$	Valeur de $\alpha^{(2)}$
$z = x \pm y$	1	$\pm$ 1
$z = x \times y$	y	x
$z = \frac{x}{y}$	$\frac{1}{y}$	$-\frac{x}{y^2}$

TAB. 3.1 – *Différentes valeurs des termes $\alpha^{(1)}$ et $\alpha^{(2)}$ de l'équation (3.20) pour différentes opérations $\{+, -, \times, \div\}$*

Cas général Les 4 exemples précédents permettent de modéliser le bruit de sortie $b_z(n)$ en fonction des 2 bruits d'entrée $b_x(n)$ et $b_y(n)$ traités de manière séparée car les termes croisés n'apparaissent pas. En effet, les termes de bruit d'ordre 2 sont négligés par rapport aux termes d'ordre 1. Cette hypothèse réaliste se base sur les modèles des bruits de quantification. En effet, si cette condition n'est pas vérifiée, le bruit n'est plus uniformément réparti sur son intervalle de définition. Le bruit de sortie $b_z(n)$ s'écrit comme la somme de deux fonctions appliquées aux deux bruits d'entrée $b_x(n)$ et $b_y(n)$:

$$b_z(n) = f1_\gamma(x,y)b_x(n) + f2_\gamma(x,y)b_y(n) \tag{3.19}$$

Les 2 bruits d'entrée contribuent donc de manière séparée à générer le bruit de sortie $b_z(n)$. Ainsi, pour étudier la propagation des bruits au sein des opérations arithmétiques, la propagation d'un seul des bruits d'entrée peut être étudiée car les termes croisés n'apparaissent pas. Le bruit en sortie est la somme des différentes sources de bruit propagées.

De plus, le bruit en sortie de l'opérateur peut s'écrire comme la somme de chaque bruit d'entrée pondéré par des termes de signaux comme le montre l'expression suivante :

$$b_z = f_\gamma\,(b_x,b_y) = \alpha^{(1)}b_x + \alpha^{(2)}b_y \tag{3.20}$$

Les valeurs de ces termes de signaux $\alpha^{(1)}$ et $\alpha^{(2)}$ dépendent du type d'opération effectuée. Le tableau 3.1 résume les valeurs des termes $\alpha^{(1)}$ et $\alpha^{(2)}$ en fonction du type d'opération et des signaux $x(n)$ et $y(n)$ en entrée de l'opération.

Opérations vectorielles

Dans cette section, les opérations effectuées sur des vecteurs ou des matrices sont étudiées. Comme dans le cas scalaire, une étude des différentes opérations arithmétiques est faite et un modèle de propagation des bruits en est déduit. Soit une opération arithmétique γ composée de deux entrées $X(n)$ et $Y(n)$ vectorielles ou matricielles et d'une sortie $Z(n)$. A chacune des deux entrées est associé un bruit noté $B_X(n)$ et $B_Y(n)$. L'objectif est de déterminer l'expression du bruit de sortie $B_Z(n)$ associé au signal de sortie $Z(n)$.

Opération d'addition vectorielle Dans le cas d'une opération d'addition, le bruit en sortie $B_Z(n)$ est égal à la somme des deux bruits d'entrée $B_X(n)$ et $B_Y(n)$ comme l'exprime l'équation suivante :

$$B_Z(n) = B_X(n) + B_Y(n) \tag{3.21}$$

Opération de multiplication vectorielle Pour une multiplication vectorielle, plusieurs cas sont à considérer. En effet, une multiplication vectorielle peut se faire sous la forme classique $Z(n) = X(n)Y(n)$ mais également en faisant intervenir des termes transposés comme $Z(n) = X^t(n)Y(n)$ ou encore $Z(n) = X(n)Y^t(n)$. Selon chaque cas, le bruit propagé s'exprime de manière différente comme le montrent les relations suivantes :

$$\begin{aligned} Z(n) = X(n)Y(n) \rightarrow B_Z(n) &= B_X(n)Y(n) + X(n)B_Y(n) \\ Z(n) = X^t(n)Y(n) \rightarrow B_Z(n) &= B_X^t(n)Y(n) + X^t(n)B_Y(n) \\ Z(n) = X(n)Y^t(n) \rightarrow B_Z(n) &= B_X(n)Y^t(n) + X(n)B_Y^t(n) \end{aligned} \tag{3.22}$$

Contrairement au cas scalaire, les termes vectoriels ne commutent pas. Dans l'expression précédente, le terme de bruit est multiplié par un terme de signal, placé soit à gauche, soit à droite du bruit. De plus, dans la deuxième expression, le terme de signal est transposé. Le modèle de propagation des bruits scalaires n'est plus valable ici. Soit $B'(n)$ le bruit dû à la propagation du bruit $B_X(n)$ dans l'opération. La propagation du bruit $B_X(n)$ au sein d'une opération peut s'écrire sous la forme :

$$B'(n) = \mathbf{A}_X B_X(n)\mathbf{D}_X + \mathbf{U}_X B_X^t(n)\mathbf{V}_X \tag{3.23}$$

où les termes $\mathbf{A}_X$, $\mathbf{D}_X$, $\mathbf{U}_X$ et $\mathbf{V}_X$ sont des termes de signaux dont les dimensions dépendent de celles des signaux traités. Ce modèle permet d'être général à l'ensemble des cas vus précédemment. Pour chaque cas traité, l'un des deux termes $\mathbf{A}_X B_X(n)\mathbf{D}_X$ ou $\mathbf{U}_X B_X^t(n)\mathbf{V}_X$ est nul. Néanmoins, pour simplifier le modèle, la propagation du bruit $B_X(n)$ est simplement notée

$$B'(n) = \mathbf{A}B_X(n)\mathbf{D} \tag{3.24}$$

Le terme de transposition n'apparaît plus par souci de clarté. Il est néanmoins pris en compte pour le calcul de la puissance du bruit de sortie. La valeur du bruit de sortie $B_Z(n)$ peut s'écrire sous la forme générale suivante

$$B_Z(n) = \mathbf{A_X} B_X(n)\mathbf{D_X} + \mathbf{A_Y} B_Y(n)\mathbf{D_Y} \tag{3.25}$$

Opération d'inversion vectorielle L'opération de division n'existe pas dans le cas vectoriel. Dans le cas matriciel, cela correspond à la multiplication par une matrice inverse. Ainsi, ces opérations sont des multiplications. Le modèle de la multiplication vectorielle s'applique donc dans ce cas.

Conclusion

Les modèles de propagation des bruits scalaires est un cas particulier du modèle de propagation des bruits vectoriels. Dans un souci de généralisation, le modèle de propagation de bruit adopté est repris dans l'expression suivante :

$$b'(n) = \mathbf{A}b(n)\mathbf{D} \tag{3.26}$$

où $b(n)$ est la source de bruit et $b'(n)$ la propagation en sortie de l'opération due à $b(n)$. Cette expression permet de modéliser la propagation d'une source de bruit dans différentes opérations arithmétiques. Les termes $\mathbf{A}$ et $\mathbf{D}$ sont des termes multiplicateurs du bruit d'entrée $b(n)$ dépendant des données d'entrée et du type d'opération effectuée. Ces termes de bruits sont de dimensions quelconques car ce modèle est valable pour les bruits scalaires, vectoriels ou matriciels. Si le bruit $b(n)$ est scalaire, l'expression obtenue pour la propagation des bruits scalaires est retrouvée. Ce modèle de propagation de bruit est utilisé par la suite pour établir l'expression du bruit en sortie d'un système composé d'opérations arithmétiques.

3.2 Modélisation du système

Dans cette partie, la modélisation du système utilisée pour déterminer l'expression du bruit en sortie est présentée. Cette méthode a pour but de développer une expression analytique de la propagation des bruits au sein du système composé d'opérations arithmétiques (addition, soustraction, multiplication, division). Dans un premier temps, la modélisation d'un système est présentée. Dans un second temps, ce modèle est appliqué à la composition de systèmes.

Dans le cadre d'un système en précision finie, des bruits sont associés aux données d'entrée du système. De plus, des bruits sont générés également au niveau des différentes opérations inclues dans le système. La modélisation du système en précision finie possède en entrée les sources de bruit (les bruits associés aux signaux d'entrée et les bruits générés au niveau des opérateurs). La sortie du système au niveau bruit comporte les différents bruits associés aux signaux de sortie du système initial. Le système est composé de N_s bruits de sortie et N_e bruits d'entrée. Le modèle doit permettre de calculer les caractéristiques statistiques de ces N_s bruits de sortie. Chacun de ces bruits de sortie pouvant être étudié indépendamment des autres, un seul bruit de sortie $b_{yu}(n)$ est pris en considération. Pour traiter les autres termes, la méthode définie ci-dessous est reprise. En fonction du système considéré, ce bruit peut être soit scalaire, soit vectoriel ou encore matriciel. L'étude proposée est développée dans le cas général. L'objectif de la méthode est de déterminer une expression analytique liant les bruits de sortie du système aux bruits d'entrée. Or, le bruit de sortie dépend du système parcouru par les bruits d'entrée. Pour obtenir une expression analytique du bruit de sortie, le système doit être modélisé.

Dans la section 3.1.2 en page 78, la propagation des bruits au sein des différentes opérations a été étudiée. Le bruit en sortie peut se modéliser sous la forme d'une somme pondérée des différentes sources de bruit en entrée de l'opération. Le bruit en sortie est donc la somme de tous les bruits d'entrée pondérés par des termes incluant les signaux ou des coefficients (figure 3.4). Chaque terme $b'_{ui}(n)$ désigne la propagation du bruit $b_i(n)$ vers la sortie u du système. Le bruit final $b_{yu}(n)$ est alors la somme de toutes ces contributions.

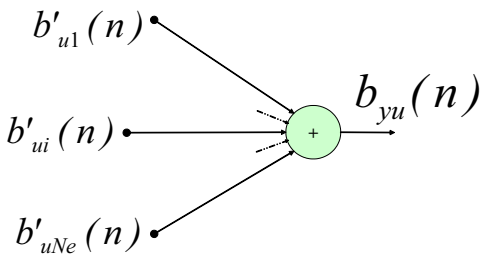

FIG. 3.4 – *Modélisation du bruit de sortie*

$$b_{yu}(n) = \sum_{i=1}^{N_e} b'_{ui}(n) \tag{3.27}$$

Pour simplifier les notations, $b_{yu}(n)$ est simplement noté $y(n)$ car un seul bruit de sortie est pris en considération. Le terme $b'_i(n)$ définit la contribution du bruit $b_i(n)$ en sortie du système. L'expression simplifiée est la suivante :

$$y(n) = \sum_{i=1}^{N_e} b'_i(n) \tag{3.28}$$

Chacune de ces contributions $b'_i(n)$ est générée par un bruit initial $b_i(n)$ passant dans un système S_i (figure 3.5).

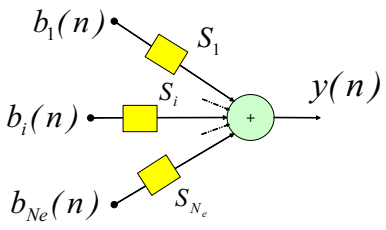

FIG. 3.5 – *Modélisation du bruit dans le système*

Le bruit en sortie est donc fonction des bruits d'entrée et des différents systèmes rencontrés par ces sources de bruit modélisé sous la forme :

$$y(n) = \sum_{i=1}^{Ne} \mathbf{h}_i^{(n)} b_i(n) \tag{3.29}$$

où $\mathbf{h}_i^{(n)}$ représente l'influence du système sur le bruit d'entrée b_i à l'instant n. L'objectif est de modéliser ce terme $\mathbf{h}_i^{(n)}$, et ainsi, de déterminer la contribution du bruit $b_i(n)$. Pour déterminer le bruit de sortie, le chemin emprunté par chacun des bruits d'entrée doit être établi. De ce fait, pour développer le modèle, la propagation des différentes sources de bruit doit être connue pour identifier les circuits empruntés par chacune d'elles.

Le bruit $b_i(n)$ contribue à générer le bruit de sortie $y(n)$ en parcourant le système S_i. Cependant, lors de son parcours du système, celui-ci peut emprunter différents circuits (chacun d'eux étant composé de différents sous-systèmes placés en série ou en parallèle) pour aboutir à la sortie $y(n)$ (figure 3.15). Les termes S_{i_k} désignent les différents sous-systèmes parcourus par le bruit $b_i(n)$. Le système S_i parcouru par $b_i(n)$ correspond donc à la superposition de sous-systèmes placés en série ou en parallèle. Pour obtenir une modélisation général du système, les sous-systèmes sont d'abord étudiés. Ensuite, le modèle développé est propagé à leur mise en série ou en parallèle pour obtenir une modélisation du système global parcouru par $b_i(n)$. Ainsi la contribution du bruit $b_i(n)$ au bruit de sortie $y(n)$ est complètement déterminée.

3.2.1 Modélisation d'un système

Dans cette partie, la modélisation d'un système est effectuée. La propagation d'un bruit d'entrée vers la sortie du système se caractérise par le passage du bruit d'entrée dans différents systèmes placés en série et en parallèle. De ce fait, pour modéliser le système global, les sous-systèmes empruntés par le bruit d'entrée doivent être modélisés. Or, les systèmes peuvent être soit LTI ou non-LTI, soit récursifs ou non récursifs. Un modèle général à l'ensemble de ces types est mis en œuvre. L'idée est de déterminer le système parcouru par le bruit sous forme d'une pseudo fonction de transfert car les termes rencontrés ne sont pas forcément constants. Pour modéliser ces différents systèmes, la contribution d'un seul bruit d'entrée est étudiée car les sources de bruit peuvent être étudiées indépendamment les unes des autres.

Le système considéré est composé du bruit $b(n)$ en entrée et le terme $b'(n)$ désigne le bruit en sortie. Ce bruit en sortie correspond à la propagation du bruit $b(n)$ dans le sous-système considéré. Le bruit de sortie $b'(n)$ est donc fonction du bruit d'entrée $b(n)$ comme le modélise l'expression

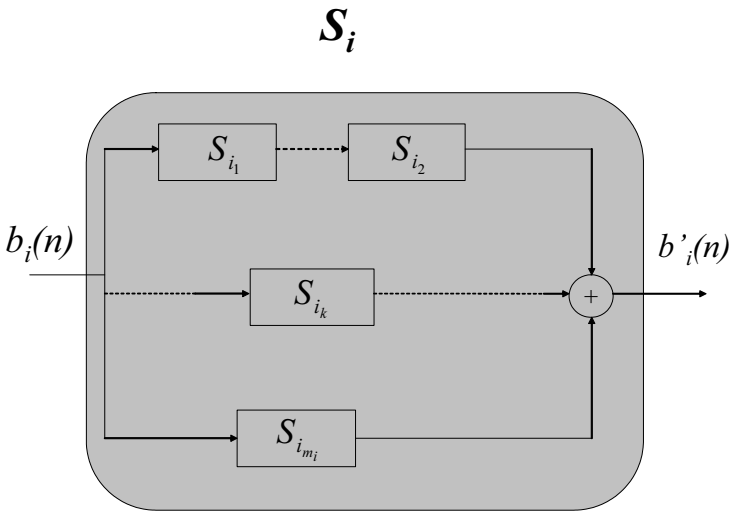

FIG. 3.6 – *Division en sous-systèmes*

$$b'(n) = \mathbf{h}^{(n)} b(n) \tag{3.30}$$

L'objectif est d'exprimer le terme $\mathbf{h}^{(n)}$ en fonction des caractéristiques du système. La détermination de ce terme permet de mesurer l'influence du système sur le bruit considéré. Il permet d'établir la contribution du bruit $b(n)$ à la génération du bruit de sortie. L'entrée du système est le bruit $b(n)$. Cependant le système peut aussi utiliser les échantillons précédents de ce bruit $b(n-m)$ pour $m \in \mathbb{N}$. Soit $g^{(n)}_{(n-m)}$ le terme appliqué à l'échantillon de bruit d'entrée $b(n-m)$ retardé de m échantillons alors que le bruit de sortie $b'(n)$ est à l'instant n. Les termes croisés n'apparaissent pas comme l'indiquent les modèles de propagation étudiés précédemment. Le nombre d'échantillons retardés pris en compte par le système est noté Q. La contribution du bruit $b(n)$ et de ses échantillons précédents est notée $b_1(n)$ et exprimée par la relation suivante :

$$b_1(n) = \sum_{m=0}^{Q} g^{(n)}_{(n-m)} b(n-m) \tag{3.31}$$

Le bruit $b_1(n)$ s'exprime comme la somme des contributions de toutes les versions retardées du bruit $b(n)$. Le terme $g^{(n)}_{(n-m)}$ correspond à la modélisation de la contribution du bruit $b(n-m)$ à la génération du bruit de sortie. Il est défini par des termes scalaires, matriciels ou vectoriels. Sa valeur est déterminée par la suite. La contribution du bruit $b(n-m)$ se définit comme la modélisation de la propagation du bruit au sein du système pour les échantillons retardés de même période d'échantillonnage.

Le bruit de sortie du système $b'(n)$ est donc fonction du bruit d'entrée $b(n)$ et de ces échantillons précédents. Cependant, le bruit de sortie peut aussi dépendre des échantillons de sortie précédents $b'(n-p)$ pour $p \in \mathbb{N}^*$. Soit $b_2(n)$ la contribution des échantillons précédents de $b'(n)$ à la génération du bruit de sortie $b'(n)$. Soit P le nombre d'échantillons précédents de $b'(n)$ pris en compte. Le terme $b_2(n)$ est représenté sur la figure 3.8 et peut s'écrire sous la forme

$$b_2(n) = \sum_{i=1}^{P} f^{(n)}_{(n-m)} b'(n-i) \tag{3.32}$$

où $f^{(n)}_{(n-k)}$ désigne le terme associé à l'échantillon de sortie retardé de k alors que la sortie est à l'instant n. Celui-ci représente des termes scalaires, vectoriels ou matriciels.

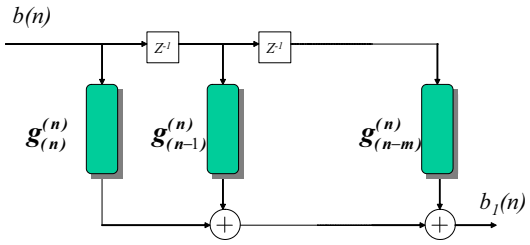

FIG. 3.7 – *Modélisation du bruit généré par les échantillons du bruit d'entrée* $b(n)$

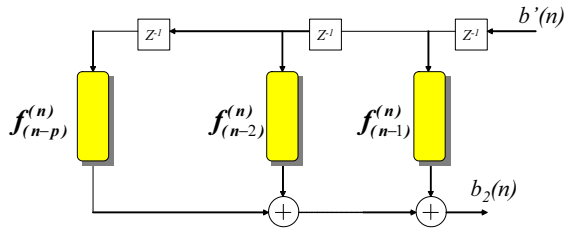

FIG. 3.8 – *Modélisation du bruit généré par les échantillons du bruit de sortie* $b'(n)$

Le bruit de sortie $b'(n)$ est la somme des contributions du bruit d'entrée $b_1(n)$ et des contributions du bruit de sortie précédentes $b_2(n)$.

$$\begin{aligned} b'(n) &= b_1(n) + b_2(n) \\ &= \sum_{i=1}^{P} f_{(n-i)}^{(n)} b'(n-i) + \sum_{l=0}^{Q} g_{(n-l)}^{(n)} b(n-l) \end{aligned} \tag{3.33}$$

Cette expression permet d'obtenir une modélisation du système parcouru par le bruit d'entrée. Par la suite, l'objectif est de définir les différentes fonctions g et f modélisant le système.

Pseudo fonction de transfert

Un système à coefficients constants peut se modéliser sous la forme d'une fonction de transfert. Dans le cas général, le système considéré n'a pas nécessairement de coefficients fixes. Ceux-ci peuvent varier au cours du temps. La notion de fonction de transfert n'est plus valide dans ce cas. Cependant, la notion de pseudo fonction de transfert peut être introduite. Au sein de cette pseudo fonction, les coefficients peuvent varier dans le temps. L'équation (3.33) peut être réécrite en utilisant la pseudo fonction de transfert $H^{(n)}(z)$ associée au système parcouru par le bruit $b(n)$. Celle-ci est exprimée dans l'équation suivante :

$$H^{(n)}(z) = \frac{\sum_{l=0}^{Q} g_{(n-l)}^{(n)} z^{-l}}{1 - \sum_{i=1}^{P} f_{(n-i)}^{(n)} z^{-i}} \tag{3.34}$$

Cette expression permet d'exprimer une relation analytique modélisant le système considéré. A l'intérieur de cette expression, les termes f et g désignent soit des scalaires, soit des vecteurs, ou encore des matrices. De ce fait, cette notation permet par la suite, de modéliser le système. Cette expression permet alors de relier le bruit de sortie $b'(n)$ au bruit d'entrée $b(n)$. Dans le cas où les coefficients sont des termes constants, la fonction de transfert classique est retrouvée comme dans le cas des filtres FIR et IIR.

Pseudo réponse impulsionnelle

La notion de réponse impulsionnelle est intéressante car elle permet d'exprimer la sortie uniquement en fonction des entrées même pour les systèmes récursifs. Dans le cadre de notre approche, la notion de pseudo réponse impulsionnelle est introduite. Dans ce paragraphe, l'expression de la pseudo réponse impulsionnelle est déterminée. L'idée est de repartir de l'équation récurrente du système (3.33) et de la développer. Cette étape permet d'obtenir l'expression du bruit de sortie $b'(n)$ en fonction du bruit d'entrée et de ses échantillons précédents $b(n-k)$. Cela aboutit à l'équation suivante :

$$\begin{aligned} b'(n) &= \sum_{i=1}^{P} f_{(n-i)}^{(n)} b'(n-i) + \sum_{l=0}^{Q} g_{(n-l)}^{(n)} b(n-l) \\ &= g_{(n)}^{(n)} b(n) + \left(f_{(n-1)}^{(n)} g_{(n-1)}^{(n-1)} + g_{(n-1)}^{(n)} \right) b(n-1) \\ &+ \left(f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} g_{(n-2)}^{(n-2)} + f_{(n-2)}^{(n)} g_{(n-2)}^{(n-2)} \right. \\ &+ \left. f_{(n-1)}^{(n)} g_{(n-2)}^{(n-1)} + g_{(n-2)}^{(n)} \right) b(n-2) .... \end{aligned} \tag{3.35}$$

Le début du développement de cette dernière équation conduit à la modélisation du bruit $b'(n)$ sous la forme :

$$b'(n) = \sum_{k=0}^{n} h_{(n-k)}^{(n)} b(k) \tag{3.36}$$

où $h^{(n)}$ désigne la pseudo réponse impulsionnelle du système pour le bruit de sortie à l'instant n. Celle-ci s'exprime sous la forme de scalaires, vecteurs ou matrices, appliqués au bruit b. La difficulté réside dans la détermination des différents $h_{(n-k)}^{(n)}$ en fonction de termes g et f modélisant le système. Les premiers termes sont obtenus assez facilement et sont égaux à :

$$\begin{aligned} h_{(n)}^{(n)} &= g_{(n)}^{(n)} \\ h_{(n-1)}^{(n)} &= f_{(n-1)}^{(n)} g_{(n-1)}^{(n-1)} + g_{(n-1)}^{(n)} \\ h_{(n-2)}^{(n)} &= f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} g_{(n-2)}^{(n-2)} + f_{(n-2)}^{(n)} g_{(n-2)}^{(n-2)} \\ &+ f_{(n-1)}^{(n)} g_{(n-2)}^{(n-1)} + g_{(n-2)}^{(n)} \end{aligned} \tag{3.37}$$

Les termes suivants peuvent être calculés en déroulant plus en profondeur la récurrence représentée par l'équation (3.33). Néanmoins, une autre méthode plus rapide consiste à trouver les termes suivants en fonction des termes précédents. En fait, l'objectif est de pouvoir déterminer

$h_{(i)}^{(n)}$ en fonction de $h_{(i+1)}^{(n)}$,$h_{(i+2)}^{(n)}$..... En étudiant plus en détails l'expression (3.37), une récurrence pour passer d'un terme au suivant est présente. En effet pour calculer $h_{(n-2)}^{(n)}$ à partir de $h_{(n)}^{(n)}$ et $h_{(n-1)}^{(n)}$, le lien suivant existe. Les termes de $h_{(n-1)}^{(n)}$ sont décrémentés ($(n-1)$ devient $(n-2)$) puis multipliés par $f_{(n-1)}^{(n)}$ d'une part, et d'autre part, les termes de $h_{(n)}^{(n)}$ sont décrémentés de 2 puis multipliés par $f_{(n-2)}^{(n)}$. Enfin, le terme $g_{(n-2)}^{(n)}$ est ajouté. De manière plus générale, pour calculer $h_{(i)}^{(n)}$, la procédure suivante doit être suivie.

- $h_{(i)}^{(n)}$ dépend des $h_{(i+j)}^{(n)}$ précédents termes où $j \in [1:P]$ et P est l'ordre du dénominateur de la pseudo fonction de transfert.
- $h_{(n)}^{(n)}$ est initialisé à $g_{(n)}^{(n)}$.
- Chaque $h_{(i+j)}^{(n)}$ est décrémenté de j puis composé par $f_{(n-j)}^{(n)}$.
- $h_{(i)}^{(n)}$ est défini comme la somme de chacun des termes précédents et de $g_{(i)}^{(n)}$.

Cette méthode est appliquée pour trouver $h_{(n-3)}^{(n)}$. Soit $h_{(n-2)}^{(n)}$ défini par :

$$h_{(n-2)}^{(n)} = f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} g_{(n-2)}^{(n-2)} + f_{(n-2)}^{(n)} g_{(n-2)}^{(n-2)} + f_{(n-1)}^{(n)} g_{(n-2)}^{(n-1)} + g_{(n-2)}^{(n)} \tag{3.38}$$

Le terme est décrémenté de 1 et composé par par $f_{(n-1)}^{(n)}$. Ceci conduit à l'expression suivante :

$$\begin{aligned} &f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} f_{(n-3)}^{(n-2)} g_{(n-3)}^{(n-3)} + f_{(n-1)}^{(n)} f_{(n-3)}^{(n-1)} g_{(n-3)}^{(n-3)} \\ &+ f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} g_{(n-3)}^{(n-2)} + f_{(n-1)}^{(n)} g_{(n-3)}^{(n-1)} \end{aligned} \tag{3.39}$$

D'autre part, l'expression de $h_{(n-1)}^{(n)}$ est décrémentée de 2, puis multipliée par $f_{(n-2)}^{(n)}$. Cela aboutit à l'équation :

$$f_{(n-2)}^{(n)} f_{(n-3)}^{(n-2)} g_{(n-3)}^{(n-3)} + f_{(n-2)}^{(n)} g_{(n-3)}^{(n-2)} \tag{3.40}$$

Ensuite, $h_{(n)}^{(n)}$ est décrémenté de 3 puis multiplié par $f_{(n-3)}^{(n)}$ conduisant à:

$$f_{(n-3)}^{(n)} g_{(n-3)}^{(n-3)} \tag{3.41}$$

Finalement, $h_{(n-3)}^{(n)}$ est la somme des trois equations (3.39), (3.40) et (3.41) et du terme $g_{(n-3)}^{(n)}$.

$$\begin{aligned} h_{(n-3)}^{(n)} &= f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} f_{(n-3)}^{(n-2)} g_{(n-3)}^{(n-3)} + f_{(n-1)}^{(n)} f_{(n-3)}^{(n-1)} g_{(n-3)}^{(n-3)} \\ &+ f_{(n-1)}^{(n)} f_{(n-2)}^{(n-1)} g_{(n-3)}^{(n-2)} + f_{(n-1)}^{(n)} g_{(n-3)}^{(n-1)} \\ &+ f_{(n-2)}^{(n)} f_{(n-3)}^{(n-2)} g_{(n-3)}^{(n-3)} + f_{(n-2)}^{(n)} g_{(n-3)}^{(n-2)} \\ &+ f_{(n-3)}^{(n)} g_{(n-3)}^{(n-3)} + g_{(n-3)}^{(n)} \end{aligned} \tag{3.42}$$

De ce fait, l'equation de récurrence permettant de trouver $h_{(i)}^{(n)}$ peut être déterminée en fonction des termes précédents.

$$h_{(i)}^{(n)} = \sum_{j=1}^{P} f_{(n-j)}^{(n)} h_{(i)}^{(n-j)} + g_{(i)}^{(n)} \tag{3.43}$$

Systèmes	Système non-récursif	Système récursif
Système LTI	FIR,FFT	IIR
Système non-LTI	Normalisation	LMS

TAB. 3.2 – *Exemples traités en fonction du type de système correspondant*

Ainsi, cette méthode permet de déterminer les termes de la pseudo réponse impulsionnelle du système par récurrence. Le bruit en sortie $b'(n)$ peut s'écrire sous la forme

$$b'(n) = \sum_{k=0}^{n} h_{(k)}^{(n)} b(k) \tag{3.44}$$

Cette expression permet de modéliser le bruit de sortie en fonction uniquement du bruit d'entrée.

Les éléments $h_{(k)}^{(n)}$ correspondent aux termes de pseudo réponse impulsionnelle du système. Ce dernier est composé d'opérations arithmétiques. Dans la partie 3.1.2, un modèle de propagation du bruit à travers les opérations arithmétiques est présenté. Celui-ci démontre la modélisation de la propagation d'un bruit dans un système composé d'opérations arithmétiques à l'aide d'un produit de deux termes $\mathbf{A}_{(k)}^{(n)}$ et $\mathbf{D}_{(k)}^{(n)}$. La fonction $h_{(k)}^{(n)}$ appliquée à l'échantillon d'entrée retardé de k, correspond à la multiplication de celui-ci par deux termes $\mathbf{A}_{(k)}^{(n)}$ et $\mathbf{D}_{(k)}^{(n)}$ définis par les opérations arithmétiques rencontrées. Le bruit en sortie $b'(n)$ du système s'écrit sous la forme :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}_{(k)}^{(n)} b(k) \mathbf{D}_{(k)}^{(n)} \tag{3.45}$$

Les termes $\mathbf{A}_{(k)}^{(n)}$ et $\mathbf{D}_{(k)}^{(n)}$ sont scalaires, vectoriels ou matriciels selon les dimensions des bruits $b(n)$ et $b'(n)$. De manière générale ces termes sont souvent matriciels car ils permettent de factoriser des termes et de simplifier le modèle. Ainsi ces termes sont considérés comme des matrices. Cette dernière expression permet d'écrire de manière exacte la valeur du bruit en sortie du sous-système. De plus, si les termes transposés sont pris en compte, l'expression s'écrit alors sous la forme suivante :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}_{(k)}^{(n)} b(k) \mathbf{D}_{(k)}^{(n)} + \mathbf{U}_{(k)}^{(n)} b^t(k) \mathbf{V}_{(k)}^{(n)} \tag{3.46}$$

où les termes $\mathbf{U}_{(k)}^{(n)}$ et $\mathbf{V}_{(k)}^{(n)}$ sont des compositions de signaux multipliant les échantillons transposés du bruit d'entrée.

Exemples illustrant le modèle

Pour illustrer le modèle, plusieurs exemples sont présentés. Des systèmes LTI, non-LTI, récursifs et non-récursifs sont détaillés. Les applications traitées sont présentées dans le tableau 3.2.

Système LTI non-récursifs : Le filtre FIR Pour illustrer l'approche avec un exemple simple, le cas du filtre à réponse impulsionnelle finie (FIR) est traité. Soient N la taille du filtre, $X(n) = [x(n)x(n-1)...x(n-N+1)]^t$ le vecteur colonne des N derniers échantillons d'entrée et H le vecteur colonne des N coefficients. De plus, $y(n)$ est la sortie du FIR à l'instant n. La relation suivante est établie :

$$y(n) = H^t X(n) \tag{3.47}$$

Soit un vecteur bruit $B(n)$ associé au vecteur $X(n)$. Dans ce cas, le bruit scalaire $b'(n)$ généré en sortie par la propagation de $B(n)$ est défini par l'expression suivante :

$$b'(n) = H^t B(n) \tag{3.48}$$

La formule du modèle précédent est retrouvée avec

$$\begin{aligned} \mathbf{A}_{(n)}^{(n)} &= H^t \\ \mathbf{A}_{(k)}^{(n)} &= 0 \ \forall k < N \\ \mathbf{D}_{(n)}^{(n)} &= 1 \\ \mathbf{D}_{(k)}^{(n)} &= 0 \ \forall k < N \end{aligned} \tag{3.49}$$

Les termes **U** et **V** sont nuls car le bruit d'entrée n'apparaît pas sous sa forme transposée. Ici, $b'(n)$ est un scalaire, et $B(n)$ un vecteur de taille N. Donc $\mathbf{A}_{(n)}^{(n)} = H^t$ est un vecteur ligne de taille N et $\mathbf{D}_{(n)}^{(n)} = 1$ est un scalaire. Le modèle s'écrit alors

$$b'(n) = \mathbf{A}_{(n)}^{(n)} B(n) \tag{3.50}$$

Cet exemple permet d'illustrer le modèle de multiplication du bruit par des termes **A** et **D**.

Exemple d'un système non-LTI non-récursif : La Normalisation Pour rendre un peu plus concrète la modélisation présentée, un exemple de système non-LTI est développé. Pour cela, la normalisation vectorielle est choisie. Soit $X(n) = [x(n)x(n-1)...x(n-N+1)]^t$ un vecteur de taille N. Dans ce cas, $M_i = N$ et $N_i = 1$ et sa normalisation est représentée à la figure 3.9.

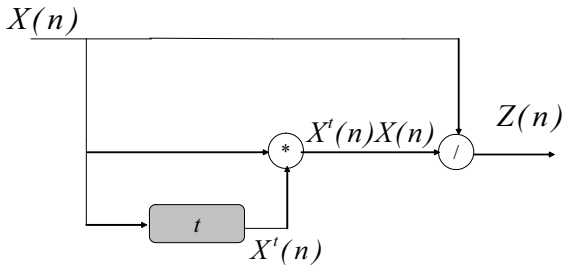

FIG. 3.9 – *Schéma de la Normalisation*

$$Z(n) = \frac{X(n)}{X^t(n)X(n)} \tag{3.51}$$

Au signal $X(n)$ est associé un bruit $B_X(n)$. L'objectif est déterminer le bruit $B_Z(n)$ (de dimension N) associé à la sortie $Z(n)$ dû à la propagation de $B_X(n)$. Ici, le bruit se situe à la fois au numérateur et au dénominateur. La valeur en sortie de l'opérateur est la suivante :

$$Z(n) + B_Z(n) = \frac{X(n) + B_X(n)}{(X^t(n) + B_X^t(n))(X(n) + B_X(n))} \tag{3.52}$$

Pour mieux étudier la propagation du bruit, la figure 3.10 illustre la modélisation de l'opération de normalisation en précision finie. En entrée du graphe, les bruits $B_X(n)$ et sa version transposée $B_X^t(n)$ sont présents. Le bruit $B_X(n)$ parcourt deux chemins différents comme le montre la figure 3.11. Le premier chemin parcouru par $B_X(n)$ aboutit à un bruit $B_{X_1}(n)$. Ce dernier est obtenu en utilisant le modèle de propagation de bruit présenté en section 3.1.2 et est égal à :

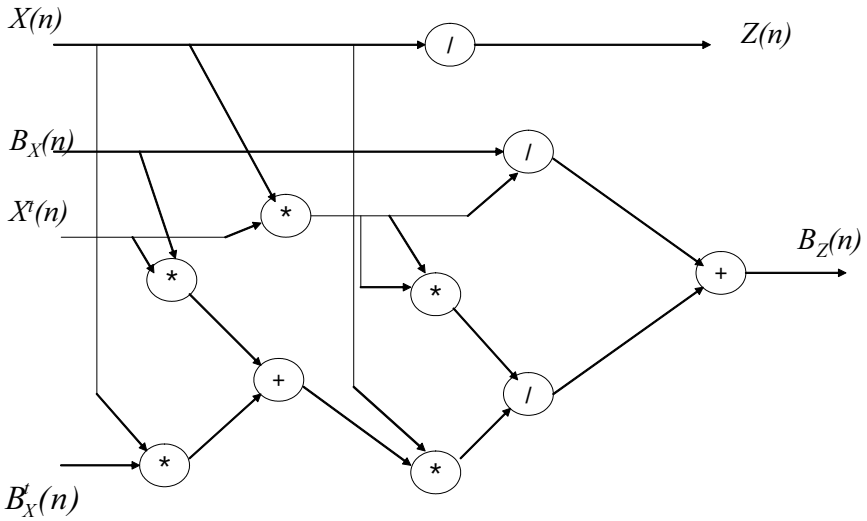

FIG. 3.10 – *Schéma de la Normalisation au niveau graphe*

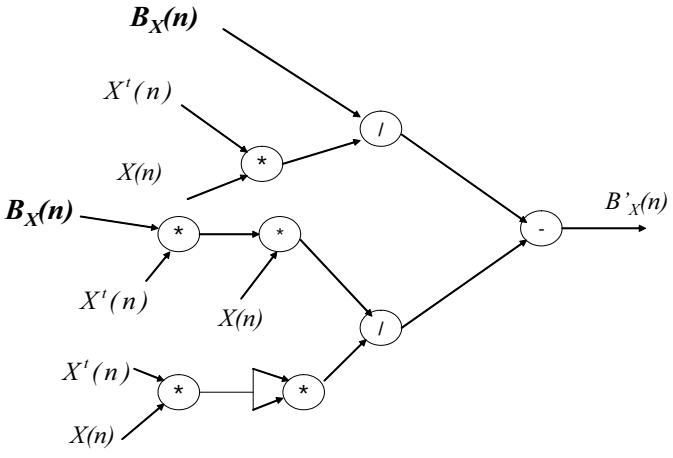

FIG. 3.11 – *Graphe du parcours du bruit* $B_X(n)$

$$B_{X_1}(n) = \frac{B_X(n)}{X^t(n)X(n)} \tag{3.53}$$

ou encore

$$B_{X_1}(n) = \mathbf{A}_1(n)B_X(n)\mathbf{D}_1(n) \tag{3.54}$$

en utilisant le modèle défini en section 3.1.2 avec $\mathbf{A}_1(n) = \frac{1}{X^t(n)X(n)}$ et $\mathbf{D}_1(n) = 1$ étant 2 scalaires dans ce cas. Le second chemin parcouru par le bruit $B_X(n)$ aboutit à un bruit $B_{X_2}(n)$ défini par la relation suivante :

$$B_{X_2}(n) = -\frac{X(n)X^t(n)B_X(n)}{(X^t(n)X(n))^2} \tag{3.55}$$

ou encore

$$B_{X_2}(n) = \mathbf{A}_2(n)B_X(n)\mathbf{D}_2(n) \tag{3.56}$$

avec $\mathbf{A}_2(n) = -\frac{X(n)X^t(n)}{(X^t(n)X(n))^2}$ et $\mathbf{D}_2(n) = 1$. Le terme $\mathbf{A}_2(n)$ est une matrice de taille NxN alors que $\mathbf{D}_2(n)$ est un scalaire. La contribution globale $B'_X(n)$ du bruit $B_X(n)$ est égale à la somme des deux contributions en sortie de chacun des deux chemins parcourus.

$$B'_X(n) = \frac{B_X(n)}{X^t(n)X(n)} - \frac{X(n)X^t(n)B_X(n)}{(X^t(n)X(n))^2} \tag{3.57}$$

Le modélisation de la propagation du bruit $B_X(n)$ est repris sur la figure 3.12. Les symboles $.*$ et $*.$ désignent, respectivement, une multiplication à gauche et à droite.

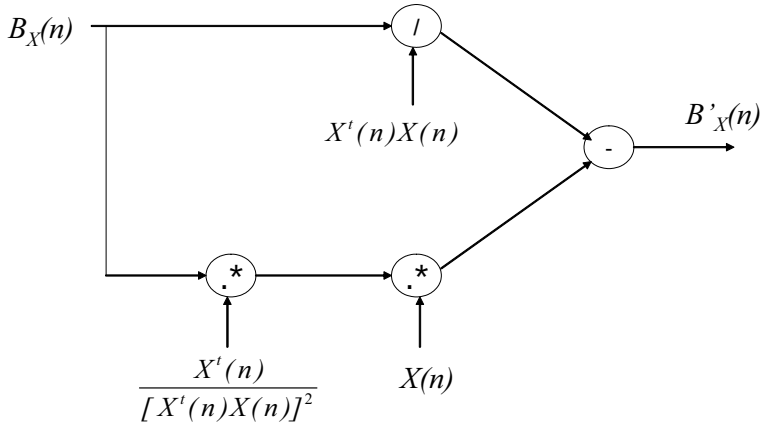

FIG. 3.12 – *Modélisation de la propagation du bruit* $B_X(n)$

Le bruit $B^t_X(n)$ parcourt un seul chemin représenté sur la figure 3.13, pour aboutir à la sortie du système. Sa contribution notée $B'_{X^t}(n)$ est donnée par la relation suivante :

$$B'_{X^t}(n) = -\frac{X(n)B^t_X(n)X(n)}{(X^t(n)X(n))^2} \tag{3.58}$$

ou encore

$$B'_{X^t}(n) = \mathbf{U}_3(n)B^t_X(n)\mathbf{V}_3(n) \tag{3.59}$$

avec $\mathbf{U}_3(n) = -\frac{X(n)}{(X^t(n)X(n))^2}$ et $\mathbf{V}_3(n) = X(n)$, étant 2 vecteurs colonne de taille N. Ces termes $\mathbf{U}$ et $\mathbf{V}$ sont introduits car le bruit d'entrée $B_X(n)$ est transposé. Les dimensions de ces termes étant différents de ceux de $\mathbf{A}$ et $\mathbf{D}$ définis précédemment, ces nouvelles notations sont utilisées dans ce cas. La propagation du bruit B^t_X est présentée sur la figure 3.14.

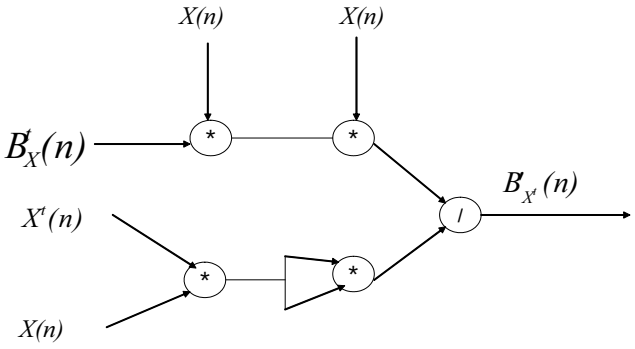

FIG. 3.13 – *Graphe de la propagation du bruit* $B^t_X(n)$

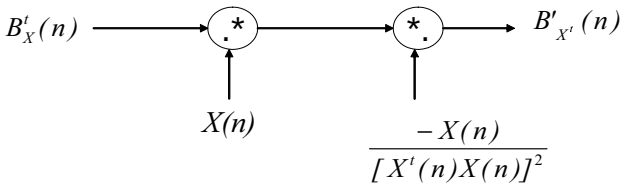

FIG. 3.14 – *Modélisation de la propagation du bruit* $B^t_X(n)$

Finalement, $B_Z(n)$ est la somme des contributions des bruits $B_X(n)$ et de $B^t_X(n)$ comme l'exprime la relation suivante :

$$B_Z(n) = \frac{B_X(n)}{X^t(n)X(n)} - \frac{X(n)B^t_X(n)X(n)}{(X^t(n)X(n))^2} - \frac{X(n)X^t(n)B_X(n)}{(X^t(n)X(n))^2} \tag{3.60}$$

Cet exemple permet de modéliser la propagation d'un bruit empruntant plusieurs chemins différents. Le bruit de sortie est alors la somme des contributions du bruit d'entrée.

Exemple d'un système LTI récursif : le Filtre IIR Pour illustrer la méthode de détermination de la pseudo réponse impulsionnelle, le modèle est appliquée à un filtre à réponse impulsionnelle infinie (IIR). Pour cela, le filtre IIR choisi est d'ordre 2 et défini par la fonction de transfert suivante :

$$H_1(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3.61}$$

Au signal d'entrée $x(n)$ (scalaire) est associé un bruit $b(n)$ également scalaire. Ce dernier génère un bruit $b'(n)$ en sortie du filtre IIR. Pour simplifier la présentation, aucun bruit n'est généré par les multiplications. De ce fait, le système se résume à un bruit $b(n)$ en entrée et un bruit $b'(n)$ en sortie. En temporel, l'équation de $b'(n)$ en fonction de $b(n)$ devient :

$$b'(n) = a_1 b'(n-1) + a_2 b'(n-2) + b(n) \tag{3.62}$$

Par analogie, avec notre modèle présenté précédemment, les termes f et g sont des scalaires définis de la manière suivante :

$$\begin{aligned} g_{(n)}^{(n)} &= 1 \ \forall n \\ f_{(n-1)}^{(n)} &= a_1 \ \forall n \\ f_{(n-2)}^{(n)} &= a_2 \ \forall n \\ f_{(n-i)}^{(n)} &= 0 \ \forall i > 2 \end{aligned} \tag{3.63}$$

En appliquant l'equation (3.37) aux données précédentes, les 3 premiers termes de réponse impulsionnelle ont pour valeur :

$$\begin{aligned} h_{(n)}^{(n)} &= 1 \\ h_{(n-1)}^{(n)} &= a_1 \\ h_{(n-2)}^{(n)} &= a_1 a_1 + a_2 \end{aligned} \tag{3.64}$$

Avec la méthodologie présentée pour déduire les termes de réponse impulsionnelle, les termes suivants sont égaux à :

$$\begin{aligned} h_{(n-3)}^{(n)} &= a_1 a_1 a_1 + a_2 a_1 + a_1 a_2 + \underbrace{a_3}_{0} \\ h_{(n-4)}^{(n)} &= a_1 a_1 a_1 a_1 + a_2 a_1 a_1 + a_1 a_2 a_1 + a_1 a_1 a_2 + a_2 a_2 + \underbrace{a_1 a_3}_{=0} \end{aligned} \tag{3.65}$$

Pour résumer, les premières valeurs de h sont les suivantes :

$$\begin{aligned} h_{(n)}^{(n)} &= 1 \\ h_{(n-1)}^{(n)} &= a_1 \\ h_{(n-2)}^{(n)} &= a_1^2 + a_2 \\ h_{(n-3)}^{(n)} &= a_1^3 + 2a_2 a_1 \\ h_{(n-4)}^{(n)} &= a_1^4 + 3a_1^2 a_2 + a_2^2 \end{aligned} \tag{3.66}$$

Pour montrer la validité de notre approche, l'objectif est de retrouver ces résultats en utilisant la fonction de transfert du système et en appliquant les techniques classiques de traitement du signal pour les systèmes LTI. La fonction de transfert du filtre est égale à :

$$H_1(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3.67}$$

En prenant la Transformée en Z inverse de $H_1(z)$, les valeurs de $h^{(n)}$ devraient être retrouvées. Si $h_1(n) = \mathcal{Z}^{-1}(H_1(z))$, alors l'égalité $h_1(n) = h$ doit être vérifiée. Les calculs sont présentés en annexe B. Le bruit de sortie $b'(n)$ s'écrit de la manière suivante :

$$b'(n) = \sum_{m=0}^{\infty} c_m b(n-m) \tag{3.68}$$

avec

$$c_m = \frac{1}{2^m} \sum_{p=0}^{\lfloor \frac{m}{2} \rfloor} C_{m+1}^{2p+1} a_1^{m-2p} (a_1^2 + 4a_2)^p \tag{3.69}$$

Ces équations permettent d'aboutir aux égalités suivantes

$$h^{(n)}_{(n-i)} = c_i \tag{3.70}$$

De ce fait, notre modèle est correct car les résultats sont identiques à ceux obtenus à l'aide d'une étude propre aux systèmes LTI. Ce modèle s'adapte donc bien aux systèmes récursifs LTI et permet de retrouver la réponse impulsionnelle de ce système.

Conclusion Les exemples présentés permettent d'illustrer le modèle développé. Celui-ci s'adapte parfaitement à chacun des cas rencontrés. Le filtre FIR illustre la méthode sur un exemple simple. La normalisation montre les différents chemins parcourus par le bruit pour aboutir à la sortie. L'algorithme IIR valide la méthode de calcul de la pseudo réponse impulsionnelle.

Modèle général

La modélisation de la propagation d'un bruit au cœur d'un système a été présentée. La propagation au sein du système peut se représenter sous la forme de la pseudo fonction de transfert définie par la relation suivante :

$$H^{(n)}(z) = \frac{\sum_{l=0}^{Q} g^{(n)}_{(n-l)} z^{-l}}{1 - \sum_{i=1}^{P} f^{(n)}_{(n-i)} z^{-i}} \tag{3.71}$$

Cette expression peut être transformée avec les termes $h^{(n)}$ de la pseudo réponse impulsionnelle :

$$H^{(n)}(z) = \sum_{k=0}^{n} h^{(n)}_{(n-k)} z^{-k} \tag{3.72}$$

De manière générale, chaque terme de pseudo réponse impulsionnelle correspond à la multiplication par deux termes de signaux $\mathbf{A}$ et $\mathbf{D}$ comme l'exprime la relation suivante :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}^{(n)}_{(k)} b(k) \mathbf{D}^{(n)}_{(k)} \tag{3.73}$$

Le bruit d'entrée $b(n)$ est de dimension quelconque (scalaire, vectoriel, matriciel) tout comme le bruit de sortie $b'(n)$. L'équation (3.73) permet d'obtenir le modèle général de la propagation d'une source de bruit en sortie d'un système quelconque. En présence de plusieurs sources de bruit, le bruit de sortie $y(n)$ est alors la somme des contributions de chacune de ces sources et donne l'équation suivante :

$$y(n) = \sum_{i=1}^{N_e} \sum_{k=0}^{n} \mathbf{A}^{(n)}_{i(k)} b_i(k) \mathbf{D}^{(n)}_{i(k)} \tag{3.74}$$

De plus, si le système considéré possède plusieurs sorties, alors l'écriture doit encore être modifiée. La $u^{\text{ème}}$ sortie notée $b_{yu}(n)$ a pour valeur :

$$b_{yu}(n) = \sum_{i=1}^{N_e} \sum_{k=0}^{n} \mathbf{A}^{(n)}_{ui(k)} b_i(k) \mathbf{D}^{(n)}_{ui(k)} \tag{3.75}$$

Cependant, ce modèle peut encore être complété. En effet, lors de la propagation de la source de bruit dans le système, cette dernière peut passer dans un opérateur de transposition. Ainsi, un nouveau terme est ajouté concernant les termes transposés en sortie du système. L'expression du bruit de sortie devient :

$$b_{yu}(n) = \sum_{i=1}^{N_e} \sum_{k=0}^{n} \mathbf{A}^{(n)}_{ui(k)} b_i(k) \mathbf{D}^{(n)}_{ui(k)} + \mathbf{U}^{(n)}_{ui(k)} b^t_i(k) \mathbf{V}^{(n)}_{ui(k)} \tag{3.76}$$

où les termes **U** et **V** sont des combinaisons de signaux associées à la propagation de la source de bruit transposée. Si la source de bruit est un scalaire, alors ce nouveau terme n'existe pas car la transposition agit uniquement sur les vecteurs et les matrices. Dans la majorité des cas ce terme n'intervient pas. Ainsi, pour simplifier l'écriture, ce terme n'apparaît pas dans nos équations sauf pour le calcul des statistiques du bruit de sortie détaillé par la suite.

L'expression comporte alors de nombreux indices et devient assez complexe. Ainsi, dès que possible, l'expression est simplifiée. Si une seule sortie est considérée, celle-ci est notée $y(n)$ et l'expression (3.74) est retrouvée. Si de plus, un seul bruit d'entrée est pris en compte, l'équation (3.73) est reprise. D'autre part, sauf cas contraire, l'instant de sortie est toujours n. De ce fait, les termes matriciels sont simplement notés $\mathbf{A}(k)$ si aucune confusion n'est possible.

Les systèmes traités peuvent être relativement complexes et composés de différents blocs. Dans cette partie, un modèle définissant la propagation d'une source de bruit dans un système de base à été défini (equation (3.73)). Dans la partie suivante, la composition de systèmes afin de déterminer la propagation d'un bruit au travers de différents systèmes cascadés. Cette approche permet d'obtenir un modèle général pour tout type de système et toute composition de systèmes entre eux.

3.2.2 Composition de systèmes

Dans la partie précédente, la modélisation d'un système a été présentée. Dans cette section, le modèle est généralisé à la composition des systèmes entre eux. La propagation d'une seule source de bruit peut se modéliser sous la forme suivante :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}(k) b(k) \mathbf{D}(k) \tag{3.77}$$

Or lors de l'étude de la précision d'une application, celle-ci peut être divisée en plusieurs blocs indépendants. Le modèle doit être capable de mettre en œuvre la composition de systèmes entre eux. Le bruit d'entrée parcourt alors plusieurs systèmes différents comme le montre la figure 3.15. Ces systèmes traversés par le bruit d'entrée $b(n)$ peuvent être placés soit en parallèle, soit en série. De plus, cette approche permet d'étudier les différents systèmes indépendamment les uns des autres. Chaque cas est traité par la suite.

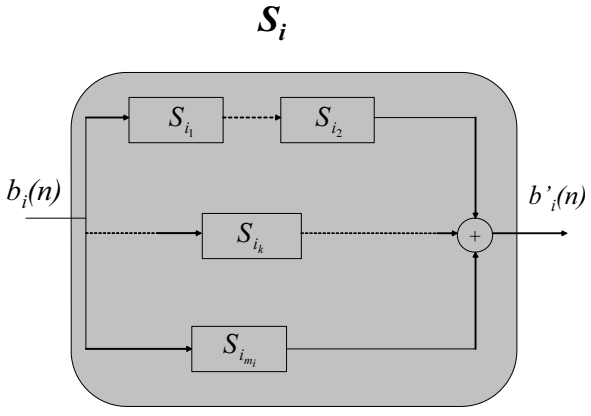

FIG. 3.15 – *Division en sous-systèmes*

Composition de sous-systèmes en parallèle

Dans cette partie, l'étude porte sur le cas où les deux systèmes sont placés en parallèle (figure 3.16). Pour simplifier les notations, les termes $\mathbf{A}_{(k)}^{(n)}$ sont simplement notés $\mathbf{A}(k)$. Le modèle développé utilise la pseudo réponse impulsionnelle. La sortie $b1(n)$ du premier sous-système peut s'écrire sous la forme suivante :

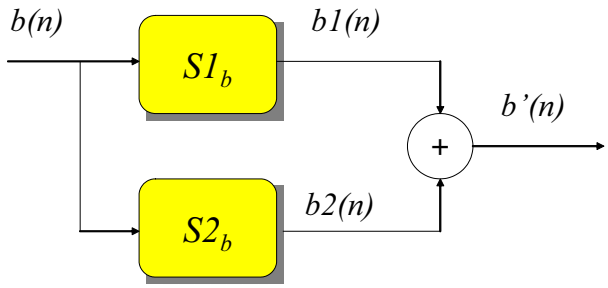

FIG. 3.16 – *Composition de sous-systèmes en parallèle*

$$b1(n) = \sum_{k=0}^{n} \mathbf{A}_1(k) b(k) \mathbf{D}_1(k) \tag{3.78}$$

La sortie $b2(n)$ du second sous-système est égale à :

$$b2(n) = \sum_{k=0}^{n} \mathbf{A}_2(k) b(k) \mathbf{D}_2(k) \tag{3.79}$$

Or la sortie de la mise en parallèle des deux sous-systèmes est définie par la somme des sorties de chaque sous-système.

$$b'(n) = b1(n) + b2(n) \tag{3.80}$$

Le bruit de sortie est alors égal à :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}_1(k) b(k) \mathbf{D}_1(k) + \sum_{k=0}^{n} \mathbf{A}_2(k) b(k) \mathbf{D}_2(k) \tag{3.81}$$

Cette expression ne peut être factorisée. Pour simplifier l'écriture, de nouvelles notations sont employées concernant la factorisation de matrices. Considérons le cas où un terme commun est présent dans deux expressions mais où la factorisation classique est impossible. Par exemple, le terme $\mathbf{A}_1(n) b(n) \mathbf{D}_1(n) + \mathbf{A}_2(n) b(n) \mathbf{D}_2(n)$ où chaque composant est matriciel ne peut se factoriser alors que le terme $b(n)$ est présent dans chacune des deux opérations. Cette expression est alors notée

$$\mathbf{A}_1(n) b(n) \mathbf{D}_1(n) + \mathbf{A}_2(n) b(n) \mathbf{D}_2(n) = \underline{\mathbf{A}}(n) b(n) \underline{\mathbf{D}}(n) \tag{3.82}$$

avec $\underline{\mathbf{A}}(n) = \binom{\mathbf{A}_1(n)}{\mathbf{A}_2(n)}$ et $\underline{\mathbf{D}}(n) = \binom{\mathbf{D}_1(n)}{\mathbf{D}_2(n)}$

De manière plus générale si la somme $y(n)$ de plusieurs produits de termes matriciels est représentée comme sur l'équation suivante :

$$y(n) = \sum_{i=1}^{M} \mathbf{A}_i(n) b(n) \mathbf{D}_i(n) \tag{3.83}$$

alors, la notation suivante peut être adoptée :

$$y(n) = \underbrace{\begin{pmatrix} \mathbf{A}_1(n) \\ \vdots \\ \mathbf{A}_M(n) \end{pmatrix}}_{\underline{\mathbf{A}}(n)} b(n) \underbrace{\begin{pmatrix} \mathbf{D}_1(n) \\ \vdots \\ \mathbf{D}_M(n) \end{pmatrix}}_{\underline{\mathbf{D}}(n)} \tag{3.84}$$

Ainsi, les termes $\underline{\mathbf{A}}(n)$ et $\underline{\mathbf{D}}(n)$ sont des juxtapositions de matrices de mêmes dimensions. Si un élément de $\underline{\mathbf{A}}(n)$ a pour taille PxQ, alors $\underline{\mathbf{A}}(n)$ a pour taille aPxQ, où a correspond au nombre de termes compris dans $\underline{\mathbf{A}}(n)$. Les composants $\underline{\mathbf{A}}(n)$ et $\underline{\mathbf{D}}(n)$ doivent avoir le même nombre de termes.

Avec les notations présentées et en posant

$$\underline{\mathbf{A}}(n) = \begin{pmatrix} \mathbf{A}_1(n) \\ \mathbf{A}_2(n) \end{pmatrix} \tag{3.85}$$

$$\underline{\mathbf{D}}(n) = \begin{pmatrix} \mathbf{D}_1(n) \\ \mathbf{D}_2(n) \end{pmatrix} \tag{3.86}$$

la sortie de la mise en parallèle de deux sous-systèmes peut finalement s'écrire sous la forme suivante :

$$b'(n) = \sum_{k=0}^{n} \underline{\mathbf{A}}(k) b(k) \underline{\mathbf{D}}(k) \tag{3.87}$$

Dans le cas où plus de deux sous-systèmes sont en parallèle, l'équation précédente est généralisée.

Composition de sous-systèmes en série

Dans cette partie, le cas de deux sous-systèmes placés en série est étudié (figure 3.17). La démarche consiste à utiliser la pseudo réponse impulsionnelle du système. L'instant de sortie du bruit n'étant plus forcément égal à n, la notation $\mathbf{A}_{(k)}^{(n)}$ est reprise.

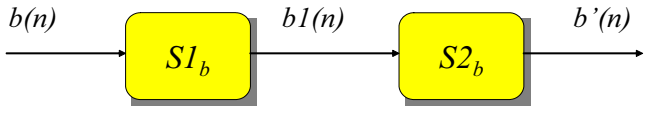

FIG. 3.17 – *Composition de sous-systèmes en série*

La sortie du premier sous-système $b1(n)$ est égale à :

$$b1(n) = \sum_{k=0}^{n} \mathbf{A}_{1(k)}^{(n)} b(k) \mathbf{D}_{1(k)}^{(n)} \tag{3.88}$$

La sortie $b'(n)$ s'exprime en fonction de $b1(n)$ sous la forme suivante :

$$b'(n) = \sum_{k=0}^{n} \mathbf{A}_{2(k)}^{(n)} b1(k) \mathbf{D}_{2(k)}^{(n)} \tag{3.89}$$

où $\mathbf{A}_{2(k)}^{(n)}$ et $\mathbf{D}_{2(k)}^{(n)}$ désignent les termes de signaux multipliant le bruit $b1(k)$ à l'instant n. En introduisant l'équation (3.88) dans l'expression (3.89), la relation suivante est obtenue :

$$b'(n) = \sum_{k=0}^{n} \sum_{j=0}^{k} \mathbf{A}_{2(k)}^{(n)} \mathbf{A}_{1(j)}^{(k)} b(j) \mathbf{D}_{1(j)}^{(k)} \mathbf{D}_{2(k)}^{(n)} \tag{3.90}$$

Pour simplifier cette expression, les notations développées dans le paragraphe précédent sont utilisées. Si les termes suivants sont introduits :

$$\underline{\mathbf{A}}_{(k)}^{(n)} = \begin{pmatrix} \mathbf{A}_{2(k)}^{(n)} \mathbf{A}_{1(k)}^{(k)} \\ \vdots \\ \mathbf{A}_{2(k)}^{(n)} \mathbf{A}_{1(j)}^{(k)} \\ \vdots \\ \mathbf{A}_{2(k)}^{(n)} \mathbf{A}_{1(0)}^{(k)} \end{pmatrix} \tag{3.91}$$

$$\underline{\mathbf{D}}_{(k)}^{(n)} = \begin{pmatrix} \mathbf{D}_{2(k)}^{(n)} \mathbf{D}_{1(k)}^{(k)} \\ \vdots \\ \mathbf{D}_{2(k)}^{(n)} \mathbf{D}_{1(j)}^{(k)} \\ \vdots \\ \mathbf{D}_{2(k)}^{(n)} \mathbf{D}_{1(0)}^{(k)} \end{pmatrix} \tag{3.92}$$

l'expression (3.90) peut se réécrire sous la forme suivante

$$b'(n) = \sum_{i=0}^{n} \underline{\mathbf{A}}_{(k)}^{(n)} b(i) \underline{\mathbf{D}}_{(k)}^{(n)} \tag{3.93}$$

La mise en série de sous-systèmes permet de modéliser la propagation du bruit sous la forme d'une multiplication par deux termes de signaux $\underline{\mathbf{A}}_{(k)}^{(n)}$ et $\underline{\mathbf{D}}_{(k)}^{(n)}$. Ces termes s'obtiennent en multipliant les pseudo réponses impulsionnelles de chaque sous-système.

Modèle général issu de la composition

D'un point de vue général, la composition de systèmes permet d'écrire la sortie d'un circuit de la façon suivante :

$$b'(n) = \sum_{k=0}^{n} \underline{\mathbf{A}}(k) b(k) \underline{\mathbf{D}}(k) \tag{3.94}$$

Les termes $\underline{\mathbf{A}}$ et $\underline{\mathbf{D}}$ sont plus complexes si le nombres de sous-systèmes empruntés par le bruit est important. Cette équation représente la contribution d'une source de bruit à la sortie du système global. En reprenant les notations initiales ($b_i(n)$ bruit d'entrée et $b'_{ui}(n)$ bruit de sortie du circuit), le bruit en sortie dû à la propagation d'un bruit en entrée à travers le système est alors égal à :

$\forall (i,u) \in [1 : Ne] \cap [1 : Ns]$

$$b'_{ui}(n) = \sum_{k=0}^{n} \underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) b_i^t(k) \underline{\mathbf{V}}_{ui}(k) \tag{3.95}$$

en séparant les termes pour lesquels le bruit est transposé. Finalement, le bruit en sortie est égal à la somme de toutes les contributions de l'ensemble des bruits d'entrée.

$$\begin{aligned} b_{yu}(n) &= \sum_{i=1}^{Ne} b'_{ui}(n) \\ &= \sum_{i=1}^{Ne}\sum_{k=0}^{n} \underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) b_i^t(k) \underline{\mathbf{V}}_{ui}(k) \end{aligned} \tag{3.96}$$

Les termes de bruit b_i et b_{yu} peuvent être de taille quelconque (scalaire, vectoriel ou matriciel). Les termes de signaux $\underline{\mathbf{A}}_{ui}(k)$, $\underline{\mathbf{D}}_{ui}(k)$, $\underline{\mathbf{U}}_{ui}(k)$ et $\underline{\mathbf{V}}_{ui}(k)$ ont des dimensions s'adaptant aux bruits. Ils peuvent être scalaires, vectoriels ou matriciels. S'ils sont scalaires, alors l'expression peut être simplifiée en mettant les termes $\underline{\mathbf{A}}_{ui}(k)$ et $\underline{\mathbf{D}}_{ui}(k)$ du même coté. Cette simplification permet d'écrire la propagation du bruit comme la multiplication par un seul terme de signaux égal à la multiplication des termes $\underline{\mathbf{A}}_{ui}(k)$ et $\underline{\mathbf{D}}_{ui}(k)$. Néanmoins, ces termes de signaux sont considérés dans le cas général comme des matrices car cette modélisation permet d'exprimer de manière plus simple le bruit de sortie.

Exemple de La Transformée de Fourier Rapide (FFT)

Dans cette partie, l'exemple de la Transformée de Fourier Rapide (FFT) est traité. Soit un signal $x(n)$ dont la Transformée de Fourier classique $X(k)$ de ce dernier définie sur N points est donnée par la relation suivante :

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{(-2j\pi \frac{nk}{N})} \tag{3.97}$$

Cette équation peut être écrite sous la forme matricielle suivante :

$$\begin{pmatrix} X(0) \\ \vdots \\ X(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & \dots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \dots & W_N^{(N-1)^2} \end{pmatrix} \times \begin{pmatrix} x(0) \\ \vdots \\ x(N-1) \end{pmatrix} \tag{3.98}$$

où $W_N^k = e^{(-2j\frac{k\pi}{N})}$. L'objectif est de modéliser la propagation des bruits au sein de ce système en utilisant le modèle précédemment développé.

Modèle d'un papillon Pour accélérer le traitement de la Transformée de Fourier classique, la complexité est réduite en utilisant des papillons. Ceux-ci sont représentés sur la figure 3.18. Un papillon est composé de deux entrées complexes $P(n)$ et $Q(n)$ et de deux sorties $P(n+1)$ et $Q(n+1)$. L'expression de la sortie en fonction de l'entrée est la suivante :

$$\begin{aligned} P(n+1) &= P(n) + W_N^l Q(n) \\ Q(n+1) &= P(n) - W_N^l Q(n) \end{aligned} \tag{3.99}$$

Dans le cadre d'une implantation en précision finie, les bruits de quantification sont générés (figure 3.19). Le terme x présent dans les sinus et cosinus désigne l'angle $x = -2\frac{l\pi}{N}$ défini sur la figure 3.18. Pour simplifier les notations, les bruits sont regroupés sous la forme d'un bruit généré par l'addition b_{add} et d'un autre généré par les multiplications b_m comme l'exprime la relation suivante :

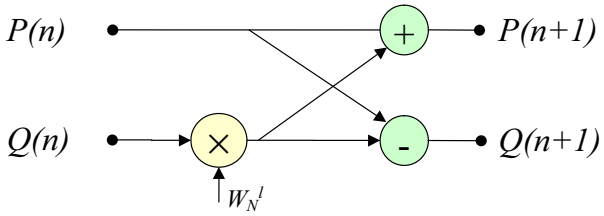

FIG. 3.18 – *Modélisation d'un papillon de FFT*

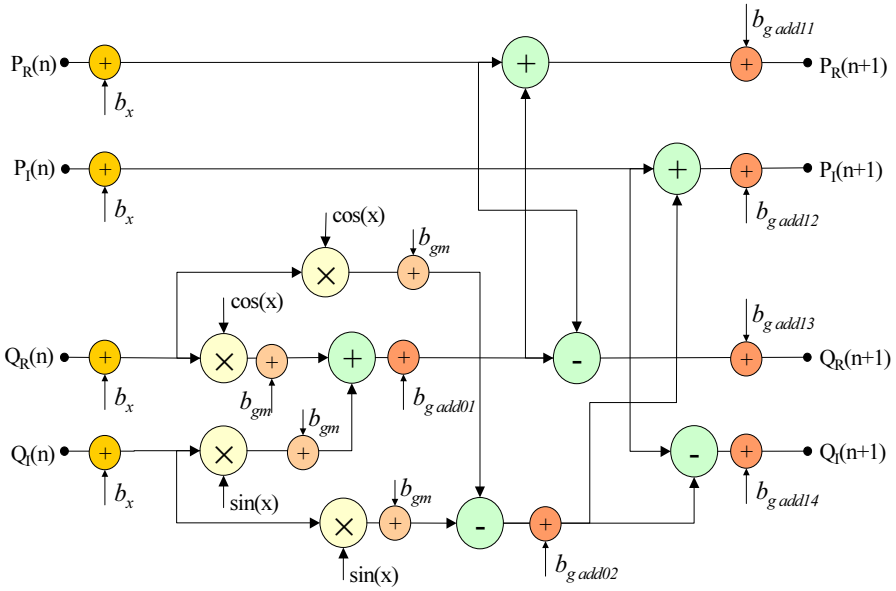

FIG. 3.19 – *Modélisation de la FFT en précision finie*

$$\begin{aligned}
b_{m1} &= b_x(cos(x) - sin(x)) \\
b_{m2} &= b_x(cos(x) + sin(x)) + 2.b_{gm} \\
b_{m3} &= b_x(sin(x) - cos(x)) \\
b_{m4} &= -b_x(sin(x) + cos(x)) + 2.b_{gm} \\
b_{add1} &= b_{gadd11} + b_{gadd01} + b_x \\
b_{add2} &= b_{gadd12} + b_{gadd02} + b_x \\
b_{add3} &= b_{gadd13} + b_{gadd03} + b_x \\
b_{add4} &= b_{gadd14} + b_{gadd04} + b_x
\end{aligned} \tag{3.100}$$

Ainsi, les sorties du papillons sont données par l'équation suivante :

$$\begin{aligned}
P_R(n+1) &= P_R(n) + cos(x)Q_R(n) - sin(x)Q_I(n) + b_{add1} + b_{m1} \\
P_I(n+1) &= P_I(n) + sin(x)Q_R(n) + cos(x)Q_I(n) + b_{add2} + b_{m2} \\
Q_R(n+1) &= P_R(n) - cos(x)Q_R(n) + sin(x)Q_I(n) + b_{add3} - b_{m3} \\
Q_I(n+1) &= P_I(n) - sin(x)Q_R(n) - cos(x)Q_I(n) + b_{add4} - b_{m4}
\end{aligned} \tag{3.101}$$

ou encore en les regroupant sous la forme complexe

$$\begin{aligned} P(n+1) &= P(n) + W_N^l Q(n) + b1_{add} + b1_m \\ Q(n+1) &= P(n) - W_N^l Q(n) + b2_{add} - b2_m \end{aligned} \tag{3.102}$$

avec les bruits complexes suivants :

$$\begin{aligned} b1_{add} &= b_{add1} + jb_{add2} \\ b2_{add} &= b_{add3} + jb_{add4} \\ b1_m &= b_{m1} + jb_{m2} \\ b2_m &= b_{m3} + jb_{m4} \end{aligned} \tag{3.103}$$

Au sein d'un papillon, la génération des bruits peut se modéliser à partir de l'expression précédente dans laquelle les termes de bruit $b1_{add}$, $b2_{add}$, $b1_m$ et $b2_m$ sont complexes. Pour modéliser complètement le système, la propagation des bruits en entrée à travers le papillon est étudiée. Soit $p(n)$ et $q(n)$, les bruits associés au signaux $P(n)$ et $Q(n)$. Ceux-ci se propagent selon l'expression suivante :

$$\begin{aligned} p(n+1) &= p(n) + W_N^l q(n) \\ p(n+1) &= p(n) - W_N^l q(n) \end{aligned} \tag{3.104}$$

Cette modélisation peut également s'écrire sous forme matricielle. Or, la FFT est composée de plusieurs niveaux de papillons composés chacun de plusieurs papillons (figure 3.20). Pour avoir un modèle plus clair, l'ensemble des papillons de chaque niveau doit être pris en compte.

Modèle du premier niveau de papillons La FFT est composé de $log2(N)$ niveaux de papillons et chaque niveau comprend $\frac{N}{2}$ papillons. Les bruits associés aux termes X initiaux se propagent dans le premier niveau de papillons. Soit B le vecteur bruit associé aux données d'entrée X. La propagation des B à travers le premier niveau de papillons se modélise grâce à l'expression (3.104) sous forme matricielle. Soit $\mathbf{H}_1$ la matrice modélisant la propagation des b au sein du premier niveau de papillons. Celle-ci s'exprime sous la forme suivante :

$$\mathbf{H}_1 = \begin{pmatrix} \begin{pmatrix} 1 & W_N^0 \\ 1 & -W_N^0 \end{pmatrix} & 0 & \cdots & & 0 \\ 0 & \cdots & 0 & \begin{pmatrix} 1 & W_N^0 \\ 1 & -W_N^0 \end{pmatrix} & 0 \ \cdots & 0 \\ & & \ddots & & & \\ 0 & \cdots & & 0 & \begin{pmatrix} 1 & W_N^0 \\ 1 & -W_N^0 \end{pmatrix} \end{pmatrix} \tag{3.105}$$

Cette matrice permet d'exprimer la valeur des bruits en sortie du premier niveau de papillon en fonction de tous les bruits présents en entrée. En sortie de ce premier niveau de papillons, sont présents les bruits d'entrée propagés tout comme les bruits générés par les papillons (b_{add} et b_m). Tous ces termes de bruits se propagent ensuite dans le deuxième niveau de papillons puis le troisième et ainsi de suite jusqu'à la sortie du système. La propagation complète nécessite de modéliser tous les niveaux de papillons.

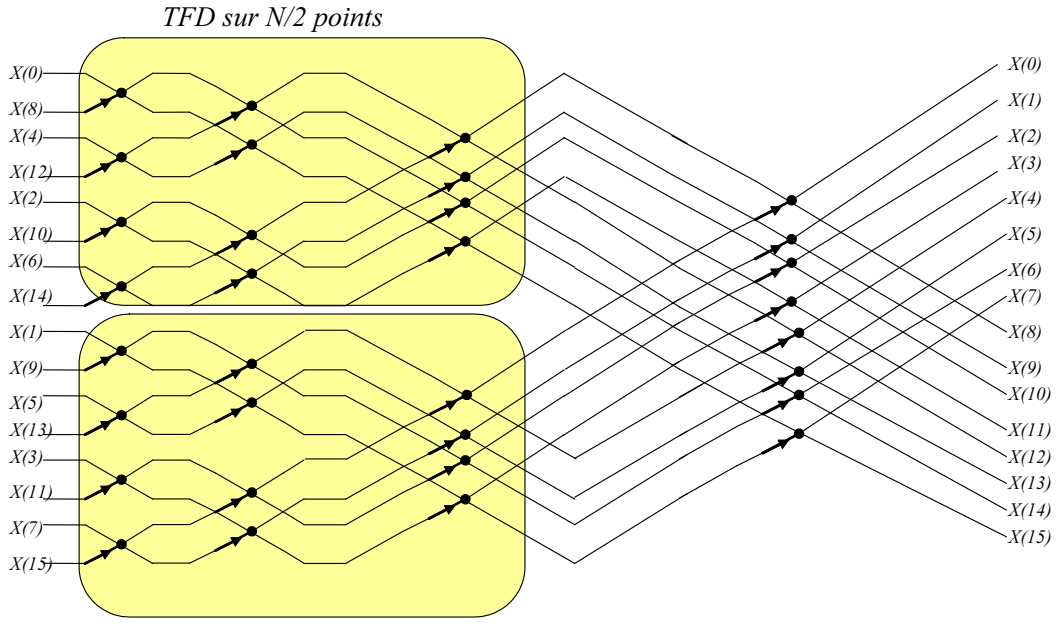

FIG. 3.20 – *Modélisation de la FFT (N=16)*

Modèle du $i^{\text{ème}}$ niveau de papillons Tous les niveaux de papillons doivent être modélisés pour obtenir la propagation complètes des bruits présents dans le système. Ainsi, dans ce paragraphe la modélisation du $i^{\text{ème}}$ niveau de papillons est effectuée pour $1 \leq i < log2(N)$. Soit $\mathbf{H}_i$ la matrice de propagation au sein du $i^{\text{ème}}$ niveau de papillons. Pour exprimer cette matrice, la matrice intermédiaire carrée $\mathbf{M}_i$ de taille 2^i est introduite.

$$\mathbf{M}_i = \begin{pmatrix} \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} W^0 & 0 & \dots & 0 \\ 0 & W^2 & \dots & 0 \\ \vdots & & & \\ 0 & \dots & 0 & W^{2(2^{i-1}-1)} \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} -W^0 & 0 & \dots & 0 \\ 0 & -W^2 & \dots & 0 \\ \vdots & & & \\ 0 & \dots & 0 & -W^{2(2^{i-1}-1)} \end{pmatrix} \end{pmatrix} \tag{3.106}$$

Cette matrice $\mathbf{M}_i$ est composée de 4 blocs distincts. En notant $\mathbf{I_{2^{i-1}}}$, la matrice identité de taille 2^{i-1} et $\mathbf{L}_i$, la matrice définie par l'expression suivante :

$$\mathbf{L}_i = \begin{pmatrix} W^0 & 0 & \dots & 0 \\ 0 & W^2 & \dots & 0 \\ \vdots & & & \\ 0 & \dots & 0 & W^{2(2^{i-1}-1)} \end{pmatrix} \tag{3.107}$$

la matrice $\mathbf{M}_i$ peut s'écrire sous la forme

$$\mathbf{M}_i = \begin{pmatrix} \mathbf{I_{2^{i-1}}} & \mathbf{L_i} \\ \mathbf{I_{2^{i-1}}} & \mathbf{-L_i} \end{pmatrix} \tag{3.108}$$

Cette matrice permet ensuite de définir la matrice $\mathbf{H_i}$ de la façon suivante

$$\mathbf{H}_i = \begin{pmatrix} \mathbf{M}_i & 0 & 0 & \dots & 0 \\ 0 & \mathbf{M}_i & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \mathbf{M}_i \end{pmatrix} \tag{3.109}$$

La matrice $\mathbf{M}_i$ est de taille 2^i alors que la matrice $\mathbf{H}_i$ est de taille N. La matrice $\mathbf{M}_i$ apparaît donc $\frac{N}{2^i}$ fois sur la diagonale de la matrice $\mathbf{H}_i$.

Composition en série Le bruit situé à l'étage k passe dans les matrices de propagation $\mathbf{H}_{k+1}.....\mathbf{H}_{log2(N)}$ pour aboutir à la sortie du système. Ainsi, la propagation du bruit se modélise comme la propagation à travers la mise en série de matrices $\mathbf{H}_i$.

La contribution d'un bruit $B_i(n)$ se modélise comme le produit des matrices $\mathbf{H}_k$ rencontrées comme expliquée dans la section 3.2.2 sur la composition de systèmes. En effet, la mise en série de systèmes se modélise en multipliant leur pseudo réponse impulsionnelle. Ici la pseudo réponse impulsionnelle est simplement modélisée par une matrice $\mathbf{H}_k$. La mise en série des ces papillons est donc analysée en multipliant ces matrices $\mathbf{H}_k$. Le bruit en sortie de la FFT, $B_y(n)$, est alors la somme de toutes les sources de bruit apparaissant à chaque niveau de papillons, puis se propageant dans les papillons suivants. Ceci peut se modéliser sous la forme suivante :

$$B_y(n) = \sum_{i=0}^{log2(N)} \mathbf{F}_i B_i(n) \tag{3.110}$$

où $\mathbf{F}_i$ est défini par :

$$\begin{aligned} \mathbf{F}_i &= \prod_{j=log2(N)}^{i+1} \mathbf{H}_j \\ \mathbf{F}_0 &= \prod_{j=log2(N)}^{0} \mathbf{H}_j \end{aligned} \tag{3.111}$$

Le bruit $B_i(n)$ est un vecteur de taille N généré au $i^{\text{ème}}$ étage de papillons où chaque terme est composé de bruits d'addition b_{add} et de bruits de multiplication b_m apparaissant en sortie du papillon précédent. La matrice $\mathbf{F}_0$ définissant la propagation des bruits d'entrée est différente. En effet cette matrice correspond à la multiplication des matrices $\mathbf{H}_i$ et de la matrice $\mathbf{H}_0$. La matrice $\mathbf{H}_0$ définit le passage des données d'entrée en bit-reversed. Ainsi, celle-ci affecte uniquement le bruit d'entrée. Elle réorganise les échantillons d'entrée de façon à permettre l'exploitation de la FFT. Cette matrice $\mathbf{F}_0$ définit la propagation des bruits d'entrée à travers tout le système. En calculant $\mathbf{F}_0$, celle-ci est égale à :

$$\mathbf{F}_0 = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & \dots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \dots & W_N^{(N-1)^2} \end{pmatrix} \tag{3.112}$$

Elle correspond à la matrice modélisant le système complet et permet de vérifier la validité du modèle. L'exemple de la FFT à permis de montrer l'intérêt de modèle matriciel pour exprimer la propagation des bruits. L'expression obtenue est nettement plus compacte que celles développées dans la littérature [83]. De plus, cet exemple illustre la mise en série de différents blocs.

3.3 Expression de la puissance du bruit

La conversion de précision infinie en précision finie se traduit par la présence de bruits influant sur les performances de l'application. La qualité des calculs est mesurée dans ce modèle par l'expression du Rapport Signal à Bruit de Quantification (RSBQ). Alors que la puissance du signal de sortie peut être déterminée par une unique simulation en virgule flottante, la puissance du bruit est exprimée de manière analytique. Dans cette partie, le calcul de la puissance du bruit en sortie et les autres termes statistiques de ce bruit sont détaillés.

3.3.1 Approche théorique de la détermination de la puissance du bruit

Dans la partie 3.1 ont été présentées les caractéristiques statistiques des bruits en entrée du système. Dans cette partie, les propriétés statistiques du bruit de sortie sont déterminées en fonction de celles des bruits d'entrée en utilisant le modèle présenté dans l'expression (3.96). L'objectif est de pouvoir exprimer sa moyenne et son autocorrélation en fonction des caractéristiques statistiques du bruit d'entrée et du système considéré. Le bruit de sortie $b_{yu}(n)$ s'exprime en fonction des bruits d'entrée $b_i(n)$ sous la forme suivante :

$$b_{yu}(n) = \sum_{i=1}^{Ne} \sum_{k=0}^{n} \underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) b_i^t(k) \underline{\mathbf{V}}_{ui}(k) \tag{3.113}$$

Ces bruits sont soit des scalaires, des vecteurs ou des matrices. De ce fait, M_{b_i} et N_{b_i} sont les dimensions du bruit d'entrée $b_i(n)$ et M_{y_u} et N_{y_u} celles du bruit de sortie $b_{yu}(n)$. Pour cette étude, les bruits sont considérés non-corrélés aux signaux. Cette hypothèse n'est pas simplificatrice mais découle directement des caractéristiques statistiques des bruits définis en section 1.2. D'autre part, les termes transposés sont pris en compte pour le calcul des statistiques.

Moyenne

L'expression de la moyenne $m_{b_{yu}} = E(b_{yu}(n))$ du bruit de sortie u est trouvée en repartant de l'équation (3.113). En prenant la moyenne de cette dernière, l'équation suivante est obtenue.

$$m_{b_{yu}} = \sum_{i=1}^{Ne} \sum_{k=0}^{n} E\Big(\underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) b_i^t(k) \underline{\mathbf{V}}_{ui}(k) \Big) \tag{3.114}$$

La non-corrélation entre le bruit et le signal permet la simplification suivante : $E(AbB) = E(AE(b)B)$ [19]. De ce fait, l'expression de moyenne du bruit de sortie est la suivante :

$$m_{b_{yu}} = \sum_{i=1}^{Ne} \sum_{k=0}^{n} E\Big(\underline{\mathbf{A}}_{ui}(k) m_{b_i} \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) m_{b_i^t} \underline{\mathbf{V}}_{ui}(k) \Big) \tag{3.115}$$

avec m_{b_i} désignant la moyenne du bruit $b_i(n)$.

Auto-corrélation

La matrice d'auto-corrélation $\mathbf{R}_{b_{yu}b_{yu}}(\theta)$ du terme de sortie $b_{yu}(n)$ est de taille M_{y_u}xM_{y_u} et est donnée par la relation suivante :

$$\mathbf{R}_{b_{yu}b_{yu}}(\theta) = E(b_{yu}(n) b_{yu}^t(n-\theta)) \tag{3.116}$$

Pour calculer cette expression, le bruit de sortie est considéré aux instants n et $n-\theta$. Ainsi les termes matriciels devraient s'écrire $\underline{\mathbf{A}}_{ui(k)}^{(n)}$ ou $\underline{\mathbf{A}}_{ui(m)}^{(n-\theta)}$ pour différencier les instants de sortie. Pour simplifier, ces termes s'écrivent $\underline{\mathbf{A}}_{ui}(k)$ et $\underline{\mathbf{A}}_{ui}(m)$. L'indice k étant celui lié à l'instant de sortie n et l'indice m, celui de l'instant de sortie $n-\theta$. Ces deux indices k et m sont toujours utilisés

dans ces conditions pour les termes statistiques développés ci-dessous.

Pour exprimer la valeur de cette autocorrélation, le cas où les bruits transposés n'apparaissent pas est pris en compte dans un premier temps. De ce fait, l'expression du bruit de sortie est la suivante :

$$b_{yu}(n) = \sum_{i=1}^{Ne} \sum_{k=0}^{n} \underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) \tag{3.117}$$

La matrice d'autocorrélation des bruits d'entrée $\mathbf{R}_{b_i b_j}(\theta)$ s'exprime alors de la façon suivante :

$$\begin{aligned} \mathbf{R}_{b_{yu}b_{yu}}(\theta) &= E(b_{yu}(n) b_{yu}^t(n-\theta)) && (3.118) \\ &= \sum_{i,j=1}^{Ne} \sum_{k=0}^{n} \sum_{m=0}^{n-\theta} E\left[\underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) \underline{\mathbf{D}}_{uj}^t(m) b_j^t(m) \underline{\mathbf{A}}_{uj}^t(m)\right] && (3.119) \end{aligned}$$

Pour simplifier le développement de l'expression précédente, des notations simplificatrices sont introduites. Les termes de signaux sont remplacés par $\underline{\mathbf{A}}_{ui}(k) = A(k)$ et $\underline{\mathbf{D}}_{ui}(k) = D(k)$. De plus, le développement est effectué dans le cas où les bruits d'entrée $b_i(n)$ sont des vecteurs. Si ces bruits sont matriciels, la même expression est obtenue. Ce cas est démontré en annexe A. Les bruits étant considérés comme des vecteurs, les bruits $b_i(n)$ et $b_j(n)$ ont pour dimensions M_{b_i}x1 et M_{b_j}x1. Les dimensions des termes de signaux s'adaptent aux tailles des bruits. Les termes $A(k)$, $A(m)$, $D(k)$ et $D(m)$ ont pour dimensions respectivement M_{y_u}xM_{b_i}, M_{yu}xM_{b_j}, 1xN_{yu} et 1xN_{y_u}. La matrice $\mathbf{S}$, de taille M_{y_u}xM_{y_u} et de valeur $\mathbf{S} = E[A(k)b_i(k)D(k)D^t(m)b_j^t(m)A^t(m)]$ est introduite. Pour déterminer complètement cette matrice, chacun de ces termes est calculé. Soient $(a,b) \in [1 : M_{y_u}]^2$ et $\mathbf{S}_{(a,b)}$ définissant l'élément de la $a^{\text{ème}}$ ligne et $b^{\text{ème}}$ colonne de $\mathbf{S}$. Cet élément s'obtient à l'aide du produit des termes contenus dans $\mathbf{S}$.

$$\mathbf{S}_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}} \sum_{d=1}^{N_{y_u}} \sum_{e=1}^{M_{b_j}} A_{(a,c)}(k) b_{i_{(c,1)}}(k) D_{(1,d)}(k) D_{(d,1)}^t(m) b_{j_{(1,e)}}^t(m) A_{(e,b)}^t(m)\right] \tag{3.120}$$

En réordonnant les termes, l'expression suivante est obtenue.

$$\mathbf{S}_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}} \sum_{d=1}^{N_{y_u}} \sum_{e=1}^{M_{b_j}} A_{(a,c)}(k) b_{i_{(c,1)}}(k) b_{j_{(1,e)}}^t(m) D_{(1,d)}(k) D_{(d,1)}^t(m) A_{(e,b)}^t(m)\right] \tag{3.121}$$

Le terme $b_{i_{c,1}}(k) b_{j_{1,e}}^t(m)$ dans ce produit de matrices est retrouvé. D'après le modèle de bruit, les termes de bruit et de signaux sont non-corrélés. En utilisant la simplification $E(AbB) = E(AE(b)B)$ utilisée pour le calcul de la moyenne, l'équation suivante est obtenue

$$\mathbf{S}_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}} \sum_{d=1}^{N_{y_u}} \sum_{e=1}^{M_{b_j}} A_{(a,c)}(k) E\left(b_{i_{(c,1)}}(k) b_{j_{(1,e)}}^t(m)\right) D_{(1,d)}(k) D_{(d,1)}^t(m) A_{(e,b)}^t(m)\right] \tag{3.122}$$

Le terme $E\left(b_{i_{(c,1)}}(k) b_{j_{(1,e)}}^t(m)\right)$ correspond exactement à l'élément de la $c^{\text{ème}}$ ligne et $e^{\text{ème}}$ colonne de la matrice d'intercorrélation entre les bruits b_i et b_j

$$E(b_{i_{(c,1)}}(k) b_{j_{(1,e)}}^t(m)) = (\mathbf{R}_{b_i b_j}(m-k))_{(c,e)} \tag{3.123}$$

Or, la matrice d'intercorrélation est égale à la somme de la matrice d'intercovariance et de la matrice produit des moyennes soit $\mathbf{R}_{b_i b_j}(m-k) = \mathbf{C}_{b_i b_j}(m-k) + m_{b_i} m_{b_j}^t$.

Ces termes sont introduits dans l'expression (3.122) pour aboutir à la formule suivante :

$$\begin{aligned}
\mathbf{S}_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{yu}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\left(\mathbf{C}_{b_ib_j}(m-k)+m_{b_i}m_{b_j}^t\right)_{(c,e)} D_{(1,d)}(k)D_{(d,1)}^t(m)A_{(e,b)}^t(m)\right] \\
&= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{yu}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\mathbf{C}_{b_ib_j}(m-k)_{(c,e)}D_{(1,d)}(k)D_{(d,1)}^t(m)A_{(e,b)}^t(m)\right] \\
&+ E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{yu}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)m_{b_{i_{(c,1)}}}D_{(1,d)}(k)D_{(d,1)}^t(m)m_{b_{j_{(1,e)}}}^t A_{(e,b)}^t(m)\right]
\end{aligned} \tag{3.124}$$

Les termes de signaux sont alors réorganisés. L'expression (3.124) s'écrit alors sous la forme :

$$\begin{aligned}
\mathbf{S}_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{yu}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\mathbf{C}_{b_ib_j}(m-k)_{(c,e)}D_{(1,d)}(k)D_{(d,1)}^t(m)A_{(e,b)}^t(m)\right. \\
&\left.+\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{yu}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)m_{b_i}(k)_{(c,1)}D_{(1,d)}(k)D_{(d,1)}^t(m)m_{b_j}^t(m)_{(1,e)}A_{(e,b)}^t(m)\right] \\
&= E\left[\underbrace{\sum_{c=1}^{M_{b_i}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\mathbf{C}_{b_ib_i}(m-k)_{(c,e)}A_{(e,b)}^t(m)}_{\left[A(k)\mathbf{C}_{b_ib_j}(m-k)A^t(m)\right]_{(a,b)}}\underbrace{\sum_{d=1}^{N_{yu}} D_{(1,d)}(k)D_{(d,1)}^t(m)}_{Tr(D(k)D^t(m))}\right] \\
&+ E\left[A(k)m_{b_i}(k)D(k)D^t(m)m_{b_j}^t(m)A^t(m)\right]_{(a,b)} \\
\mathbf{S} &= E\left[A(k)\mathbf{C}_{b_ib_j}(m-k)A^t(m)Tr(D(k)D^t(m))+A(k)m_{b_i}(k)D(k)D^t(m)m_{b_j}^t(m)A^t(m)\right]
\end{aligned} \tag{3.125}$$

Chaque élément de la matrice $\mathbf{S}$ est déterminé. La matrice d'autocorrélation du bruit de sortie est alors égale à :

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m)] \\
&= \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(Tr[(\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m))]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right)}_{\text{Terme de covariance}} \\
&+ \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{A}}_{uj}^t(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{3.126}$$

Ce terme peut être analysé comme étant composé de deux parties :

– Une partie regroupant des termes de moyenne correspondant à la moyenne au carré du bruit de sortie $b_{yu}(n)$

– Une partie intégrant des termes de covariance définissant la covariance du bruit de sortie.

Ainsi, la matrice d'autocorrélation est égale à la somme de la matrice de covariance et de la matrice de moyenne au carré. Dans le cas où les termes de transposition sont pris en compte, le développement est présenté en annexe A. L'expression de l'autocorrélation du bruit de sortie est définie dans l'expression suivante :

$$\begin{aligned}
\mathbf{R}_{b_{yu}b_{yu}}(\theta) &= E[b_{yu}(n)b_{yu}^t(n-\theta)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(Tr\big[(\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m))\big]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\Big[\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\Big] \\
&\cdot\Big[\underline{\mathbf{A}}_{uj}(m)m_{b_j}\underline{\mathbf{D}}_{uj}(m) + \underline{\mathbf{U}}_{uj}(m)m_{b_j}^t\underline{\mathbf{V}}_{uj}(m)\Big]^t\Big)
\end{aligned} \tag{3.127}$$

Au sein d'un système, la grande majorité des bruits apparaissant sont des bruits de quantification issus d'un changement de format au sein du système. Les seuls termes de bruit pouvant avoir des caractéristiques statistiques différentes sont les bruits associés aux données d'entrée. En effet, si les données d'entrée proviennent de la sortie d'un autre système, les bruit associés n'ont plus les caractéristiques des bruits blancs. Ainsi, les termes statistiques des bruits d'entrée sont simplifiés et la complexité de la relation donnant la puissance du bruit de sortie est fortement diminuée. Par exemple, les matrices d'intercovariances sont nulles dans ce cas. Les simplifications engendrées par les bruits blancs sont présentées par la suite. La valeur de la puissance du bruit est obtenue pour une valeur nulle de θ. Ainsi, le puissance du bruit est calculée et le RSBQ de l'application est déterminé à l'aide de cette formule.

Covariance

La matrice de covariance $\mathbf{C}_{b_{yu}b_{yu}}(\theta)$ s'obtient en soustrayant les équations (3.127) et (3.115) au carré. Ainsi, la covariance du bruit de sortie évolue dans le temps selon l'expression suivante :

$$\mathbf{C}_{b_{yu}b_{yu}}(\theta) = \mathbf{R}_{b_{yu}b_{yu}}(\theta) - m_{b_{yu}}m_{b_{yu}}^t \tag{3.128}$$

Pour une valeur de θ nulle, la variance du bruit de sortie $b_{yu}(n)$ est retrouvée.

Corrélation entre les bruits

La sortie du système est composée de N_s bruits de sortie. La corrélation existant entre ces différents bruits de sortie peut être déterminée. Elle permet de regarder si les bruits en sortie du système sont indépendants entre eux. Cette corrélation dépend des bruits d'entrée mais également des pseudo réponses impulsionnelles inhérentes à la propagation des sources de bruit pour aboutir à une sortie. Soit deux bruits de sortie $b_{yu}(n)$ et $b_{yv}(n)$ $\forall u \neq v \in [1 : Ns]^2$. Ces matrices ont pour tailles $M_{b_{yu}}\mathrm{x}N_{b_{yv}}$ et sont calculées par la suite.

Intercorrélation entre deux bruits de sortie La matrice d'intercorrélation de deux bruits de sortie $b_{yu}(n)$ et $b_{yv}(n)$ est définie par :

$$\mathbf{R}_{b_{yu}b_{yv}}(\theta) = E(b_{yu}(n)b_{yv}^t(n-\theta)) \tag{3.129}$$

Son expression est obtenue en utilisant un développement identique au calcul de l'autocorrélation. L'intercorrélation entre deux bruits de sortie est alors égale à :

$$\begin{aligned}
\mathbf{R}_{b_{yu(n)}b_{yv}}(\theta) &= E[b_{yu}(n)b_{yv}^t(n-\theta)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(Tr\Big[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{vj}^t(m)\Big]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{vj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{vj}(m)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{vj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{vj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{vj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{vj}^t(m)Tr\Big[\underline{\mathbf{V}}_{vj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)\Big]\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\Big[\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\Big] \\
&\quad \cdot \Big[\underline{\mathbf{A}}_{vj}(m)m_{b_j}\underline{\mathbf{D}}_{vj}(m) + \underline{\mathbf{U}}_{vj}(m)m_{b_j}^t\underline{\mathbf{V}}_{vj}(m)\Big]^t\Big)
\end{aligned} \tag{3.130}$$

Intercovariance entre deux bruits de sortie L'intercovariance est la matrice définie par la différence entre la matrice d'intercorrélation et du produit des moyennes.

$$\mathbf{C}_{b_{yu}b_{yv}}(\theta) = \mathbf{R}_{b_{yu}b_{yv}}(\theta) - m_{b_{yu}}m_{b_{yv}}^t \tag{3.131}$$

Si les deux bruits de sortie sont non-corrélés, cette matrice est alors nulle pour tout θ.

Cas des bruits de quantification (bruits blancs)

Dans cette partie, le cas des bruits blancs est pris en considération. En effet, les bruits de quantification générés au sein du système sont des bruits blancs. Ceux-ci correspondent à la majorité des bruits présents dans le système. Leur modèle, présenté en section 1.2, définit ces termes comme des bruits blancs. Ainsi, traiter le cas des bruits blancs permet d'étudier les caractéristiques des bruits de sortie dans le cas où l'ensemble des bruits sont des bruits de quantification. Ces termes sont non corrélés les uns aux autres. De plus, les échantillons composant les bruits d'entrée $b_i(n)$ ont les mêmes caractéristiques statistiques (moyenne μ_{b_i} et variance $\sigma_{b_i}^2$). Ainsi les termes statistiques des bruits de sortie sont simplifiés.

La moyenne du bruit de sortie est définie par l'expression suivante :

$$m_{b_{yu}} = \sum_{i=1}^{Ne}\mu_{b_i}^2\sum_{k=0}^{n} E\Big(\underline{\mathbf{A}_i(k)}1\underline{\mathbf{D}_i(k)} + \underline{\mathbf{U}_i(k)}1^t\underline{\mathbf{V}_i(k)}\Big) \tag{3.132}$$

La matrice d'autocorrélation devient :

$$
\begin{aligned}
\mathbf{R}_{b_{yu}b_{yu}}(\theta) &= \sum_{i=1}^{Ne}\sigma_{b_i}^2\sum_{k=0}^{n-\theta}\Big[E\Big(Tr\big[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{ui}^t(k)\big]\underline{\mathbf{A}}_{ui}(k)\underline{\mathbf{A}}_{ui}^t(k)\Big)\\
&+ E\Big(\underline{\mathbf{A}}_{ui}(k)\underline{\mathbf{V}}_{ui}^t(k)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{ui}^t(k)\Big) + E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{ui}^t(k)\underline{\mathbf{A}}_{ui}^t(k)\Big)\\
&+ E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{ui}^t(k)Tr\big[\underline{\mathbf{V}}_{ui}^t(k)\underline{\mathbf{V}}_{ui}(k)\big]\Big)\Big]\\
&+ \sum_{i=1}^{Ne}\sum_{j=1}^{Ne}\mu_{b_i}\mu_{b_j}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta}\Big[E\Big(\underline{\mathbf{A}}_{ui}(k)1\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)1^t\underline{\mathbf{A}}_{uj}^t(m)\Big)\\
&+ E\Big(\underline{\mathbf{A}}_{ui}(k)1\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)1\underline{\mathbf{U}}_{uj}^t(m)\Big) + E\Big(\underline{\mathbf{U}}_{ui}(k)1^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)1^t\underline{\mathbf{A}}_{uj}^t(m)\Big)\\
&+ E\Big(\underline{\mathbf{U}}_{ui}(k)1^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)1\underline{\mathbf{U}}_{uj}^t(m)\Big)\Big] \qquad (3.133)
\end{aligned}
$$

Cette expression est simplifiée par rapport à l'expression (3.127). Dans la majorité des cas considérés, l'ensemble des bruits présents au sein du système sont des bruits de quantification. Alors, l'expression précédente (3.133) est utilisée pour estimer la puissance du bruit en sortie. Si les signaux d'entrée du système proviennent de la sortie d'un autre système, cette expression ne peut être utilisée. L'équation (3.127) est alors reprise pour les bruits d'entrée. Sinon, la puissance du bruit peut s'écrire sous la forme simplifiée suivante :

$$P_{b_y} = \sum_{i=1}^{Ne}\sigma_{b_i}^2 Ka_i + \sum_{i=1}^{Ne}\sum_{j=1}^{Ne}\mu_{b_i}\mu_{bj}Km_{ij} \qquad (3.134)$$

Elle s'exprime alors comme la somme de la puissance de chacun des bruits, les termes Ka et Km étant des constantes définissant le système.

Ces deux constantes correspondent à des espérances de termes de signaux. Lors de l'évaluation analytique de la précision, le temps requis le plus important est celui de la détermination de ces constantes. En effet, pour les établir, les termes statistiques de combinaisons de signaux sont calculés. Cette étape peut être longues pour des systèmes de taille élevée. Une fois cette étape franchie, ces constantes sont multipliées par les moyennes et variances des bruits de quantifications. Ces termes sont calculés à l'aide des modèles des bruits comme expliqué dans la partie 1.2. Cette seconde étape consiste à évaluer l'expression analytique de la puissance du bruit P_{b_y}. Le temps requis est alors très faible.

L'intercorrélation entre deux bruits de sortie s'exprime alors sous la forme suivante :

$$
\begin{aligned}
\mathbf{R}_{b_{yu}b_{yv}}(\theta) &= \sum_{i=1}^{Ne}\sigma_{b_i}^2\sum_{k=0}^{n-\theta}\Big[E\Big(Tr\big[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{vi}^t(k)\big]\underline{\mathbf{A}}_{ui}(k)\underline{\mathbf{A}}_{vi}^t(k)\Big)\\
&+ E\Big(\underline{\mathbf{A}}_{ui}(k)\underline{\mathbf{V}}_{vi}^t(k)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{vi}^t(k)\Big) + E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{vi}(k)\underline{\mathbf{V}}_{ui}^t(k)\underline{\mathbf{A}}_{vi}^t(k)\Big)\\
&+ E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{vi}^t(k)Tr\big[\underline{\mathbf{V}}_{vi}^t(k)\underline{\mathbf{V}}_{ui}(k)\big]\Big)\Big]\\
&+ \sum_{i=1}^{Ne}\sum_{j=1}^{Ne}\mu_{b_i}\mu_{b_j}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta}\Big[E\Big(\underline{\mathbf{A}}_{ui}(k)1\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{vj}^t(m)1^t\underline{\mathbf{A}}_{vj}^t(m)\Big)\\
&+ E\Big(\underline{\mathbf{A}}_{ui}(k)1\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{vj}^t(m)1\underline{\mathbf{U}}_{vj}^t(m)\Big) + E\Big(\underline{\mathbf{U}}_{ui}(k)1^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{vj}^t(m)1^t\underline{\mathbf{A}}_{vj}^t(m)\Big)\\
&+ E\Big(\underline{\mathbf{U}}_{ui}(k)1^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{vj}^t(m)1\underline{\mathbf{U}}_{vj}^t(m)\Big)\Big] \qquad (3.135)
\end{aligned}
$$

Elle s'exprime comme la produit des termes de moyenne. L'intercovariance entre deux bruits de sortie différents s'exprime selon la relation suivante :

$$\mathbf{C}_{b_{yu}b_{yv}}(\theta) = \mathbf{R}_{b_{yu}b_{yv}}(\theta) - m_{b_{yu}} m_{b_{yv}}^t \tag{3.136}$$

Dans ce paragraphe, les calculs des statistiques des bruits de sortie sont exprimés dans le cas des bruits blancs. Ce contexte représente la majorité des cas traités. Les expressions obtenues sont alors simplifiées car les statistiques des bruits sont mises en facteur.

Si les signaux sont complexes, alors les bruits peuvent l'être aussi. Dans ce cas, les matrices sont définies par l'expression suivante :

$$\mathbf{R}_{b_{yu}\overline{b}_{yu}}(\theta) \tag{3.137}$$

où $\overline{b_{yu}}$ désigne le conjugué de b_{yu}. Pour le reste, le modèle n'est pas modifié.

3.3.2 Calcul de la complexité

Les caractéristiques statistiques sont données par les expressions précédentes. Dans le but d'implanter ces calculs dans un outil, leur complexité doit être évaluée. La complexité est déterminée à partir du nombre de multiplications et d'additions à effectuer. Pour ce faire, les complexités dans le cas du calcul de la moyenne et de l'autocorrélation sont développées ci-dessous.

Cas de la moyenne

La moyenne est donnée par l'expression suivante :

$$m_{b_{yu}} = \sum_{i=1}^{Ne} \sum_{k=0}^{n} E(\underline{\mathbf{A}}_{ui}(k) m_{b_i} \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) m_{b_i}^t \underline{\mathbf{V}}_{ui}(k)) \tag{3.138}$$

Elle correspond au produit de six matrices. Pour rappel, le produit de deux matrices de taille PxQ et QxR nécessite PQR multiplications et $P(Q-1)R$ additions. Or, pour plus de clarté, le nombre d'additions est approximé par PQR comme pour les multiplications. L'objectif étant d'obtenir un ordre de complexité, cette hypothèse n'influe pas sur les calculs. De plus, pour simplifier les notations, $M_s = M_{y_u}$, $N_s = N_{y_u}$, $M_i = M_{b_i}$ et $N_i = N_{b_i}$. Les différents termes présents dans l'équation ont les tailles suivantes :

- $\underline{\mathbf{A}}_{ui}(k)$ a pour tailles M_sxM_i.
- m_{b_i} a pour dimensions M_ixN_i.
- $\underline{\mathbf{D}}_{ui}(k)$ a pour tailles N_ixN_s.

Donc le produit $\overline{\mathbf{A}_i(k)} m_{b_i}$ nécessite $M_s M_i N_i$ multiplications et produit une matrice de taille M_sxN_i. Le produit $\overline{\mathbf{A}_i(k)} m_{b_i}$ par $\underline{\mathbf{D}_i(k)}$ requiert $M_s N_i N_s$ multiplications donnant un nombre global de $M_s M_i N_i + \overline{M_s N_i} N_s$ multiplications. De même, $M_s M_i N_i + M_s M_i N_s$ multiplications sont nécessaires pour obtenir le terme $\underline{\mathbf{U}_i(k)} m_{b_i}^t \underline{\mathbf{V}_i(k)}$.

Le nombre de multiplications pour le calcul de la moyenne est égal à :

$$N_{mult} = 2M_s M_i N_i + M_s N_s (M_i + N_i) \tag{3.139}$$

Le nombre d'additions nécessaires est égal à la somme du nombre de multiplications précédent et de $M_s N_s$ liés à la somme des termes $\underline{\mathbf{A}}_{ui}(k) m_{b_i} \underline{\mathbf{D}}_{ui}(k)$ et $\underline{\mathbf{U}}_{ui}(k) m_{b_i}^t \underline{\mathbf{V}}_{ui}(k)$

$$N_{add} = 2M_s M_i N_i + M_s N_s (M_i + N_i) + M_s N_s \tag{3.140}$$

Or, le calcul de la moyenne est composé d'une double somme $\sum_{i=1}^{Ne}\sum_{k=0}^{n}$. La somme $\sum_{k=0}^{n}$ est a priori une somme infinie. En pratique, cela est résolu en arrêtant la somme après le calcul d'un nombre p_i d'échantillons. Ce nombre p_i est fixé de façon à s'assurer que les termes de bruit $b_i(n)$ non pris en compte dans le calcul soient négligeables par rapport aux termes pris en compte. En effet, les échantillons du bruit d'entrée à l'instant n prédominent en terme de puissance. Les échantillons précédents ont une puissance décroissante. Ceci est dû à la décroissance des matrices $\mathbf{A}$ et $\mathbf{D}$, assurant ainsi la stabilité du système. En pratique ce nombre p_i peut varier entre 50 et 1000 selon la vitesse de décroissance des termes à calculer. Plus les termes décroissent rapidement, moins le nombre de termes à prendre en compte est important. Le nombre de multiplications nécessaires au calcul de la moyenne pour un bruit d'entrée $b_i(n)$ est égal à :

$$p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i)\right) \tag{3.141}$$

Le nombre d'additions est alors de :

$$p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i) + M_sN_s\right) \tag{3.142}$$

Le nombre complet de multiplications est alors la somme sur l'ensemble des bruits d'entrée. Cela correspond à la somme sur i de l'expression (3.141).

$$\sum_i p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i)\right) \tag{3.143}$$

Le nombre complet d'additions est alors la somme sur i de l'expression (3.142)

$$\sum_i p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i) + M_sN_s\right) \tag{3.144}$$

Pour terminer, le nombre de termes pour le calcul de l'espérance doit être pris en compte. Soit T le nombre de termes de calcul d'espérance utilisé. En pratique $T = 100$ permet d'avoir un résultat correct. Le nombre complet d'additions et de multiplications est alors de :

$$T\sum_i p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i)\right) \tag{3.145}$$

$$T\sum_i p_i \left(2M_sM_iN_i + M_sN_s(M_i + N_i) + M_sN_s\right) \tag{3.146}$$

Pour rendre ces expressions un peu plus lisibles, un exemple est traité. Considérons le cas où les bruits d'entrée sont des vecteurs de taille N (donc $M_i = N$ et $N_i = 1$) et où les sorties sont des scalaires ($M_s = 1$ et $N_s = 1$). De plus, le nombre p_i de termes nécessaires au calcul est fixé à 500 et le nombre de termes pour l'espérance est $T = 100$. Le nombre de multiplications nécessaires est égal à :

$$T\sum_i p_i N_{mult} = 50000N_e(3N + 1) \tag{3.147}$$

où N_e désigne le nombre de sources de bruit. La complexité obtenue est alors en $\mathcal{O}(N)$.

Cas de l'autocorrélation

Dans cette section, la complexité de l'autocorrélation est calculée. L'autocorrélation s'exprime avec l'équation suivante :

$$
\begin{aligned}
\mathbf{R}_{b_{yu}b_{yu}}(\theta) &= E[b_{yu}(n)b_{yu}^t(n-\theta)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(Tr\big[(\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m))\big]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\Big) \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\Big[\big(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)+\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\big) \\
&.\big(\underline{\mathbf{A}}_{uj}(m)m_{b_j}\underline{\mathbf{D}}_{uj}(m)+\underline{\mathbf{U}}_{uj}(m)m_{b_j}^t\underline{\mathbf{V}}_{uj}(m)\big)^t\Big]
\end{aligned}
\tag{3.148}
$$

Cette équation est composée de cinq termes. La complexité de chacun de ces cinq termes est déterminée.

Complexité du premier terme Tout d'abord, le premier terme est composé de la trace du produit de D et D^t. Donc, pour déterminer la trace, les termes présents sur la diagonale de DD^t doivent être connus. Cela représente un nombre de N_i termes. Chacun de ces N_i termes est généré par la multiplication de N_s termes. Cette étape nécessite donc N_sN_i multiplications. Ensuite, le produit des matrices ACA^t est considéré. Pour le produit de ces trois matrices, $M_sM_i^2 + M_s^2M_i$ multiplications sont effectuées. Ceci conduit au nombre de multiplications suivant :

$$M_sM_i^2 + M_s^2M_i + N_sN_i \tag{3.149}$$

Puis, ces deux termes sont multipliés, d'où l'ajout de M_s^2 multiplications.

$$M_sM_i^2 + M_s^2M_i + N_sN_i + M_s^2 \tag{3.150}$$

Le calcul de la trace nécessite environ N_i additions, mais aucune addition n'intervient dans le produit de la trace par ACA^t. Le nombre d'additions nécessaires est de :

$$M_sM_i^2 + M_s^2M_i + N_sN_i + N_i \tag{3.151}$$

Complexité du deuxième terme En analysant le deuxième terme comme le premier, cela conduit à un nombre de multiplications de :

$$M_sM_i^2 + M_sM_iN_s + M_sN_sN_i + M_s^2N_i \tag{3.152}$$

Le nombre d'additions est de :

$$M_sM_i^2 + M_sM_iN_s + M_sN_sN_i + M_s^2N_i \tag{3.153}$$

Complexité du troisième terme Le nombre de multiplication pour le troisième terme est égal à :

$$M_s M_i^2 + M_s M_i N_s + M_s N_s N_i + M_s^2 M_i \tag{3.154}$$

et le nombre d'additions est de :

$$M_s M_i^2 + M_s M_i N_s + M_s N_s N_i + M_s^2 M_i \tag{3.155}$$

Complexité du quatrième terme Pour le quatrième terme, cela conduit à un nombre de multiplications de :

$$N_s M_i^2 + N_s M_i + M_s^2 + M_s^2 N_i \tag{3.156}$$

et un nombre d'additions égal à :

$$N_s M_i^2 + N_s M_i + N_s + M_s^2 N_i \tag{3.157}$$

Complexité du cinquième terme Pour le cinquième terme, le nombre de multiplications a pour valeur :

$$2(2M_s M_i N_i + N_s N_i M_s + M_s M_i N_s) + M_s^2 N_s \tag{3.158}$$

et le nombre d'additions est égal à :

$$2(2M_s M_i N_i + M_s N_s (M_i + N_i) + M_s N_s) + M_s^2 N_s \tag{3.159}$$

Complexité globale La complexité de l'autocorrélation est la somme de la complexité prise pour chacun des cinq termes. Comme dans le cas de la moyenne, p_i désigne le nombre de termes du bruit $b_i(n)$ pris en compte dans le calcul et T correspond au nombre de termes de calcul de la moyenne. Le nombre complet de multiplications est égal à :

$$T \sum_i p_i \Big((M_i + N_i)(4M_s N_s + 2M_s^2 + N_s + 3M_s M_i) + M_i(N_i M_s + M_i N_s) + M_s^2 (2 + N_s) \Big) \tag{3.160}$$

Pour simplifier la compréhension, le cas où le bruit $b_i(n)$ est un vecteur de taille N et, où la sortie est un scalaire, est repris. Le nombre de multiplication devient :

$$T \sum_i p_i N_{mult} = N_e 50000(4N^2 + 11N + 10) \tag{3.161}$$

Dans ce cas, le nombre d'additions est du même ordre :

$$T \sum_i p_i N_{add} = N_e 50000(4N^2 + 11N + 10) \tag{3.162}$$

Ainsi la complexité est donc en $\mathcal{O}(N^2)$. Dans le cas où l'entrée devient une matrice et la sortie un vecteur, la complexité est en $\mathcal{O}(N^3)$. La complexité est du même ordre pour les autres termes (covariance et intercorrélation). Ces calculs sont faits dans le pire cas. Pour les systèmes non-récursifs, l'ordre de complexité est moins important car le terme de somme disparaît.

3.3.3 Méthode de réduction de la complexité

Dans cette section, une méthode de réduction de la complexité au moyen d'une approche par prédiction linéaire est présentée. En effet, les termes de signaux **A**, **D**,**U** et **V** sont présents dans les expressions des statistiques des bruits de sortie sous la forme d'une somme infinie (équations (3.127) et (3.133)). L'objectif de l'approche est d'exprimer ces sommes infinies d'une manière différente permettant de réduire la complexité des calculs. Dans un premier temps, la prédiction linéaire est présentée. Ensuite, la méthode est définie pour notre modèle. Finalement, elle est appliquée aux sommes infinies pour montrer les simplifications obtenues.

Méthode de prédiction linéaire

Dans la partie précédente, les calculs de complexité du modèle ont été présentés. Dans un souci d'optimisation, réduire cette complexité permettrait d'obtenir des gains importants en temps de calcul. L'idée est de simplifier le développement de la récursion en approchant la somme infinie par une méthode basée sur la prédiction linéaire. La prédiction linéaire se définit de la manière suivante. Elle permet d'approximer les termes d'une séquence au moyen d'une fonction linéaire appliquée aux P précédents termes. Les premiers termes étant $h(n)$,$h(n-1)$..., le terme $h(i)$ est estimé à partir des P précédents termes $h(i+j)$. Soit $\lambda_1 ... \lambda_P$ les P coefficients de prédiction linéaire. La valeur estimée $\widehat{h}(i)$ de $h(i)$ est donnée par l'expression suivante :

$$\widehat{h}(i) = \sum_{j=1}^{P} \lambda_j h(i+j) \tag{3.163}$$

L'objectif est d'approcher les termes de la somme infinie à partir des premiers échantillons de cette dernière et des coefficients de prédiction calculés. Pour ce faire, le nombre de coefficients P et leur valeur doivent être déterminés.

Présentation de la méthode

La méthode de prédiction linéaire définie pour notre modèle est présentée dans ce paragraphe. Cette méthode est appliquée aux termes de pseudo réponse impulsionnelle permettant de décrire le système. En pratique, l'approche est utilisée sur les termes de signaux **A** et **D**. Les termes de pseudo réponse impulsionnelle se déduisent les uns des autres d'après la formule explicitée en section 3.2.1. L'objectif est de modéliser ces relations de façon linéaire. Ainsi, les sommes infinies peuvent ensuite être modélisées par ces coefficients. Cependant, cette approximation génère un biais entre la valeur obtenue et la valeur réelle. Ce biais ne diverge pas comme expliqué par la suite, et son influence sur la qualité des calculs est mesuré dans la dernière partie de ce chapitre. Le premier élément à déterminer pour appliquer cette méthode est l'ordre de prédiction P. Or un système récursif peut se modéliser sous la forme d'une pseudo-fonction de transfert

$$H^{(n)}(z) = \frac{\sum_{l=0}^{Q} g_{(n-l)}^{(n)} z^{-l}}{1 - \sum_{i=1}^{L} f_{(n-i)}^{(n)} z^{-i}} \tag{3.164}$$

Le calcul de la pseudo réponse impulsionnelle, présenté dans la partie 3.2.1, est obtenu en déterminant les termes en fonction des L précédents termes calculés. En effet, l'ordre L d'un système récursif définit le nombre de sorties précédentes dont est fonction la sortie actuelle. De ce fait, l'ordre de prédiction P représentant le nombre de coefficients de prédiction à déterminer, est égal à l'ordre L du système récursif.

$$P = L \tag{3.165}$$

Ensuite, les coefficients doivent être déterminés. Ces coefficients sont des scalaires de valeurs constantes. La principale difficulté est de définir une méthode de calcul de ceux-ci. Ils doivent minimiser l'erreur entre la valeur réelle $h(i)$ et la valeur estimée $\widehat{h}(i)$. Ces termes sont composés de scalaires, vecteurs ou matrices. Ainsi, dans le cas général, ces termes sont considérés comme des matrices. Le but est donc de minimiser cette erreur quadratique définie par :

$$E(e^2) = E\Big(Tr\Big(\big(h(i) - \widehat{h}(i)\big)\big(h^t(i) - \widehat{h}^t(i)\big)\Big)\Big) \tag{3.166}$$

L'opérateur $Trace$ permet d'obtenir des scalaires. Les P coefficients de prédiction sont notés $\lambda_1....\lambda_P$. La dérivée de cette erreur quadratique en fonction du coefficient λ_k donne

$$\frac{\partial E(e^2)}{\partial \lambda_k} = Tr\Big(E\big[2h(i)h^t(i+k) - 2\sum_j \lambda_j h(i+j)h^t(i+k)\big]\Big) \ \forall k \in [1:P] \tag{3.167}$$

L'erreur quadratique est minimale si sa dérivée est nulle donc si

$$Tr\Big(E[h(i)h^t(i+k)]\Big) = \sum_j \lambda_j Tr\Big(E[h(i+j)h^t(i+k)]\Big) \ \forall k \in [1:P] \tag{3.168}$$

De ce fait, cette expression peut être représentée sous forme matricielle. La matrice $\mathbf{S}$ de dimension PxP est introduite. L'élément de sa $m^{\text{ème}}$ ligne et de sa $n^{\text{ème}}$ colonne est donné par la relation suivante :

$$\mathbf{S}_{(m,n)} = Tr\Big(E[h(i+m)h^t(i+n)]\Big) \tag{3.169}$$

Un vecteur ligne J de taille P peut également être introduit. Son $n^{\text{ème}}$ terme est défini par l'expression suivante

$$J_{(n)} = Tr\Big(E[h(i)h^t(i+n)]\Big) \tag{3.170}$$

Alors, le vecteur $\underline{\lambda} = [\lambda_1...\lambda_P]^t$ des coefficients de prediction linéaire vérifie l'égalité suivante :

$$\underline{\lambda}^t \mathbf{S} = J \tag{3.171}$$

Pour déterminer les coefficients λ, la matrice $\mathbf{S}$ doit être inversée. Pour cela, cette matrice doit être inversible. Si ce n'est pas le cas, les coefficients ne peuvent être déterminés de cette façon. En effet, dans le cas où les fonctions $h(i)$ sont des constantes, les lignes de la matrices $\mathbf{S}$ sont liées deux à deux. Pour comprendre, traitons le cas d'un IIR 2 de fonction de transfert

$$H(Z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3.172}$$

L'ordre de récursion de ce système est $P = 2$. Deux coefficients λ_1 et λ_2 sont à déterminer. Les premiers termes de la réponse impulsionnelle sont les suivants

$$\begin{aligned} h(n) &= b_0 \\ h(n-1) &= a_1 b_0 \\ h(n-2) &= a_1^2 b_0 + a_2 b_0 \end{aligned} \tag{3.173}$$

La valeur exacte du troisième terme de réponse impulsionnelle est $h(n-2) = a_1^2 b_0 + a_2 b_0$. Cherchons à calculer la valeur des coefficients permettant d'approcher au mieux cette valeur. Pour ce faire, la matrice $\mathbf{S}$ est définie. Les termes h sont des constantes scalaires.

$$\mathbf{S} = \begin{pmatrix} h(n-1)^2 & h(n-1)h(n) \\ h(n-1)h(n) & h(n)^2 \end{pmatrix} \tag{3.174}$$

En remplaçant les termes h par leur valeur, cette matrice est égale à :

$$\mathbf{S} = \begin{pmatrix} a_1^2 b_0^2 & a_1 b_0^2 \\ a_1 b_0^2 & b_0^2 \end{pmatrix} \tag{3.175}$$

La première colonne de cette matrice se déduit de la seconde en la multipliant par a_1. Ces deux colonnes étant liées, la matrice n'est pas inversible. Pour pouvoir appliquer la méthode dans l'ensemble des cas traités, cette matrice doit être modifiée. Les colonnes de la matrice étant liées entre elles, une seule doit être conservée. Par choix, la colonne la plus à droite est conservée. Elle regroupe les termes les plus anciens. Cette colonne permet d'obtenir une relation linéaire liant les différents coefficients λ. De la même façon, le vecteur J doit être modifié. Seul le dernier élément $Tr\left(E[h(i)h(i+P)^t]\right)$ de ce vecteur est conservé. Cependant, une infinité de solution est possible. Pour restreindre le choix à une solution unique, des contraintes doivent être fixées. Reprenons l'exemple du filtre IIR pour trouver ces contraintes. Les coefficients de prédiction correspondent aux coefficients du filtre. En effet, les termes de réponse impulsionnelle se déduisent les uns des autres en multipliant les termes précédents par les coefficients a_1 et a_2, comme expliqué dans la partie 3.2.1. De ce fait, les coefficients de prédiction à obtenir sont égaux à :

$$\begin{aligned} \lambda_1 &= a_1 \\ \lambda_2 &= a_2 \end{aligned} \tag{3.176}$$

La contrainte linéaire obtenue par la dernière colonne de la matrice $\mathbf{S}$ pour ces coefficients est la suivante :

$$\lambda_1 a_1 b_0^2 + \lambda_2 b_0^2 = (a_1^2 + a_2) b_0^2 \tag{3.177}$$

Pour obtenir la valeur correcte des coefficients des prédiction, la contrainte suivante est fixée :

$$\lambda_1 + \lambda_2 = a_1 + a_2 \tag{3.178}$$

Cette expression doit être réécrite pour l'utilisation de la forme générale des pseudo fonctions de transfert. Les termes a_1 et a_2 correspondent aux termes du dénominateur de la fonction de transfert du IIR, donc aux termes $f^{(n)}$ de la pseudo fonction de transfert générale. Ces termes $f^{(n)}$ peuvent être de dimension M quelconque. Pour obtenir un scalaire, la moyenne des termes le composant est pris en compte. La moyenne des termes est notée $\|f\|$ comme l'exprime la relation suivante :

$$\|f^{(n)}_{(n-1)}\| = \frac{\sum_{i,j}^{M} f^{(n)}_{(n-1)_{i,j}}}{M^2} \tag{3.179}$$

où M désigne la taille de cette matrice. Avec cette notation, la contrainte définie pour l'IIR s'écrit sous la forme suivante :

$$\lambda_1 + \lambda_2 = E\left[\|f^{(n)}_{(n-1)}\| + \|f^{(n)}_{(n-2)}\|\right] \tag{3.180}$$

Pour déterminer de manière unique ces P coefficients, P relations doivent être établies entre eux. La première relation désigne l'égalité linéaire provenant de la colonne conservée de la matrice $\mathbf{S}$. Les $P-1$ autres contraintes sont généralisées depuis l'expression précédente.

$$\lambda_i + \lambda_{i+1} = E\left[\|f^{(n)}_{(n-i)}\| + \|f^{(n)}_{(n-i-1)}\|\right] \ \forall i \in [1 : P-1] \tag{3.181}$$

De manière plus générale, les contraintes permettent de normaliser les coefficients de prédiction par rapport aux coefficients du système. La nouvelle matrice $\mathbf{S}$ ainsi obtenue est la suivante :

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & Tr\Big(E[h(i+1)h^t(i+P)]\Big) \\ 1 & 1 & 0 & \dots & 0 & Tr\Big(E[h(i+2)h^t(i+P)]\Big) \\ \vdots & & & & & \\ 0 & \dots & 1 & 1 & 0 & Tr\Big(E[h(i+P-2)h^t(i+P)]\Big) \\ 0 & \dots & 0 & 1 & 1 & Tr\Big(E[h(i+P-1)h^t(i+P)]\Big) \\ 0 & \dots & 0 & 0 & 1 & Tr\Big(E[h(i+P)h^t(i+P)]\Big) \end{pmatrix} \tag{3.182}$$

Le vecteur J est défini par l'expression suivante :

$$J = \begin{pmatrix} E\left[\|f^{(n)}_{(n-1)}\| + \|f^{(n)}_{(n-2)}\|\right] \\ E\left[\|f^{(n)}_{(n-2)}\| + \|f^{(n)}_{(n-3)}\|\right] \\ \dots \\ E\left[\|f^{(n)}_{(n-P+1)}\| + \|f^{(n)}_{(n-P)}\|\right] \\ Tr\Big(E[h(i)h^t(i+P)]\Big) \end{pmatrix}^t \tag{3.183}$$

Ainsi les coefficients de prediction peuvent être déterminés en utilisant ces deux matrices. Cette méthode peut alors être utilisée pour prédire les termes de signaux. Néanmoins, pour le calcul, les premiers termes de la réponse impulsionnelle sont déterminés. Ces termes corespondent aux $h(i+1)...h(i+P)$ nécessaires à la détermination de la matrice $\mathbf{S}$. Le nombre de termes nécessaires au calcul de ces coefficients doit être fixé. Si Q désigne l'ordre du numérateur de la pseudo fonction de transfert, alors les $P+Q$ premiers termes de réponses impulsionnelles sont calculés. La nécessité de calculer les $P+Q$ premiers termes permet de prendre en compte l'ensemble des termes contenus dans la pseudo fonction de transfert pour calculer les coefficients de prédiction linéaire et, ainsi, d'avoir un modèle encore plus réaliste car tous les termes interviennent. Par la suite, les termes de prédiction linéaire sont établis. A partir de là, le système est parfaitement modélisé. Pour appliquer cette méthode, la démarche suivante est proposée et résumée à la figure 3.21. Les termes P et Q désignent l'ordre du numérateur et du dénominateur de la pseudo fonction de transfert du bruit considéré.

- Les $P+Q$ premiers termes de réponses impulsionnelles sont calculés par la méthode présentée dans la partie 3.2.1.

- La matrice $\mathbf{S}$ et le vecteur J sont alors établis et ainsi, les coefficients de prédiction λ sont calculés $\lambda = J\mathbf{S^{-1}}$.

- Uniquement les coefficients λ et les $P+Q-1$ premiers termes de réponse impulsionnelle sont conservés en mémoire. Ainsi pour appliquer la méthode, seuls les ordres du numérateur Q, du dénominateur P, et les P coefficients de prédiction linéaire sont nécessaires. Le gain de place en mémoire est alors très important.

Le terme $P+Q-1$ (somme des ordres du numérateur et du dénominateur moins 1) s'appelle ordre initial et le terme P (ordre du dénominateur) définissant le nombre de coefficients est dénommé ordre de prédiction. Cependant, pour un système non-récursif, le dénominateur de la fonction de transfert est d'ordre nul. Dans ce cas, un coefficient de prédiction est calculé de valeur nulle car aucun terme n'est à prédire.

Le modèle a été présenté pour les fonctions h. Or celles-ci correspondent à la multiplication par deux termes de signaux $\mathbf{A}$ et $\mathbf{D}$. En pratique, Deux séquences de termes de prédiction sont

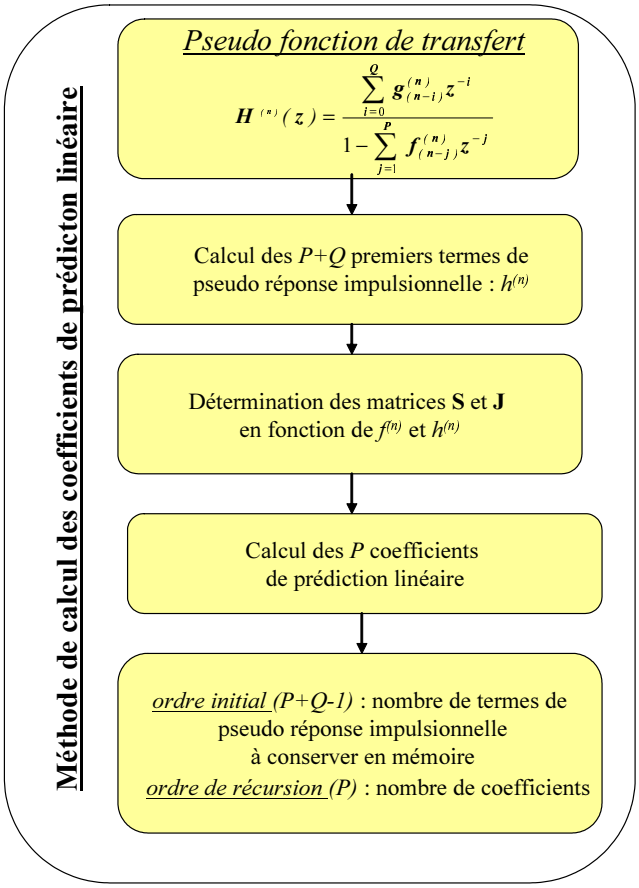

FIG. 3.21 – *Méthode de calcul des coefficients de prédiction linéaire*

alors calculées, une pour les termes **A** et une pour les termes **D**. Pour illustrer la méthode, celle-ci est appliquée sur l'exemple de l'IIR traité plus haut.

La fonction de transfert est la suivante :

$$H(z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3.184}$$

Avant de déterminer les coefficients de prédiction linéaire, il convient d'établir la valeur des premiers termes de la réponse impulsionnelle. Si Q correspond au nombre de termes du numérateur de la fonction de transfert, le nombre de termes de réponse impulsionnelle à établir est $P + Q$. Ici $P = 2$ et $Q = 1$, donc les 3 premiers termes de réponse impulsionnelle sont définis.

$$\begin{aligned} h(n) &= b_0 \\ h(n-1) &= a_1 b_0 \\ h(n-2) &= a_1^2 b_0 + a_2 b_0 \end{aligned} \tag{3.185}$$

Les termes étant des constantes, la trace de l'espérance de ces termes correspond aux termes eux-mêmes. La matrice **S** est définie par l'expression suivante

$$\mathbf{S} = \begin{pmatrix} 1 & h(n-1)h(n) \\ 1 & h(n)^2 \end{pmatrix} \tag{3.186}$$

En remplaçant par leur valeur, la matrice **S** est égale à

$$\mathbf{S} = \begin{pmatrix} 1 & a_1 b_0^2 \\ 1 & b_0^2 \end{pmatrix} \tag{3.187}$$

Le vecteur J est défini par l'équation suivante :

$$J = \left(\begin{array}{cc} E\left[\|f_{(n-1)}^{(n)}\| + \|f_{(n-2)}^{(n)}\|\right] & Tr\left(E[h(n-2)h^t(n)]\right) \end{array} \right) \tag{3.188}$$

En introduisant la valeur de chaque terme, le vecteur J est égal à :

$$J = \left(\begin{array}{cc} a_1 + a_2 & a_1^2 b_0^2 + a_2 b_0^2 \end{array} \right) \tag{3.189}$$

La multiplication de J par l'inverse de $\mathbf{S}$ fournit comme résultats les valeurs a_1 et a_2. Les coefficients du filtre sont bien retrouvés.

Application de la méthode

L'application de cette méthode dans le calcul de la puissance du bruit de sortie est explicitée ci-dessous. L'expression définissant la puissance du bruit comporte des sommes, en théorie infinies ($\sum_{k=0}^{n} \mathbf{A}(\mathbf{k})S(b)\mathbf{D}(\mathbf{k})$) où S désigne une statistique du bruit b. De la même façon, les termes $\mathbf{U}$ et $\mathbf{V}$ sont traités en considérant les termes transposés. Pour la présentation de l'application de la méthode, seuls les termes non transposés sont pris en compte. D'autre part, dans la plupart des cas traités, seuls un des termes $\mathbf{A}$ ou $\mathbf{D}$ apparaît. Ainsi, dans un premier temps, la cas où seul un des deux termes apparaît est considéré. Dans un second temps, la méthode est généralisée à la présence des deux termes.

Si un seul terme de bruit apparaît (par exemple $\mathbf{A}$), la somme s'écrit sous la forme suivante :

$$\sum_{k=0}^{n} \mathbf{A}(k)S(b) \tag{3.190}$$

L'objectif est d'exprimer de manière différente le terme $\sum_{k=0}^{n} \mathbf{A}(k)$. Les termes $\mathbf{A}(k)$ et les $\mathbf{D}(k)$ définissent la pseudo réponse impulsionnelle du système parcouru par le bruit d'entrée. Ces termes sont calculés en développant la pseudo fonction de transfert du système. Cette pseudo fonction de transfert s'écrit sous la forme suivante :

$$H^{(n)}(z) = \frac{\sum_{l=0}^{Q} g_{(n-l)}^{(n)} z^{-l}}{1 - \sum_{i=1}^{P} f_{(n-i)}^{(n)} z^{-i}} \tag{3.191}$$

Avec la méthode explicitée précédemment, les $P+Q-1$ premiers termes de la pseudo réponse impulsionnelle sont déterminés directement. Ainsi, les termes $\mathbf{A}(n)$.....$\mathbf{A}(n-P-Q+1)$ sont calculés par la méthode classique de détermination de la pseudo réponse impulsionnelle. La matrice $\mathbf{\Omega}_0$ est introduite et définie par la relation suivante :

$$\mathbf{\Omega}_0 = \begin{pmatrix} \mathbf{A}(n-Q+1) \\ \mathbf{A}(n-Q) \\ \vdots \\ \mathbf{A}(n-P-Q+2) \end{pmatrix} \tag{3.192}$$

D'après l'expression précédente, $\mathbf{\Omega}_0$ semble être un vecteur. Or les termes $\mathbf{A}(n-Q)$ la composant sont en général des matrices. De ce fait, $\mathbf{\Omega}_0$ est considérée comme une matrice. Cette dernière contient les P termes de pseudo réponse impulsionnelle à partir de l'échantillons $n-Q$. De la même façon, la matrice $\mathbf{\Omega}_m$ est introduite et définie par la relation suivante :

$$\mathbf{\Omega}_m = \begin{pmatrix} \mathbf{A}(n-Q-m+1) \\ \mathbf{A}(n-Q-m) \\ \vdots \\ \mathbf{A}(n-P-Q-m+2) \end{pmatrix} \tag{3.193}$$

Or, les coefficients de prédiction permettent de calculer les termes suivants de pseudo réponse impulsionnelle. La méthode de calcul des coefficients s'applique à partir de l'échantillon $P+Q$ donc à partir de l'instant $n-P-Q+1$. L'échantillon de l'instant $n-P-Q+1$ est présent dans la matrice $\mathbf{\Omega}_1$, puis dans les $P-1$ suivantes. Comme l'échantillon à l'instant $n-P-Q+1$ peut être calculé par les coefficients de prédiction, la matrice $\mathbf{\Omega}_1$ peut également être déterminée à partir de ces coefficients. Ainsi, toutes les matrices $\mathbf{\Omega}_m$ pour m non nuls peuvent être déterminée à l'aide des coefficients de prédiction. Soit $\mathbf{\Lambda}$ la matrice carrée de taille P définie par la relation suivante :

$$\mathbf{\Lambda} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \dots & \vdots \\ 0 & \dots & \dots & 0 & 1 \\ \lambda_P & \lambda_{P-1} & \dots & \dots & \lambda_1 \end{pmatrix} \tag{3.194}$$

où les λ_i désignent les termes de prédiction des $\mathbf{A}$. Avec l'aide de cette matrice $\mathbf{\Lambda}$ la relation de récurrence suivante est explicitée :

$$\begin{aligned} \mathbf{\Omega}_{m+1} &= \mathbf{\Lambda}\mathbf{\Omega}_m \\ &= \mathbf{\Lambda}^{m+1}\mathbf{\Omega}_0 \end{aligned} \tag{3.195}$$

La matrice limite $\mathbf{\Lambda}^m$ possède une limite nulle à l'infini. La somme des matrices $\mathbf{\Omega_m}$ peut s'exprimer sous la forme suivante :

$$\begin{aligned} \sum_{m=0}^{\infty} \mathbf{\Omega}_m &= \sum_{m=0}^{\infty} \mathbf{\Lambda}^m \mathbf{\Omega}_0 \\ &= \left(\mathbf{I_P} - \mathbf{\Lambda}\right)^{-1} \mathbf{\Omega}_0 \end{aligned} \tag{3.196}$$

D'autre part, la première ligne de la somme des matrices $\mathbf{\Omega}_m$ est égale à :

$$\mathbf{A}(n-Q+1) + \mathbf{A}(n-Q) + ... + \mathbf{A}(0) = \sum_{k=0}^{n-Q+1} \mathbf{A}(k) \tag{3.197}$$

Cette expression représente une partie de la somme infinie à déterminer. Seuls les $Q-1$ premiers termes de la pseudo réponse impulsionnelles sont absents pour avoir l'égalité. Or ces $Q-1$ premiers termes sont calculés par la méthode classiques pour déterminer les coefficients de prédiction λ. Pour aboutir à l'expression de la somme infinie, ces termes sont ajoutés à la première ligne de la somme des matrices $\mathbf{\Omega}_m$. La valeur de la matrice $\left(\mathbf{I_P} - \mathbf{\Lambda}\right)^{-1}$ présente dans l'équation (3.196) est calculée. Celle-ci est égale à :

$$\left(\mathbf{I_P} - \mathbf{\Lambda}\right)^{-1} = \frac{1}{\sum_{i=1}^{P} \lambda_i - 1} \begin{pmatrix} \sum_{i=1}^{P-1} \lambda_i - 1 & \sum_{i=1}^{P-2} \lambda_i - 1 & \dots & \lambda_1 - 1 & -1 \\ -\lambda_P & \sum_{i=1}^{P-2} \lambda_i - 1 & \dots & \lambda_1 - 1 & -1 \\ -\lambda_P & -\lambda_P - \lambda_{P-1} & \dots & \lambda_1 - 1 & -1 \\ \vdots & \ddots & \ddots & \dots & \vdots \\ -\lambda_P & -\lambda_P - \lambda_{P-1} & -\lambda_P - \lambda_{P-1} - \lambda_{P-2} & \dots & -1 \end{pmatrix} \tag{3.198}$$

Or dans cette matrice, seule la première ligne a un intérêt pour le calcul de la somme. En effet, la somme à calculer est égale au produit de cette première ligne et de la matrice $\mathbf{\Omega_0}$. Cette somme $\sum_{k=0}^{n-Q+1} \mathbf{A}(k)$ est donc égale à :

$$\sum_{k=0}^{n-Q+1} \mathbf{A}(k) = \frac{1}{\sum_{i=1}^{P} \lambda_i - 1} \sum_{j=0}^{P-1} \left(\sum_{i=1}^{P-j-1} \lambda_i - 1 \right) \mathbf{A}(n-Q+1-j) \tag{3.199}$$

La somme infinie $\sum_{k=0}^{n} \mathbf{A}(\mathbf{k})$ est obtenue en ajoutant les $Q-1$ premiers termes de réponse impulsionnelle calculés précédemment. Cette somme infinie peut s'exprimer comme une fonction Φ des premiers termes calculés et des coefficients de prédiction.

$$\begin{aligned} \sum_{k=0}^{n} \mathbf{A}(k) &= \sum_{i=0}^{Q-2} \mathbf{A}(n-i) + \frac{1}{\sum_{i=1}^{P} \lambda_i - 1} \sum_{j=0}^{P-1} \left(\sum_{i=1}^{P-j-1} \lambda_i - 1 \right) \mathbf{A}(n-Q+1-j) \\ &= \Phi(\mathbf{A},\lambda) \end{aligned} \tag{3.200}$$

L'expression précédente montre l'intérêt des coefficients de prédiction. La somme infinie à calculer est transformée. Lorsque l'ordre de récursion P est égal à 1, un seul coefficient λ est considéré. L'expression est alors simplifiée en une série géométrique immédiate :

$$\sum_{k=0}^{n} \mathbf{A}(k) = \sum_{i=0}^{Q-2} \mathbf{A}(n-i) + \frac{1}{1-\lambda} \mathbf{A}(n-Q+1) \tag{3.201}$$

Cette méthode permet de réduire la complexité des calculs effectués.

Dans un deuxième temps, le cas où les termes $\mathbf{A}$ et $\mathbf{D}$ sont présents est considéré. L'objectif est de modéliser la somme suivante :

$$\sum_{k=0}^{n} \mathbf{A}(k) S(b) \mathbf{D}(k) \tag{3.202}$$

Maintenant, l'objectif est de trouver les valeurs λ des coefficients de prédiction correspondant à cette somme. Précédemment, les sommes $\sum_{k=0}^{n} \mathbf{A}(k)$ et $\sum_{k=0}^{n} \mathbf{D}(k)$ ont été modélisées. En fait, les termes λ correspondent au produit de λ_A et λ_D. La valeur ainsi obtenue d'un coefficient λ_i pour $i \in [1:P]$ est déterminée par l'expression suivante :

$$\lambda_i = \lambda_{A_i} \lambda_{D_i} \tag{3.203}$$

Les puissances des sommes infinies sont calculées par la modélisation précédente au moyen des coefficients de prédiction. Ainsi, les coefficients λ sont déterminés. La valeur de la somme infinie est alors déterminée par l'expression suivante :

$$\sum_{k=0}^{n} \mathbf{A}(k) S(b) \mathbf{D}(k) = \Phi(\mathbf{A} S(b) \mathbf{D}, \lambda) \tag{3.204}$$

L'équation précédente permet de modéliser la somme infinie dans les calculs des statistiques des bruits de sortie. Les prédictions étant effectuées sur les termes de pseudo réponse impulsionnelle, l'approximation ne diverge pas. En effet, l'approximation linéaire fournit une valeur estimée de la somme infinie. Cette valeur est réelle car les coefficients de prédiction sont inférieurs à 1. D'autre part, la stabilité du système est assurée par le fait que la somme des termes de pseudo réponse

impulsionnelle est réelle et ne diverge pas. Ainsi le biais généré par cette approche n'entraine pas de dégénérescence. Néanmoins, pour mesurer les valeurs de ce biais, différentes expérimentations sont effectués dans la partie 3.5. Elles permettent de s'assurer que le biais ne dégrade pas de manière importante la qualité des calculs. Ainsi, l'approche par coefficients de prédiction permet de réduire la complexité des calculs. Dans la partie suivante, cette complexité est évaluée pour la nouvelle approche.

Calcul de la complexité avec l'approche par prédiction linéaire

La méthode par prédiction linéaire est introduite pour réduire la complexité des expressions traitées. Dans ce paragraphe, les complexités des calculs de la moyenne et de l'autocorrélation sont évaluées.

Calcul de la complexité de la moyenne avec l'approche par prédiction linéaire Lae calcul de la complexité de la moyenne des bruits de sortie est présenté ci-dessous. Soit $b_i(n)$ le bruit d'entrée, P_i et Q_i les ordres du numérateur et du dénominateur de sa pseudo fonction de transfert. Seuls les $P_i + Q_i - 1$ premiers termes de $\mathbf{A}$,$\mathbf{D}$,$\mathbf{U}$ et $\mathbf{V}$ sont déterminés par la méthode de calcul de la pseudo réponse impulsionnelle. La complexité de ces termes a été calculé lors du calcul de la complexité de la méthode générale. L'expression est établie à partir de ces termes multipliés par des sommes de coefficients de prédiction.

L'équation (3.200) permet d'établir la complexité introduite par les coefficients de prédiction. Considérons les termes $\mathbf{A}$. Pour ces termes, le calcul des coefficients de prédiction nécessite $Mult_{P_i}$ multiplications et Add_{P_i} additions. Ensuite, chacun des termes $\mathbf{A}(n - Q_i + 1 - j)$ pour $j \in [0 : P_i - 1]$ est multiplié par une somme de ces coefficients (equation (3.200)). Ces multiplications requièrent $M_s M_i$ produits pour chaque terme $\mathbf{A}(n - Q_i + 1 - j)$, soit un total de $P_i M_s M_i$. Ces multiplications sont également effectuées pour les termes $\mathbf{D}$, $\mathbf{U}$ et $\mathbf{V}$. Le nombre de multiplications supplémentaires est égal à :

$$P_i(M_s + N_s)(M_i + N_i) \tag{3.205}$$

De plus, les sommes des coefficients de prédiction $\sum_i \lambda_i - 1$ nécessitent P_i additions pour les termes $\mathbf{A}$, et autant pour les autres termes.

Le nombre complet de multiplications N_{mult} est alors donné par l'expression suivante :

$$\begin{aligned} N_{mult} &= \sum_i T\Big[(P_i + Q_i - 1)\big(2M_s M_i N_i + M_s N_s(M_i + N_i)\big)\Big] \\ &+ P_i(M_s + N_s)(M_i + N_i) + 4Mult_{P_i} \end{aligned} \tag{3.206}$$

Le nombre complet d'additions est égal à :

$$\begin{aligned} N_{mult} &= \sum_i T\Big[(P_i + Q_i - 1)\big(2M_s M_i N_i + M_s N_s(M_i + N_i)\big)\Big] \\ &+ 4P_i + 4Add_{P_i} \end{aligned} \tag{3.207}$$

Par rapport aux expressions développées pour l'approche générale, le nombre de points de somme p_i disparaît. D'autre part, l'espérance est prise en compte uniquement sur les $P_i + Q_i - 1$ premiers termes. Pour rendre ces expressions un peu plus parlantes, un exemple est traité. L'exemple du vecteur d'entrée de taille N est repris. Les nombres Q_i et P_i sont fixés à 1. Dans ce cas, une série géométrique apparaît. Le nombre de multiplications nécessaires est de :

$$N_{mult} = 100(3N + 1)N_e + (2N + 2)N_e + 4Mult_1 \tag{3.208}$$

Le terme $Mult_1$ désigne le nombre de multiplications nécessaires au calcul du coefficient de prédiction. Dans cet exemple, $(2N+1)$ multiplications sont effectuées afin de déterminer la valeur du coefficient de chaque terme de bruit.

La complexité en termes de multiplications est alors égale à :

$$N_{mult} = 100(3N+1)N_e + N_e(4N+3) \tag{3.209}$$

La complexité semble alors réduite par 500 car la somme $\sum_{k=0}^{n}$ disparaît. Néanmoins le calcul du coefficient réduit légèrement le gain obtenu. Ce dernier demeure cependant très important.

Calcul de la complexité de l'autocorrélation avec l'approche par prédiction linéaire

Comme pour la moyenne, la complexité est réduite d'un facteur 500 correspondant au nombre de points utilisés pour calculer la somme par rapport à la méthode de calcul direct de cette somme. Néanmoins, le temps de calcul des coefficients pour réduire ce gain. Les temps requis sont explicités dans le chapitre 4.

Conclusion

La méthode de prédiction linéaire est une méthode permettant de réduire la complexité des calculs. De plus, le gain mémoire est important dans l'ensemble des cas car la mémorisation de la pseudo réponse impulsionnelle n'est plus nécessaire jusqu'à un ordre p. Seuls les premiers termes et les coefficients de prédiction sont mémorisés. De plus, le gain en complexité est légèrement inférieur à p correspondant au nombre de points de somme utilisés dans l'approche générale. En pratique, la valeur de p est fixée à 500. Ce gain n'est jamais atteint car le calcul des coefficients de prédiction réduit ce dernier. Les gains obtenus sont compris entre 100 et 300.

3.3.4 Exemples

Pour illustrer l'application du calcul de la puissance du bruit de sortie, deux exemples sont traités.

Filtre FIR

Soit l'exemple du FIR défini en section 3.2.1. Pour rappel, la sortie du système est égale à :

$$b'(n) = H^t B(n) \tag{3.210}$$

où $B(n)$ est le vecteur bruit d'entrée et $b'(n)$ le bruit de sortie scalaire. Seul ce terme de bruit est considéré.

Caractéristiques du bruit d'entrée Comme $B(n) = [b(n)b(n-1)...b(n-N+1)]^t$, la matrice d'autocorrélation du bruit $B(n)$ est égale à :

$$E(B(n)B(n-\theta)^t) = \begin{pmatrix} \mu_B^2 & \dots & \mu_B^2 & \dots & \mu_B^2 \\ \dots & \dots & \dots & \dots & \dots \\ \mu_B^2+\sigma_B^2 & \dots & \mu_B^2 & \dots & \mu_B^2 \\ \dots & \mu_B^2+\sigma_B^2 & \dots & \dots & \dots \\ \mu_B^2 & \dots & \mu_B^2+\sigma_B^2 & \dots & \mu_B^2 \end{pmatrix} \tag{3.211}$$

Cette matrice est composée des termes de moyenne au carré μ_B^2 et de variance σ_B^2. Alors que les termes de moyenne apparaissent au niveau de chaque échantillon, la variance apparaît sur la $\theta^{\text{ème}}$ diagonale inférieure. Si $\theta = 0$ alors σ_B^2 est situé sur la diagonale principale. De ce fait, si $\theta > N-1$, alors σ_B^2 n'apparaît pas.

Caractéristiques du bruit de sortie Les statistiques du bruit de sortie sont déterminées dans ce paragraphe. Sa moyenne est égale à :

$$\begin{aligned} E(b'(n)) = & \quad H^t m_B \\ = & \quad \mu_B \sum_i h_i \end{aligned} \tag{3.212}$$

où $m_B = [\mu_B,....,\mu_B]^t$ est le vecteur des moyennes de $B(n)$ et μ_B la moyenne d'un échantillon du vecteur $B(n)$. L'autocorrélation du bruit de sortie est définie par la relation suivante :

$$E(b'(n)b'(n-\theta)) = E(H^t B(n) B^t(n-\theta) H) \tag{3.213}$$

Différentes valeurs de θ sont étudiées. La puissance de $b'(n)$ ($\theta = 0$) est alors donnée par l'expression suivante :

$$\begin{aligned} E(b'(n)^2) = & \quad \sigma_B^2 H^t H + \mu_B^2 H^t \mathbf{1_N} H \\ = & \quad \sigma_B^2 \sum_i h_i^2 + \mu_B^2 (\sum_i h_i)^2 \end{aligned} \tag{3.214}$$

où $\mathbf{1_N}$ désigne la matrice composée de 1. Si $0 < \theta < N-1$, alors l'autocorrélation du bruit de sortie a pour valeur :

$$E(b'(n)b'(n-\theta)) = \sigma_B^2 \sum_{i=0}^{N-1-\theta} h_i h_{i+\theta} + \mu_B^2 (\sum_i h_i)^2 \tag{3.215}$$

Si $\theta > N-1$, alors l'autocorrélation de $b'(n)$ se résume à :

$$E(b'(n)b'(n-\theta)) = \mu_B^2 (\sum_i h_i)^2 \tag{3.216}$$

De ce fait, la covariance de $b'(n)$ est égale à :

$$E(b'(n)b'(n-\theta)) - E(b'(n))^2 = \sigma_B^2 \sum_{i=0}^{N-1-\theta} h_i h_{i+\theta} \tag{3.217}$$

Sa variance, définie pour $\theta = 0$, est égale à :

$$\sigma_{b'}^2 = \sigma_B^2 \sum_{i=0}^{N-1} h_i^2 \tag{3.218}$$

La puissance du bruit de sortie est identique à celle obtenue avec la méthode proposée en section 1.3.3, page 26. Le modèle développé permet d'aboutir à des résultats corrects pour un système LTI non-récursif.

Filtre IIR

Dans ce paragraphe, le calcul de la puissance du bruit est développé pour un système LTI récursif. L'exemple du filtre IIR est repris. Seul le bruit $b(n)$ sur les données d'entrée est considéré. Le bruit en sortie du système est défini par la relation suivante :

$$b'(n) = \sum_{i=0}^{n} h(i) b(i) == \sum_{i=0}^{n} \mathbf{A}_{(i)}^{(n)} b(i) \tag{3.219}$$

où h désigne la réponse impulsionnelle du système. L'exemple s'exprime sous la forme d'une somme infinie.

Caractéristiques des bruits d'entrée Les bruits $b(i)$ sont blancs et indépendants les uns des autres donc leur corrélation est égale à :

$$E(b(i)b(j)) = \mu_b^2 + \sigma_b^2\delta(i-j) \tag{3.220}$$

où μ_b désigne la moyenne d'un échantillon de bruit d'entrée et σ_b^2 sa variance.

Caractéristiques du bruit de sortie Dans ce paragraphe, la puissance du bruit de sortie est calculée. La moyenne de $b'(n)$ est égale à :

$$m_{b'} = E(b'(n)) = \sum_{i=0}^{n} h(i)\mu_b \tag{3.221}$$

Son autocorrélation est définie par l'expression suivante :

$$\begin{aligned} E(b'(n)b'(n-\theta)) = & \quad E(\sum_{i=0}^{n} h(i)b(i) \sum_{j=0}^{n-\theta} h(j)b(j)) \\ = & \quad \sum_{i=0}^{n}\sum_{j=0}^{n-\theta} h(i)h(j)E(b(i)b(j)) \\ = & \quad \mu_b^2 \sum_{i=0}^{n}\sum_{j=0}^{n-\theta} h(i)h(j) + \sigma_b^2 \sum_{i=0}^{n-\theta} h(i)^2 \end{aligned} \tag{3.222}$$

Sa covariance est donnée par la relation :

$$E[(b'(n)-\mu_{b'})(b'(n-\theta)-\mu_{b'})] = \sigma_b^2 \sum_{i=0}^{n-\theta} h(i)^2 \tag{3.223}$$

Sa variance est alors égale à :

$$\sigma_{b'}^2 = \sigma_b^2 \sum_{i=0}^{n} h(i)^2 \tag{3.224}$$

L'expression obtenue en section 1.3.3 pour les système LTI est retrouvée ici. En effet, le terme de variance du bruit de sortie est composé de la variance du bruit d'entrée multipliée par la somme des termes de la réponse impulsionnelle au carré. Cependant, cette expression s'exprime sous la forme d'une somme infinie. La première solution consiste à tronquer cette somme au bout d'un nombre p d'échantillons de réponse impulsionnelle. L'autre solution, la prédiction linéaire est présentée dans le paragraphe suivant.

Prédiction linéaire Pour simplifier les calculs, la prédiction linéaire peut être appliquée. Si le système est un IIR d'ordre 1 alors le coefficient de prédiction linéaire λ vérifie l'équation suivante :

$$h(i-1) = \lambda h(i) \tag{3.225}$$

La variance de $b'(n)$ s'écrit alors

$$\sigma_{b'}^2 = \sigma_b^2 \frac{h(n)^2}{1-\lambda^2} \tag{3.226}$$

et sa puissance est donnée par la relation suivante

$$E(b'(n)^2) = \sigma_b^2 \frac{h(n)^2}{1-\lambda^2} + \mu_b^2 \frac{h(n)^2}{(1-\lambda)^2} \tag{3.227}$$

Les équations sont donc fortement simplifiées par cette méthode. La somme infinie n'apparaît plus. Pour cet exemple, l'expression précédente est identique à l'équation (1.79) pour les systèmes

LTI en utilisant la fonction de transfert. En effet, le système traité est un IIR d'ordre 1 de fonction de transfert

$$H(z) = \frac{b_0}{1 - a_1 z^{-1}} \tag{3.228}$$

Le coefficient de prédiction λ est égal à a_1. Le terme multipliant la variance des termes de bruit dans l'expression (1.79) est défini de la manière suivante :

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\theta})|^2 d\theta \tag{3.229}$$

Ce terme peut être calculé de deux façons. Soit l'égalité de Parseval est utilisée :

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\theta})|^2 d\theta = \sum_{i=0}^{\infty} |h^2(i)| \tag{3.230}$$

Ou bien, la fonction de transfert est développée.

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\theta})|^2 d\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{b_0^2}{1 + a_1^2 - 2a_1 cos(\theta)} d\theta \tag{3.231}$$

Dans les deux cas, le résultat suivant est obtenu.

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\theta})|^2 d\theta = \frac{b_0^2}{(1 - a_1^2)} \tag{3.232}$$

Or, dans notre approche, le terme multipliant la variance a pour valeur :

$$\frac{h(n)^2}{1 - \lambda^2} \tag{3.233}$$

Comme $\lambda = a_1$ et $h(n) = b_0$, l'égalité entre les termes est vérifiée. De la même façon, l'égalité entre les termes multipliant les moyennes peut être vérifiée. Pour un IIR d'ordre supérieur, notre approche fournit également un résultat identique à celui de l'expression (1.79). Cet exemple illustre l'utilisation de la méthode par prédiction linéaire pour le calcul de la puissance du bruit. Les méthodes utilisant la somme ou les termes de prédiction permettent de retrouver des résultats bien connus pour les systèmes LTI. Pour ces systèmes, la méthode utilisant la somme est analogue au calcul de la puissance du bruit avec les termes de réponse impulsionnelle. L'approche par prédiction est identique à la méthode utilisant la fonction de transfert.

Conclusion

Ces deux exemples ont permis d'appliquer les expressions de calcul des caractéristiques du bruit de sortie. De plus, la méthode de prédiction linéaire est utilisée pour permettre de réduire la complexité des calculs. Pour les systèmes LTI, des résultats bien connus sont retrouvés par les deux méthodes.

3.4 Application à l'algorithme LMS

Pour illustrer le modèle précédemment développé l'exemple complet du LMS est traité dans cette partie. Le LMS en précision infinie est représenté sur la figure 3.22. Lors du passage de l'algorithme LMS en précision finie, plusieurs bruits sont générés.

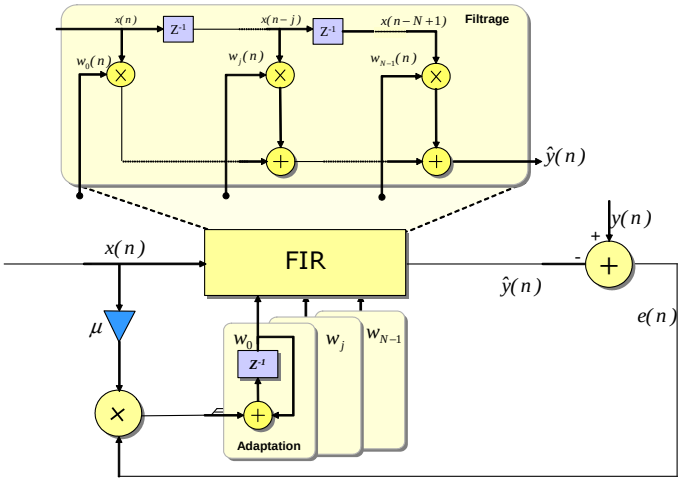

FIG. 3.22 – *Algorithme LMS en précision infinie*

– Le bruit $\alpha(n)$ est généré par la quantification du signal d'entrée $X(n)$. Le signal d'entrée quantifié est noté $X'(n)$.

– Le bruit $\beta(n)$ est dû à la quantification du signal désiré $y(n)$, dont la version quantifiée est notée $y'(n)$.

– Le bruit $\gamma(n)$ est généré par la quantification du résultat du produit de l'erreur quantifiée $e'(n)$, du signal d'entrée quantifié $X'(n)$ et du pas d'adaptation μ lors de la mise à jour des coefficients.

– Le bruit $\eta(n)$ modélise l'ensemble des bruits apparaissant dans le FIR. Le signal en sortie du FIR est alors noté $\widehat{y}'(n)$.

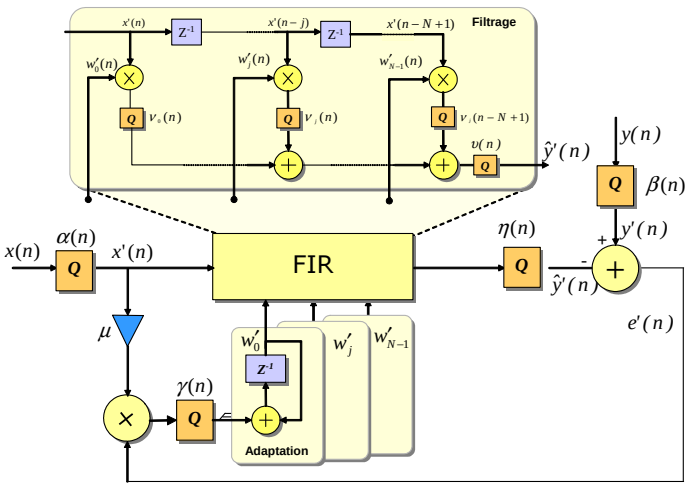

FIG. 3.23 – *Algorithme LMS en précision finie*

L'algorithme LMS en précision finie est représenté sur la figure 3.23. Le bruit en sortie du système $b_y(n)$ est généré par la somme des contributions de chaque source de bruit. Celles-ci sont au nombre de quatre et ont été définies auparavant. La propagation de ces sources est maintenant étudiée.

3.4.1 Propagation des bruits dans le système

Dans cette partie sont étudiées les propagations des différentes sources de bruit générées par l'implantation en précision finie de l'application. L'instant de sortie est n. De ce fait, pour simplifier les notations, les termes de pseudo réponse impulsionnelle sont simplement notés $h(k)$ et non plus $h_{(k)}^{(n)}$.

Propagation du bruit $\gamma(n)$

Le bruit $\gamma(n)$ est généré lors de la mise à jour des coefficients. Ce terme de bruit est un vecteur de taille N. Tout d'abord, le chemin parcouru par ce bruit est déterminé. Celui-ci est représenté sur le graphe 3.24. Ce graphe peut être modifié de façon à relier les branches de même retard. Le graphe ainsi modifié apparaît en figure 3.25. Celui-ci illustre la propagation du bruit $\gamma(n)$ au travers de deux blocs au sein du système. Le premier est la récursion propre au système et le second correspond à la multiplication par le signal d'entrée $X^t(n)$. Le bruit $\gamma(n)$ se propage donc au travers de deux blocs en série notés $S1_\gamma$ et $S2_\gamma$. En notant $\mathbf{F}(n)$ la matrice définie par

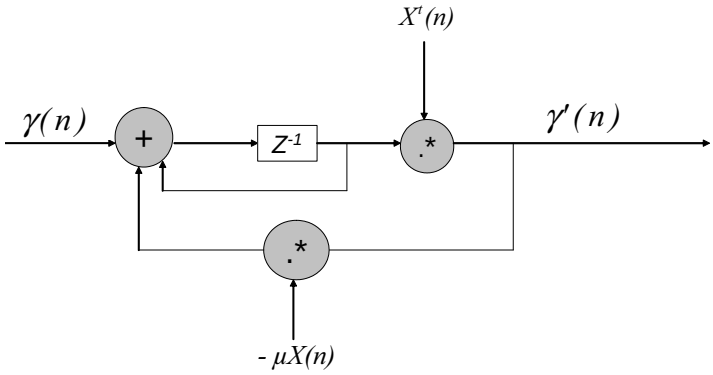

FIG. 3.24 – *Propagation du bruit* $\gamma(n)$

$$\mathbf{F}(n) = \mathbf{I_N} - \mu X(n) X^t(n) \tag{3.234}$$

La pseudo fonction de transfert $H_\gamma^{(n)}(z)$ du système parcouru par le bruit $\gamma(n)$ est définie par l'expression suivante :

$$H_\gamma^{(n)}(z) = X^t(n) \frac{z^{-1}}{1 - \mathbf{F}(n-1) z^{-1}} \tag{3.235}$$

A partir de l'expression de la pseudo fonction de transfert du bruit $\gamma(n)$, le calcul de sa pseudo réponse impulsionnelle $h_\gamma(n)$ est effectué. Ses premiers termes sont donnés par les expressions suivantes :

$$\begin{aligned} h_\gamma(n) &= 0 \\ h_\gamma(n-1) &= X^t(n) \\ h_\gamma(n-2) &= X^t(n)\mathbf{F}(n-1) \end{aligned} \tag{3.236}$$

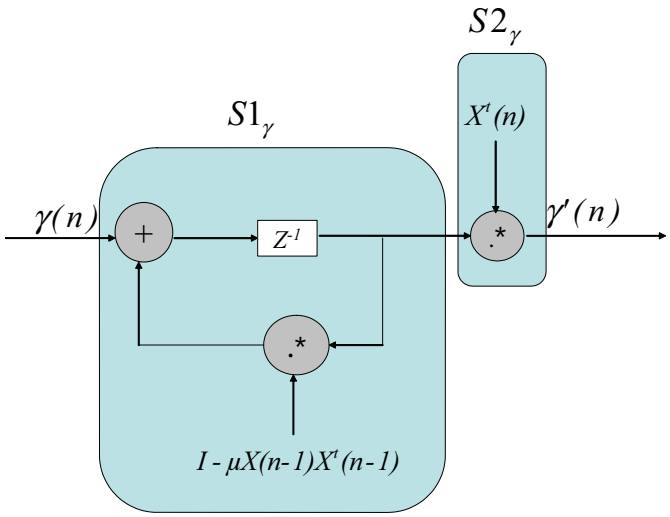

FIG. 3.25 – *Modélisation de la propagation du bruit* $\gamma(n)$

La fonction h_γ s'écrit comme le produit de deux matrices $\mathbf{A}_\gamma$ et $\mathbf{D}_\gamma$. Les termes de pseudo réponse impulsionnelle calculés précédemment montrent que le bruit $\gamma(n)$ est multiplié à gauche par la fonction h_γ correspondant à la réponse impulsionnelle. La matrice $\mathbf{A}_\gamma$ est donc égale à h_γ et la matrice $\mathbf{D}_\gamma$ est réduite à l'unité 1. Les valeurs calculées de h_γ la définissent comme un vecteur ligne de taille N. La matrice $\mathbf{A}_\gamma$ est donc un vecteur noté A_γ. Les termes A_γ sont alors définis en continuant le développement de la pseudo fonction de transfert $H_\gamma^{(n)}(z)$. Leur expression générale est donnée par la formule suivante :

$$A_\gamma(k) = X^t(n) \prod_{m=k+1}^{n-1} \mathbf{F}(m) \tag{3.237}$$

La contribution $\gamma'(n)$ du bruit $\gamma(n)$ est alors exprimée par la relation suivante :

$$\begin{aligned} \gamma'(n) &= \sum_{k=0}^{n-1} A_\gamma(k)\gamma(k) \\ \gamma'(n) &= \sum_{k=0}^{n-1} X^t(n) \prod_{m=k+1}^{n-1} \Big(\mathbf{I_N} - \mu X(m)X^t(m)\Big)\gamma(k) \end{aligned} \tag{3.238}$$

Cette expression s'écrit comme une somme infinie. Pour calculer les caractéristiques du bruits en sortie, un nombre représentatif d'échantillons de la pseudo réponse impulsionnelle doit être pris en compte. Ce nombre peut être très important. Pour résoudre ce problème, les termes de pseudo réponse impulsionnelle sont approchés en utilisant la prédiction linéaire présentée en partie 3.3.3. Pour appliquer ce modèle, les coefficients de prédiction linéaire sont appliqués comme expliqué en section 3.3.3 page 115. La pseudo fonction de transfert précédente donne un ordre de récursion de valeur 1 (terme en z^{-1} au dénominateur). Pour calculer le coefficient de prédiction, les premiers termes de A_γ sont déterminés. Comme l'ordre de prédiction est 1, et l'ordre du numérateur de la fonction de transfert est de 2 (terme en z^{-1} au numérateur), les 3 premiers termes de réponse impulsionnelle de A_γ sont établis et repris dans l'expression suivante :

$$\begin{aligned} A_\gamma(n) &= 0 \\ A_\gamma(n-1) &= X^t(n) \\ A_\gamma(n-2) &= X^t(n)\big(\mathbf{I_N} - \mu X(n-1)X^t(n-1)\big) \end{aligned} \tag{3.239}$$

La matrice $\mathbf{S}$ est alors définie par $E(A_\gamma(n-1)A_\gamma^t(n-1))$ et est donnée par l'expression suivante :

$$\mathbf{S} = E\big(X^t(n)X(n)\big) \tag{3.240}$$

L'opérateur Trace n'apparaît pas car les termes $E(A_\gamma A_\gamma^t)$ sont des scalaires. Pour calculer le coefficient de prédiction linéaire, le vecteur J est calculé. Il est défini par la relation suivante :

$$\begin{aligned} J &= E(A_\gamma(n-2)A_\gamma^t(n-1)) \\ &= E\Big(X^t(n)\big(\mathbf{I_N} - \mu X(n-1)X^t(n-1)\big)X(n)\Big) \end{aligned} \tag{3.241}$$

Ces deux termes se résument ici à des scalaires. Le coefficient de prédiction linéaire (noté λ_γ) est défini par la relation suivante :

$$\begin{aligned} \lambda_\gamma &= \frac{J}{\mathbf{S}} \\ &= \frac{E\Big(X^t(n)\big(\mathbf{I_N} - \mu X(n-1)X^t(n-1)\big)X(n)\Big)}{E\big(X^t(n)X(n)\big)} \end{aligned} \tag{3.242}$$

Si les échantillons d'entrées contenus dans le vecteur $X(n)$ sont non-corrélés entre eux, le coefficient λ_γ peut s'écrire sous la forme simplifiée suivante :

$$\lambda = \frac{Tr(\mathbf{R}_{xx}) - \mu Tr(\mathbf{R}_{xx}^2)}{Tr(\mathbf{R}_{xx})} \tag{3.243}$$

avec $\mathbf{R}_{xx}$ la matrice d'autocorrélation de $X(n)$ définie par $\mathbf{R}_{xx} = E(X(n)X^t(n))$ et Tr l'opérateur trace sommant les termes de la diagonale de la matrice considérée. Pour appliquer le modèle, seuls les deux premiers termes de pseudo réponse impulsionnelle correspondant à $A_\gamma(n)$ et $A_\gamma(n-1)$ sont conservés. Le coefficient λ_γ permet alors de calculer tous les autres termes de pseudo réponse impulsionnelle.

Propagation du bruit $\beta(n)$

Le terme $\beta(n)$ correspond au bruit généré par le signal désiré $y(n)$. Ce bruit est donc un scalaire. La propagation de ce bruit au sein du système est modélisé par la figure 3.26. Celui-ci génère en sortie du système un bruit $\beta'(n)$ correspondant à sa contribution au bruit de sortie $b_y(n)$. Le bruit $\beta(n)$ parcourt trois systèmes placés en séries. Le premier système $S1_\beta$ est défini comme la multiplication par $\mu X(n)$. Le deuxième système $S2_\beta$ correspond au passage dans le système récursif. Le dernier système $S3_\beta$ est défini par la multiplication par les données d'entrée $X^t(n)$.

La pseudo fonction de transfert modélisant la propagation du bruit $\beta(n)$ au sein du système est donnée par la relation suivante :

$$H_\beta^{(n)}(z) = X^t(n)\frac{z^{-1}}{1 - \mathbf{F}(n-1)z^{-1}}\mu X(n) \tag{3.244}$$

Le calcul de la pseudo réponse impulsionnelle h_β aboutit aux termes suivants :

$$\begin{aligned} h_\beta(n) &= 0 \\ h_\beta(n-1) &= \mu X^t(n)X(n-1) \\ h_\beta(n-2) &= \mu X^t(n)\mathbf{F}(n-1)\mu X(n-2) \end{aligned} \tag{3.245}$$

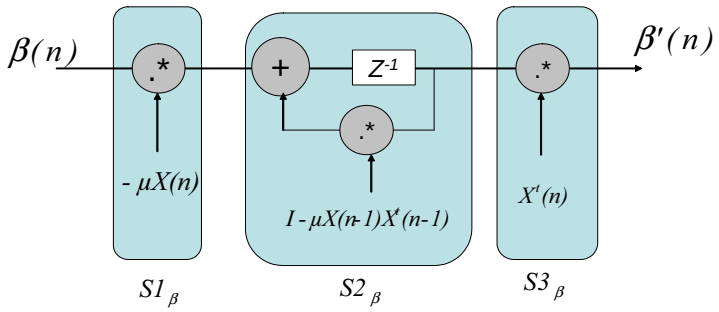

FIG. 3.26 – *Propagation du bruit* $\beta(n)$

La pseudo réponse impulsionnelle h_β s'exprime par un terme scalaire a_β. De ce fait, le bruit généré par $\beta(n)$ s'écrit sous la forme

$$\begin{aligned}\beta'(n) &= \sum_{k=0}^{n-1} a_\beta(k)\beta(k) \\ &= \mu X^t(n) \sum_{k=0}^{n-1} \prod_{m=k+1}^{n-1} \Big(\mathbf{I_N} - \mu X(m)X^t(m)\Big) X(k)\beta(k) \end{aligned} \tag{3.246}$$

Son terme de prédiction linéaire λ_β est également calculé et est égal à :

$$\begin{aligned}\lambda_\beta &= \frac{E\left[a_\beta(n-2)a_\beta^t(n-1)\right]}{E\left[a_\beta(n-1)a_\beta^t(n-1)\right]} \\ &= \frac{\mu^2 E(X^t(n)[I_N - \mu X(n-1)X^t(n-1)]X(n-2)X^t(n-1)X(n))}{\mu^2 E(X^t(n)X(n-1)X^t(n-1)X(n))} \end{aligned} \tag{3.247}$$

La contribution du bruit $\beta(n)$ est donc déterminée de cette façon.

Propagation du bruit $\eta(n)$

Le bruit $\eta(n)$ modélise l'ensemble des bruits généré dans la FIR comme le montre la figure 3.27 et l'expression suivante :

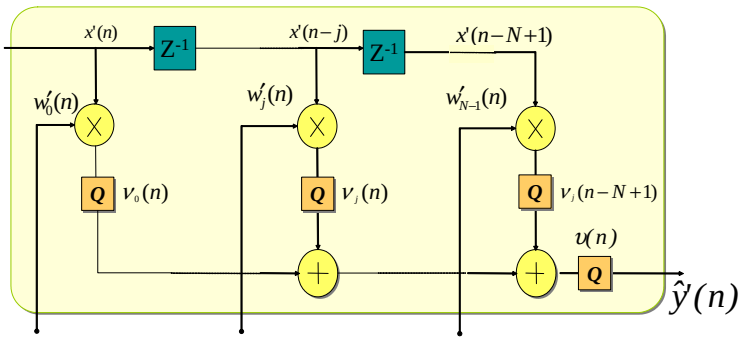

FIG. 3.27 – *Schéma du FIR du LMS en précision finie*

$$\eta(n) = \sum_{i=0}^{N-1} \nu_i(n) + \upsilon(n) \tag{3.248}$$

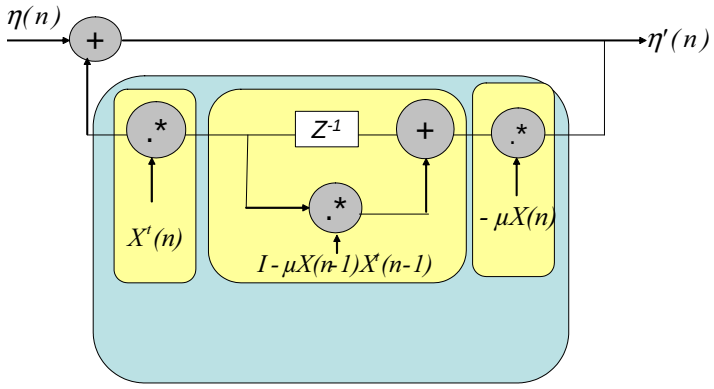

FIG. 3.28 – *Propagation du bruit* $\eta(n)$

Ce bruit est scalaire et se propage ensuite dans le système selon le graphe présenté sur la figure 3.28. La propagation de ce bruit se modélise sous la forme de la pseudo fonction de transfert suivante :

$$H_\eta^{(n)}(z) = 1 - X^t(n)\frac{z^{-1}}{1 - \mathbf{F}(n-1)z^{-1}}\mu X(n) \tag{3.249}$$

La pseudo réponse impulsionnelle h_η est égale à :

$$\begin{aligned} h_\eta(n) &= 1 \\ h_\eta(n-1) &= -\mu X^t(n)X(n-1) \\ h_\eta(n-2) &= -\mu X^t(n)\mathbf{F}(n-1)\mu X(n-2) \end{aligned} \tag{3.250}$$

La pseudo réponse impulsionnelle h_η s'exprime par un terme scalaire a_η. Le terme $\mathbf{D}$ n'apparaît pas dans l'expression. Le bruit généré par $\eta(n)$ s'écrit sous la forme suivante :

$$\begin{aligned} \eta'(n) &= \sum_{k=0}^{n} a_\eta(k)\eta(k) \\ \eta'(n) &= \eta(n) - \mu X^t(n)\sum_{k=0}^{n-1}\prod_{m=k+1}^{n-1}\Big(\mathbf{I_N} - \mu X(m)X^t(m)\Big)X(k)\eta(k) \end{aligned} \tag{3.251}$$

Les termes a_η sont définis par l'expression suivante :

$$\begin{aligned} a_\eta(n) &= 1 \\ a_\eta(k) &= -\mu X^t(n)\prod_{m=k+1}^{n-1}\Big(I - \mu X(m)X^t(m)\Big)X(k) \text{ si k<n} \end{aligned} \tag{3.252}$$

Le coefficient de prédiction linéaire λ_η est ensuite calculé comme pour les bruits précédents.

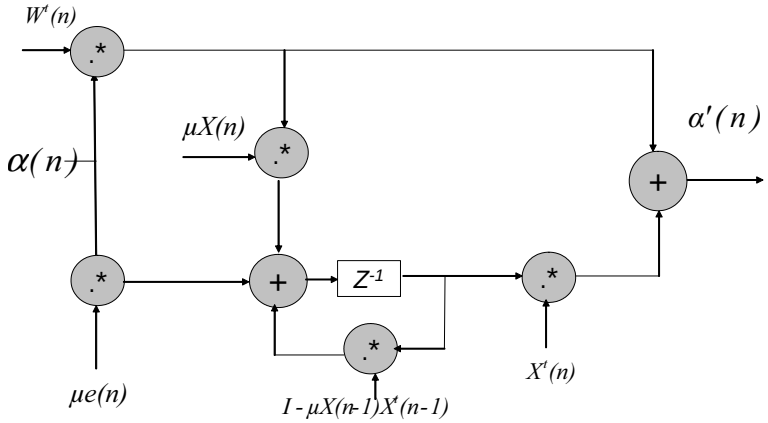

FIG. 3.29 – *Propagation du bruit* $\alpha(n)$

Propagation du bruit $\alpha(n)$

Le dernier terme de bruit $\alpha(n)$ est généré par la quantification des données d'entrée $X(n)$. Ce terme est donc un vecteur de taille N. La propagation de ce bruit est représenté sur la figure 3.29.

La pseudo fonction de transfert H_α modélisant le système complet est alors définie par l'expression suivante :

$$H_\alpha^{(n)}(z) = \mu X^t(n) \frac{z^{-1}}{1 - \mathbf{F}(n-1)z^{-1}} (e(n) + X(n)W^t(n)) + W^t(n) \tag{3.253}$$

Le modèle de la propagation du bruit $\alpha(n)$ est obtenu en utilisant les pseudo fonctions de transfert. En développant cette fonction, la pseudo réponse impulsionnelle h_α est déterminée dans l'équation suivante :

$$\begin{aligned} h_\alpha(n) &= W^t(n) \\ h_\alpha(n-1) &= \mu X^t(n)(e(n-1) - X(n-1)W^t(n-1)) \end{aligned} \tag{3.254}$$

La réponse impulsionnelle h_α est donc composée d'un terme vectoriel A_α défini par

$$\begin{aligned} A_\alpha(n) &= W^t(n) \\ A_\alpha(n-1) &= \mu X^t(n)(e(n-1) + X(n-1)W^t(n-1)) \end{aligned} \tag{3.255}$$

Le bruit $\alpha'(n)$ dû à la propagation du bruit $\alpha(n)$ dans le système s'écrit donc sous la forme suivante :

$$\alpha'(n) = \sum_{k=0}^{n} A_\alpha(k)\alpha(k) \tag{3.256}$$

Pour obtenir la valeur du coefficient de prédiction λ_α associé au bruit $\alpha(n)$, une méthode identique à celle développée pour les autres termes est mise en œuvre.

3.4.2 Puissance du bruit de sortie

Le bruit de sortie $b_y(n)$ s'écrit comme la somme de toutes les contributions des bruits d'entrée comme le montre l'expression suivante :

$$b_y(n) = \sum_{k=0}^{n} A_\alpha(k)\alpha(k) + a_\beta(k)\beta(k) + a_\eta(k)\eta(k) + A_\gamma(k)\gamma(k) \tag{3.257}$$

Avant de calculer la puissance du bruit de sortie, les caractéristiques statistiques des bruits d'entrée sont définies dans le paragraphe suivant.

Caractéristiques des bruits d'entrée Les 4 bruits d'entrée considérés sont des bruits de quantification. Ils sont donc non-corrélés entre eux. Les bruits $\eta(n)$ et $\beta(n)$ sont scalaires. Leurs caractéristiques statistiques sont définies dans l'expression suivante :

$$\begin{aligned} E(\beta(k)\beta(l)) &= \mu_\beta^2 + \sigma_\beta^2\delta(k-l) \\ E(\eta(k)\eta(l)) &= \mu_\eta^2 + \sigma_\eta^2\delta(k-l) \\ E(\beta(k)\eta(l)) &= \mu_\beta\mu_\eta \end{aligned} \tag{3.258}$$

L'autocorrélation du bruit vectoriel γ est donnée par l'expression suivante :

$$E(\gamma(k)\gamma(k-\theta))^t) = \begin{pmatrix} \mu_\gamma^2 + \sigma_\gamma^2\delta(\theta) & \dots & \mu_\gamma^2 & \dots & \mu_\gamma^2 \\ \dots & \dots & \dots & \dots & \dots \\ \mu_\gamma^2 & \dots & \mu_\gamma^2 + \sigma_\gamma^2\delta(\theta) & \dots & \mu_\gamma^2 \\ \dots & \dots & \dots & \dots & \dots \\ \mu_\gamma^2 & \dots & \mu_\gamma^2 & \dots & \mu_\gamma^2 + \sigma_\gamma^2\delta(\theta) \end{pmatrix} \tag{3.259}$$

Sa covariance est alors égale à :

$$\mathbf{C}_{\gamma\gamma}(\theta) = E((\gamma(k) - m_\gamma)(\gamma(k-\theta) - m_\gamma)^t) = \begin{pmatrix} \sigma_\gamma^2\delta(\theta) & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \sigma_\gamma^2\delta(\theta) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \sigma_\gamma^2\delta(\theta) \end{pmatrix} \tag{3.260}$$

Cette dernière s'écrit simplement sous la forme suivante :

$$\mathbf{C}_{\gamma\gamma}(\theta) = \sigma_\gamma^2\mathbf{I_N}\delta(\theta) \tag{3.261}$$

De la même façon, la covariance du vecteur bruit $\alpha(n)$ s'écrit de la manière suivante :

$$\mathbf{C}_{\alpha\alpha}(\theta) = \sigma_\alpha^2\mathbf{I_N}\delta(\theta) \tag{3.262}$$

Caractéristiques du bruit de sortie L'objectif de ce paragraphe est de déterminer la puissance du bruit en sortie du système $b_y(n)$. Les termes m_α et m_γ représentent les vecteurs moyennes des bruits $\alpha(n)$ et $\gamma(n)$. Comme toutes les sources de bruit correspondent à des bruits de quantification, la puissance du bruit s'écrit pour $\theta = 0$ dans la relation (3.133). Au sein de cette expression, les termes transposition n'apparaissent pas. Cette expression s'écrit sous la forme générale suivante :

$$\begin{aligned} \mathbf{R}_{b_y b_y}(0) &= E[b_{yu}(n)b_{yu}^t(n-\theta)] \\ &= \sum_{i=1}^{Ne}\sigma_{b_i}^2\sum_{k=0}^{n} E\Big(Tr[\underline{\mathbf{D}}_i(k)\underline{\mathbf{D}}_i^t(k)]\underline{\mathbf{A}}_i(k)\underline{\mathbf{A}}_i^t(k)\Big) \\ &+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n} E\Big[\big(\underline{\mathbf{A}}_i(k)m_{b_i}\underline{\mathbf{D}}_i(k)\big).\big(\underline{\mathbf{A}}_j(m)m_{b_j}\underline{\mathbf{D}}_j(m)\big)^t\Big] \end{aligned} \tag{3.263}$$

Dans notre exemple, $N_e = 4$ définissant les quatre sources de bruit. Les termes $\underline{\mathbf{D}}_i$ n'interviennent pas ici. Les termes $\underline{\mathbf{A}}_i$ ont été définis précédemment. En remplaçant les termes par les valeurs définies pour le système, la puissance du bruit $b_y(n)$ est alors donnée par l'expression suivante :

$$\begin{aligned} E(b_y^2(n)) &= \sum_{k=0}^{n} \sigma_\alpha^2 E(A_\alpha(k)A_\alpha^t(k)) + \sum_{k=0}^{n} \sigma_\eta^2 E(a_\eta^2(k)) \\ &+ \sum_{k=0}^{n} \sigma_\beta^2 E(a_\beta^2(k)) + \sum_{k=0}^{n} \sigma_\gamma^2 E(A_\gamma(k)A_\gamma^t(k)) \\ &+ \sum_{k=0}^{n}\sum_{l=0}^{n} E\Big(\big[A_\alpha(k)m_\alpha + a_\beta(k)\mu_\beta + a_\eta(k)\mu_\eta + A_\gamma(k)\mu_\gamma\big] \\ &. \quad \big[m_\alpha^t A_\alpha^t(l) + \mu_\beta^t a_\beta^t(l) + \mu_\eta^2 a_\eta^t(l) + \mu_\gamma^t A_\gamma^t(l)\big]\Big) \end{aligned} \tag{3.264}$$

Cette expression permet de calculer la puissance du bruit en sortie du système. Le calcul des coefficients de prédiction permet de réduire la complexité de l'expression. En effet, les termes A sont estimés par prédiction linéaire en fonction des termes précédents. Les systèmes parcourus sont récursifs d'ordre 1. L'expression (3.200) est appliquée pour $Q = 2$ et $P = 1$. De ce fait, une série géométrique apparaît comme le montre la relation suivante.

$$\begin{aligned} \sum_{k=0}^{n} A(k) &= A(n) + A(n-1) + A(n-2) + ... \\ &= A(n) + \lambda A(n) + \lambda^2 A(n) + ... \\ &= A(n)[1 + \lambda + \lambda^2 + ...] \\ &= \frac{A(n)}{1-\lambda} \end{aligned} \tag{3.265}$$

où λ désigne le coefficient de prédiction. De la même façon, les termes A^2 s'exprime sous la forme suivante :

$$\sum_{k=0}^{n} A^2(k) = \frac{A^2(n)}{1-\lambda^2} \tag{3.266}$$

Pour appliquer ce modèle, les $P + Q - 1$ premiers termes de la pseudo réponse impulsionnelle sont calculés à la main (avec P et Q les ordres du numérateur et du dénominateur de la pseudo fonction de transfert associé). Ici $P = 1$ et $Q = 2$ car le système global est récursif d'ordre 1 et un terme en z^{-1} apparaît au numérateur des pseudo fonctions de transfert. Les deux premiers termes sont calculés manuellement. La méthode par prédiction s'applique à partir du troisième échantillon de la pseudo réponse impulsionnelle. La somme infinie s'écrit dans ce cas sous la forme suivante :

$$\begin{aligned} \sum_{k=0}^{n} A(k) &= A(n) + A(n-1) + \lambda A(n-1) + \lambda^2 A(n-1)..... \\ &= A(n) + \frac{A(n-1)}{1-\lambda} \end{aligned} \tag{3.267}$$

Ainsi, dans l'expression (3.264), chaque somme est remplacée par la simplification précédente. Les termes de pseudo réponse impulsionnelle à l'instant n apparaissent séparément car la méthode

par prédiction s'applique à partir de l'instant $n-1$. Cependant, les termes $A_\gamma(n)$ et $a_\beta(n)$ ont été déterminés précédemment et sont nuls. Ceux-ci n'apparaissent pas dans l'expression. En notant λ_b, le coefficient de prédiction du bruit b, l'equation (3.264) s'écrit sous la forme suivante :

$$\begin{aligned}
E(b_y^2(n)) &= \Big[\frac{\sigma_\alpha^2}{1-\lambda_\alpha^2}E\big(A_\alpha(n-1)A_\alpha^t(n-1)\big) + E\big(A_\alpha(n)A_\alpha^t(n)\big)\Big] + \Big[\frac{\sigma_\eta^2}{1-\lambda_\eta^2}E(a_\eta^2(n-1)) + E(a_\eta^2(n))\Big] \\
&+ \frac{\sigma_\beta^2}{1-\lambda_\beta}E(a_\beta^2(n-1)) + \Big[\frac{\sigma_\gamma^2}{1-\lambda_\gamma^2}E\big(A_\gamma(n-1)A_\gamma^t(n-1)\big)\Big] \\
&+ \Bigg(\Big[E(A_\alpha(n)) + \frac{E(A_\alpha(n-1))}{1-\lambda_\alpha}\Big]m_\alpha + E(a_\beta(n-1))\frac{\mu_\beta}{1-\lambda_\beta} \\
&+ \Big[\frac{1}{1-\lambda_\eta}E(a_\eta(n-1)) + E(a_\eta(n))\Big]\mu_\eta + E(A_\gamma(n-1))\frac{m_\gamma}{1-\lambda_\gamma}\Bigg) \\
&. \Bigg(m_\alpha^t\Big[\frac{1}{1-\lambda_\alpha}E(A_\alpha^t(n-1)) + E(A_\alpha^t(n))\Big] + \frac{\mu_\beta}{1-\lambda_\beta}E(a_\beta^t(n-1)) \\
&+ \mu_\eta\Big[\frac{1}{1-\lambda_\eta}E(a_\eta^t(n-1)) + E(a_\eta^t(n))\Big] + \frac{m_\gamma^t}{1-\lambda_\gamma}E(A_\gamma^t(n-1))\Bigg)
\end{aligned} \tag{3.268}$$

La puissance du bruit s'exprime alors grâce à une équation simplifiée car les sommes infinies n'apparaissent plus. Ainsi, lors d'un processus d'optimisation, les termes de signaux $\mathbf{A}$ présents dans l'expression sont tout d'abord déterminés. Ensuite, à chaque itération du processus d'optimisation, seuls les termes m et σ^2 varient en fonction des formats appliqués aux opérateurs. L'expression précédente est alors simplement évaluée pour les nouvelles valeurs de m et σ^2. Dans cette section, la méthode de calcul de la puissance du bruit a été appliquée à l'algorithme LMS. La méthode par approximation linéaire permet de réduire la complexité de l'application.

3.4.3 Comparaison avec le modèle dédié

Dans le chapitre précédent, des modèles spécifiques à des types d'applications précises ont été définis. Pour chacun de ces modèles, les particularités inhérentes au système associé ont été prises en compte dans l'établissement du modèle dédié. Dans cette partie, au contraire, un modèle général à l'ensemble des systèmes à été présenté. Ce dernier ne fait aucune hypothèse sur le système considéré. Dans le cas où des hypothèses identiques à celles du modèle dédié sont appliquées au modèle général, la même équation doit être retrouvée. Ceci est fait dans cette section au travers de l'exemple du LMS. L'exemple du LMS a été traité en page 48 avec son modèle dédié et précédemment avec le modèle général. La puissance du bruit de sortie fournie par le modèle dédié donne l'expression suivante :

$$Pb = \|W_{opt}\|^2(\sigma_\alpha^2 + m_\alpha^2) + m_\gamma^2\frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx}^{-1}{}_{ki})}{\mu^2} + \frac{N(\sigma_\gamma^2 - m_\gamma^2)}{2\mu} + m_\eta^2 + \sigma_\eta^2 \tag{3.269}$$

L'objectif est de retrouver cette expression à partir du modèle général défini précédemment. Pour ce faire, le terme $\gamma(n)$ est traité car celui-ci prédomine dans l'expression précédente. La puissance du bruit généré par le terme $\gamma(n)$ est $m_\gamma^2\frac{\sum_{i=1}^{N}\sum_{k=1}^{N}(\mathbf{R}_{xx}^{-1}{}_{ki})}{\mu^2} + \frac{N(\sigma_\gamma^2-m_\gamma^2)}{2\mu}$. Le coefficient de prediction linéaire a été calculé et est égal à :

$$\lambda_\gamma = \frac{Tr(\mathbf{R}_{xx}) - \mu Tr(\mathbf{R}_{xx}^2)}{Tr(\mathbf{R}_{xx})} \tag{3.270}$$

Pour utiliser ce coefficient, les deux premiers termes de la pseudo réponse impulsionnelle du système parcouru par le bruit $\gamma(n)$ sont établis et ont pour valeur :

$$\begin{aligned} A_\gamma(n) &= 0 \\ A_\gamma(n-1) &= X^t(n) \end{aligned} \tag{3.271}$$

Le bruit $\gamma(n)$ est blanc. La puissance du bruit de sortie généré par $\gamma(n)$ est exprimée par l'expression (3.133) appliquée ici.

$$Pb = \sigma_\gamma^2 \sum_{i=0}^{n} E(A_\gamma(i)A_\gamma^t(i)) + m\gamma^2 \sum_{i=0}^{n}\sum_{j=0}^{n} E(A_\gamma(i)\mathbf{1_N}A_\gamma^t(j)) \tag{3.272}$$

où $\mathbf{1_N}$ est la matrice carrée unitaire de taille N. En utilisant le coefficient de prédiction λ_γ, le terme $\sum_{i=0}^{n} A_\gamma(i)A_\gamma^t(i)$ est égal à :

$$\begin{aligned} \sum_{i=0}^{n} E(A_\gamma(i)A_\gamma^t(i)) &= \frac{E(X^t(n)X(n))}{1-\lambda_\gamma^2} \\ &= \frac{Tr(\mathbf{R}_{xx})^3}{2\mu Tr(\mathbf{R}_{xx})Tr(\mathbf{R}_{xx}^2) - \mu^2 Tr(\mathbf{R}_{xx}^2)^2} \end{aligned} \tag{3.273}$$

Les termes en μ^2 sont négligés comme dans le modèle dédié. L'expression suivante est obtenue :

$$\sum_{i=0}^{n} E(A_\gamma(i)A_\gamma^t(i)) = \frac{Tr(\mathbf{R}_{xx})^2}{2\mu Tr(\mathbf{R}_{xx}^2)} \tag{3.274}$$

Or dans le modèle dédié, l'hypothèse faite de non-corrélation entre le bruit sur les coefficients et les données d'entrée suppose que ces dernières sont indépendantes. En notant m_x la moyenne et σ_x^2 la variance des données d'entrées, l'expression suivante est obtenue :

$$\begin{aligned} \sum_{i=0}^{n} E(A_\gamma(i)A_\gamma^t(i)) &= \frac{Tr(\mathbf{R}_{xx})^2}{2\mu Tr(\mathbf{R}_{xx}^2)} \\ &= \frac{N^2(\sigma_x^2+m_x^2)^2}{2\mu[N(\sigma_x^2+m_x^2)^2 + (N-1)m_x^2]} \end{aligned} \tag{3.275}$$

La moyenne des données d'entrée au carré m_x^2 étant supposée plus faible que la variance σ_x^2, l'expression précédente est équivalente à $\frac{N}{2\mu}$ désignant le terme multiplicateur de σ_γ^2 dans le modèle dédié. Dans les deux approches, les mêmes termes sont obtenus. De la même façon, en développant l'expression de la moyenne $\sum_{i=0}^{n}\sum_{j=0}^{n} E(\mathbf{A}_\gamma(i)\mathbf{1_N}\mathbf{A}_\gamma(j)^t)$, avec des hypothèses identiques à celles du modèle dédié, la même expression est obtenue.

Les deux autres termes présents dans le modèle dédié sont $\|W_{opt}\|^2(\sigma_\alpha^2+m_\alpha^2)$ et $\sigma_\eta^2+m_\eta^2$ et se retrouvent en prenant les termes A des bruits $\alpha(n)$ et $\eta(n)$ à l'instant n. De ce fait, l'expression de la puissance du bruit en sortie du système dû au bruit $\gamma(n)$ se retrouve à partir du modèle général. De la même façon, la puissance du bruit générée par les autres termes $\alpha(n)$, $\beta(n)$ et $\eta(n)$ peut être retrouvée de la même façon. Ainsi, cet exemple permet de mettre en lumière le lien existant entre le modèle général et la modèle dédié. La même expression est obtenue dans les deux cas en utilisant les mêmes hypothèses.

3.5 Evaluation de la qualité de la méthode proposée

Dans cette partie, les expérimentations effectuées pour évaluer la qualité du modèle général sont présentées. Pour ce faire, différents systèmes sont testés. Des systèmes LTI, non-LTI, récursifs

Quantification	**Erreur relative moyenne**	**Erreur relative maximale**
Arrondi	2.49%	7.9%
Troncature	1.14%	2.41%

TAB. 3.3 – *Erreur relative commise par l'approche générale pour un FIR32*

Bloc	**Erreur relative moyenne**	**Erreur relative maximale**
Filtre polyphase	5.24%	20.57%
DCT	8.42%	49.8%

TAB. 3.4 – *Erreur relative commise par l'approche générale pour un codeur/décodeur MP3*

et non-récursifs sont expérimentés. La qualité de l'estimation du modèle général est effectuée sur des systèmes LTI non-récursifs comme le FIR ou des blocs composant le codeur/décodeur MP3, des systèmes LTI récursifs tels que le filtre IIR, ou encore des systèmes non-LTI non-récursifs comme le carré, le module ou un filtre de Volterra. Enfin, la méthode générale est appliquée aux systèmes non-LTI récursifs représentés par les filtres adaptatifs. Pour ce faire, la méthode de calcul directe de la somme est appliquée. Les résultats de l'estimation sont comparés à l'approche basée sur l'approximation linéaire.

3.5.1 Calcul direct de la somme

Dans cette partie, le modèle général est évalué en utilisant le calcul direct de la somme. Celui-ci est appliqué à différents types de système. L'erreur relative est mesurée pour différentes spécifications virgule fixe. Les erreurs moyenne et maximale obtenues par ces expérimentations sont présentées pour chaque système.

Systèmes LTI non-récursifs

Dans ce paragraphe, des systèmes LTI non-récursifs sont évalués à l'aide de la méthode générale. Un filtre FIR de taille 32 et un codeur/décodeur MP3 sont testés. Le filtre polyphase et la DCT contenus dans le codeur/décodeur MP3 sont testés. Ces deux blocs représentent des systèmes LTI. Pour des systèmes non-récursifs, les sommes présentes dans le modèle général n'apparaissent pas. Dans ce cas, les modèles dédiés aux applications traitées sont retrouvés. Le tableau 3.3 résume les valeurs de l'erreur relative commise par l'approche générale. Les résultats fournis sont excellents puisque l'erreur moyenne est de 2.49% pour une loi de quantification par arrondi et de seulement 1.14% par troncature. De même, les valeurs maximales de l'erreur sont très faibles (de l'ordre de 8%).

Le codeur/décodeur MP3 testé est composé d'un filtre polyphase et d'une Transformée en Cosinus Discret (DCT) [97]. Le tableau 3.4 présente les erreurs moyennes et maximales obtenues par la méthode générale pour les deux blocs définis précédemment. L'erreur moyenne commise est de 5.24% pour le filtre polyphase et de 8.42% pour la DCT. Les erreurs maximales commises sont de 20.57% pour le filtre polyphase et de 49.8% pour la DCT. Cette dernière valeur est élevée mais elle représente un écart de seulement 3 dB entre la valeur réelle du RSBQ et la valeur estimée par le modèle. Les valeurs maximales sont obtenues dans des cas particuliers. En moyenne, les résultats sont bons.

Cellules	**Erreur relative moyenne**	**Erreur relative maximale**
$1^{\text{ère}}$ cellule	0.09%	1.3%
$2^{\text{ème}}$ cellule	0.78%	1.47%
$3^{\text{ème}}$ cellule	0.68%	1.43%
$4^{\text{ème}}$ cellule	1.68%	3.3%

TAB. 3.5 – *Erreur relative commise par l'approche générale pour un filtre IIR8*

Systèmes LTI récursifs

L'application de la méthode générale aux systèmes LTI récursifs est étudiée au travers du filtre IIR. Pour un système LTI récursif, la méthode générale correspond au calcul de la puissance du bruit en utilisant la réponse impulsionnelle. Les termes de la réponse impulsionnelle au carré sont sommés. Pour ce faire, le nombre p de termes pris en compte doit être défini. Pour notre exemple, le filtre IIR est d'ordre 8. Sa réponse impulsionnelle peut être tronquée au bout de 100 points. Le filtre est composée de 4 cellules cascadées d'ordre 2. Les erreurs obtenues sont résumées dans le tableau 3.5 en sortie de chacune des 4 cellules composant le filtre.

Les résultats obtenus montrent que l'erreur maximale est de 3.3%. Les estimations sont alors très précises. Pour un filtre IIR dont les pôles sont en limite de stabilité, la qualité de l'estimation est inférieure. Des erreurs de 20% sont obtenues dans ce cas. L'erreur moyenne reste néanmoins inférieure à 10%.

Systèmes non-LTI non-récursifs

Le modèle d'évaluation de la précision par la méthode directe est appliquée aux systèmes non-LTI non-récursifs dans cette partie. Pour ce faire, quatre applications sont prises en compte : le carré, le module au carré, le filtre de Volterra d'ordre 2 et le corrélateur. Soit $x(n)$ le signal en entrée du système. Le carré désigne tout simplement la multiplication du signal d'entrée par lui-même. Cette opération simple est néanmoins non-LTI tout comme le module. Le filtre de Volterra d'ordre 2 est défini par l'expression suivante :

$$y(n) = a_1 x(n-1) + a_2 x(n-2) + a_{11} x^2(n-1) + a_{22} x^2(n-2) + a_{21} x(n-1)x(n-2) \quad (3.276)$$

Enfin, pour deux séquences de scalaires $x(n)$ et $y(n)$ la corrélation est définie dans cet exemple par

$$\sum_{i=0}^{N-1} h_i [x(n-i) - y(n-i)] \quad (3.277)$$

où les coefficients h_i valent ± 1.

Le modèle général appliqué à ces systèmes non-récursifs correspond en réalité à leur modèle dédié. En effet, les systèmes étant non-récursifs, la somme du modèle général disparaît. La formule (1.82) page 32 définissant la puissance du bruit pour les systèmes non-récursif est retrouvée. Les expérimentations sont effectuées pour des lois de quantification par troncature et par arrondi. Les

Système	**Erreur relative moyenne**	**Erreur relative maximale**
Carré	0.65%	0.8%
Module	1.96%	7.58%
Filtre de volterra	1.79%	3.22%
Corrélateur	1.35%	5.78%

TAB. 3.6 – *Erreur relative commise par l'approche générale pour des systèmes non-LTI non récursifs*

résultats sont présentés dans le tableau 3.6. Pour ces applications, l'erreur est inférieure à 8% et en moyenne égale à 1.4%.

Ces résultats démontrent la validité du modèle général appliqué aux systèmes non-LTI non-récursifs. Dans la partie suivante, le modèle général est appliqué aux filtres adaptatifs

Evaluation de la méthode directe appliquée aux filtres adaptatifs

La méthode directe du modèle général est testée pour les filtres adaptatifs. Dans un premier temps l'algorithme LMS est expérimenté. Pour cet exemple, la taille du filtre est fixé à 16. Le modèle général proposé s'exprime au moyen d'une somme infinie. En pratique cette somme est tronquée à partir d'un certain nombre d'échantillons. Le nombre d'échantillons p pris en compte influe sur la qualité de l'estimation. D'autre part, des termes d'espérance mathématique sont à calculer dans l'expression de la puissance du bruit de sortie. Cette espérance prend en compte un nombre d'échantillons q. Ce nombre doit être suffisamment important pour modéliser de façon correcte la valeur exacte de l'espérance calculée. Sur la figure 3.30, l'évolution de l'erreur relative est exprimée en fonction du nombre de points pris en compte pour le calcul de la somme p, et du nombre de points q utilisés pour le calcul de l'espérance.

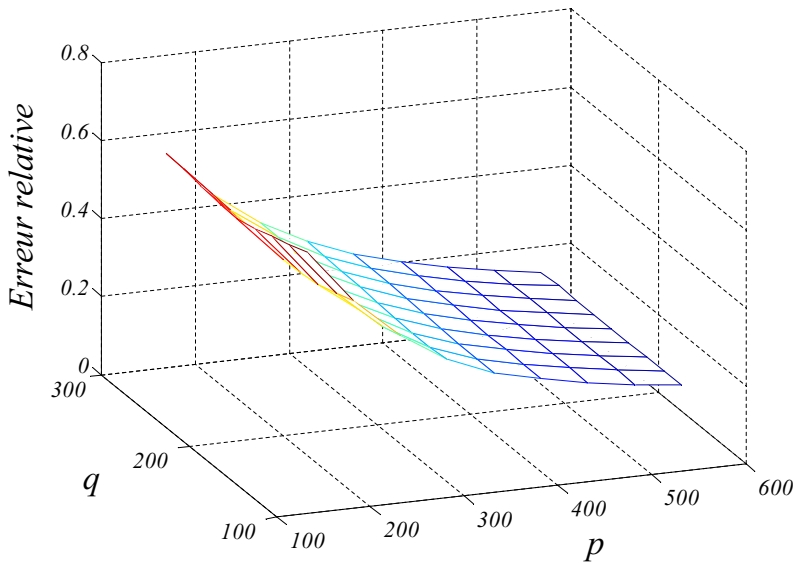

FIG. 3.30 – *Erreur relative en fonction du nombre de points de somme et d'espérance*

La figure montre que la qualité de l'estimation augmente avec l'accroissement du nombre de

points p de la somme. D'autre part, le nombre de points utilisés pour le calcul de la moyenne semble moins influer sur la qualité de l'estimation. La figure 3.31, illustre la variation de la qualité de l'estimation en fonction du nombre de points q pour le calcul de la somme pour une valeur de p fixée à 100.

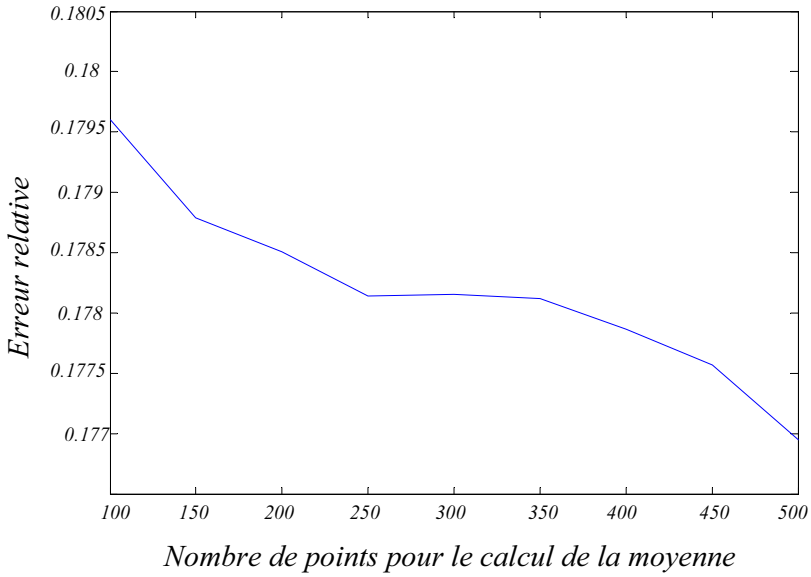

FIG. 3.31 – *Erreur relative en fonction du nombre de points d'espérance*

La variation entre les deux valeurs extrêmes $q = 100$ et $q = 500$ est très faible et de l'ordre de 1%. Un nombre de points de 100 semble suffisant pour assurer un résultat correct en fonction de la valeur du nombre de point de somme p choisi. Néanmoins, pour réduire le temps de calcul, le nombre q peut être réduit sans dégrader la qualité de l'estimation.

Sur la figure 3.34, l'évolution de l'erreur relative en fonction du nombre de points de somme p est présentée pour un nombre de points d'espérance q fixé à 100. Cette évolution est représentée pour un signal d'entrée très peu corrélé $\beta = 0.05$, moyennement corrélé $\beta = 0.5$ ou très corrélé $\beta = 0.95$. Pour chaque cas, l'erreur diminue quand la valeur de p augmente. En effet, plus p est grand, plus le nombre de termes pris en compte est important. De ce fait, la puissance de bruit évaluée est plus proche de sa valeur réelle. D'autre part, la décroissance de l'erreur relative augmente avec la corrélation du signal. En effet, les termes de signaux définissant la propagation du bruit d'entrée à l'instant k vers la sortie à l'instant n sont les suivants :

$$\mathbf{A}_{(k)}^{(n)} = \prod_{i=k}^{n-1} (\mathbf{I_N} - \mu X(i) X^t(i)) \tag{3.278}$$

Les échantillons de la diagonale de ces termes sont les plus représentatifs. La figure 3.32 illustre la décroissance de ces termes en fonction de k. Cette décroissance assure la stabilité du système. D'autre part, une décroissance est également liée à la corrélation du signal. En effet, sur la figure 3.33 est représentée la variation d'un terme de la diagonale de $\mathbf{A}_{(n-1)}^{(n)}$ en fonction de β. Plus les signaux d'entrée sont corrélés, plus les termes $X(i)X^t(i)$ sont grands. De la même façon, cette décroissance est observée pour les autres termes de signaux $\mathbf{A}_{(k)}^{(n)}$. Par conséquent, les termes de signaux $\mathbf{A}_{(k)}^{(n)}$ sont d'amplitude plus faible pour des signaux très corrélés et la convergence est plus rapide dans ce cas.

Ainsi, pour aboutir à une erreur inférieure à 20%, un nombre p de points de somme égal

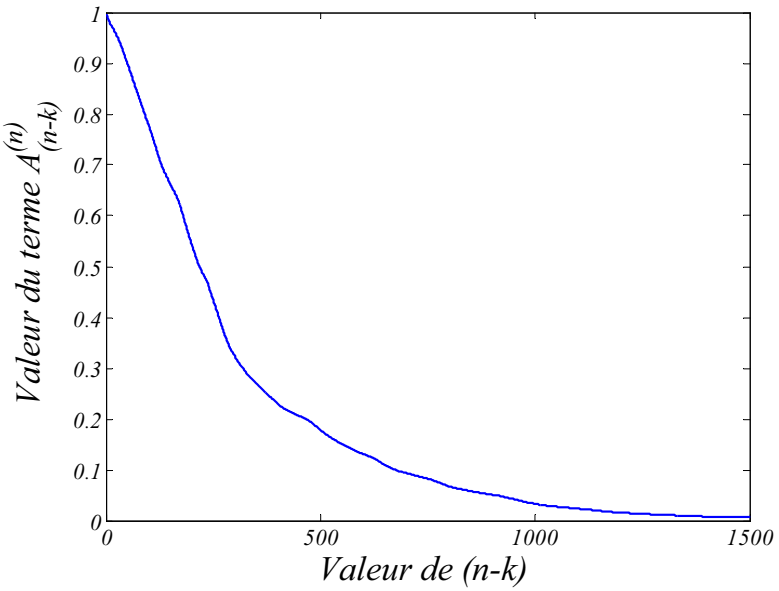

FIG. 3.32 – *Décroissance des termes de signaux modélisant la propagation du bruit*

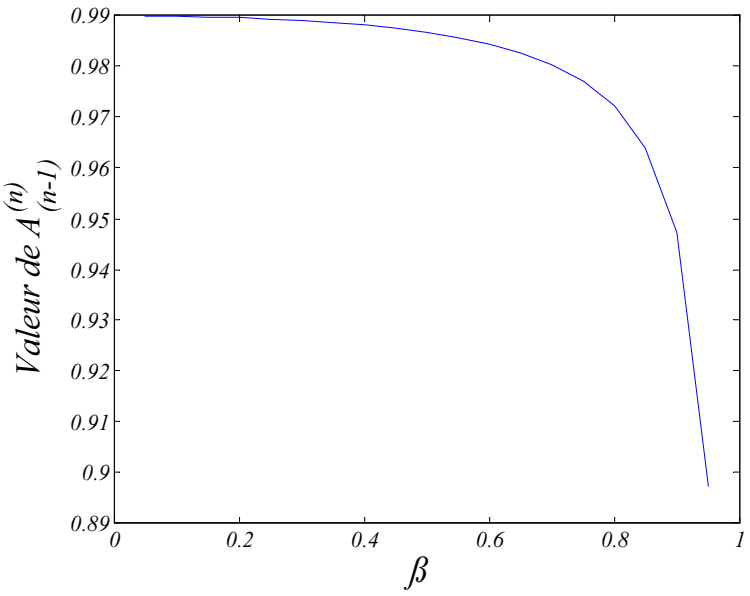

FIG. 3.33 – *Décroissance des termes de signaux en fonction de la corrélation du signal*

à 600 est nécessaire pour des signaux faiblement corrélés contre seulement 300 (soit deux fois moins) pour des signaux très corrélés. Pour un nombre de points p encore supérieur, l'erreur relative diminue encore. Ainsi, selon le type de corrélation des données d'entrée, la complexité peut doubler pour aboutir à la même qualité d'estimation. Néanmoins, pour avoir des résultats satisfaisants, un nombre p égal à 500 peut être choisi.

Dans un second temps, les Algorithmes de Projection Affine sont évalués par cette méthode. La figure 3.35 montre l'erreur relative commise en troncature pour un nombre de point de somme p égal à 500 et des données d'entrée très peu corrélées ($\beta = 0.05$).

L'erreur relative obtenue est inférieure à 40%. A titre de comparaison, avec cette approche et pour le nombre de points de somme défini précédemment, l'erreur relative est entre 5 et 10% supérieure aux résultats obtenus avec les modèles dédiés. Pour aboutir à des résultats du même ordre, un nombre de points de somme $p = 700$ doit être utilisé. Cependant, la corrélation des

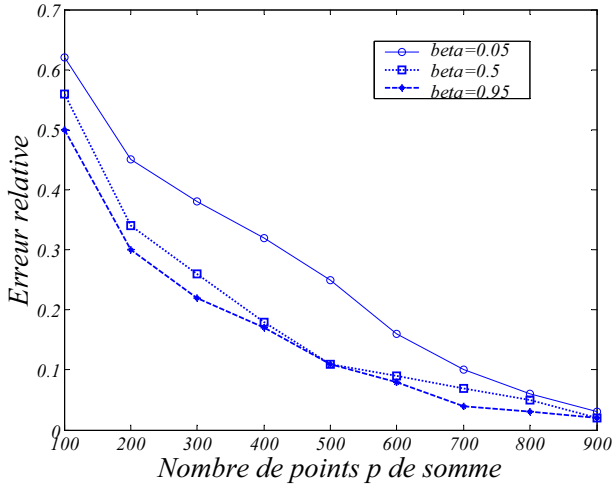

FIG. 3.34 – *Erreur relative en fonction du nombre de points de somme*

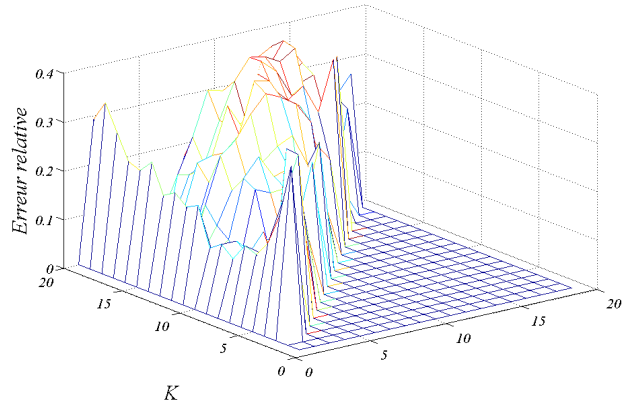

FIG. 3.35 – *Erreur relative pour les algorithmes de projection affine par la méthode directe*

données d'entrée intervient peu dans la convergence de l'erreur relative. En effet, la normalisation effectuée au sein des ces algorithmes permet d'assurer une convergence, en théorie indépendante, des propriétés de corrélation des données d'entrée.

La méthode directe permet d'aboutir à de bons résultats pour les systèmes non-récursifs comme pour les filtres adaptatifs. Pour ce faire, un nombre de points de somme de p de l'ordre de 500 doit être utilisé.

3.5.2 Méthode basée sur la prédiction linéaire

Pour réduire la complexité de la méthode directe, l'approche par prédiction linéaire a été définie. Cette méthode définie en section 3.3.3, est évaluée dans cette partie. Cette approche permet de réduire la complexité de façon très importante en approchant les termes de la somme au moyen de coefficients de prédiction. Néanmoins, cette approche introduit un biais. Ce dernier est estimé dans cette partie en mesurant l'erreur relative obtenue au moyen de diverses expérimentations. Pour les systèmes LTI, cette approche est équivalente à l'expression (1.79), page 31 de la puissance du bruit en utilisant la fonction de transfert.

Pour un filtre LMS de taille $N = 32$, et avec des données d'entrée moyennement corrélées ($\beta = 0.5$), l'erreur relative commise par la méthode utilisant les coefficients de prédiction est de 21.72%. La figure 3.36 montre l'erreur relative commise avec et sans les coefficients de prédiction en

Valeur de β	**Erreur relative**	**Nombre p equivalent**
0.05	17%	650
0.5	5.31%	575
0.95	5.83%	503

TAB. 3.7 – *Erreur relative commise par l'approche par prédiction et nombre de points de somme p aboutissant à la même erreur pour l'algorithme LMS*

fonction du nombre de termes de somme. L'erreur commise par la méthode utilisant les coefficients est indépendante du nombre de point de somme car elle la supprime. Elle dépend uniquement du nombre de points d'espérance. Pour obtenir un résultat meilleur avec la méthode directe, 350 points de somme p sont nécessaires. Considérons le cas où la taille du filtre est $N = 128$ pour une loi de quantification par troncature. Le pas d'adaptation est fixé à sa valeur médiane $\mu = \frac{\mu_{max}}{2}$. Le tableau 3.7 montre l'erreur relative obtenue pour les trois valeurs de β par la méthode de prédiction linéaire. L'erreur obtenue est comparée au nombre de points de somme p nécessaire pour aboutir à un résultat équivalent.

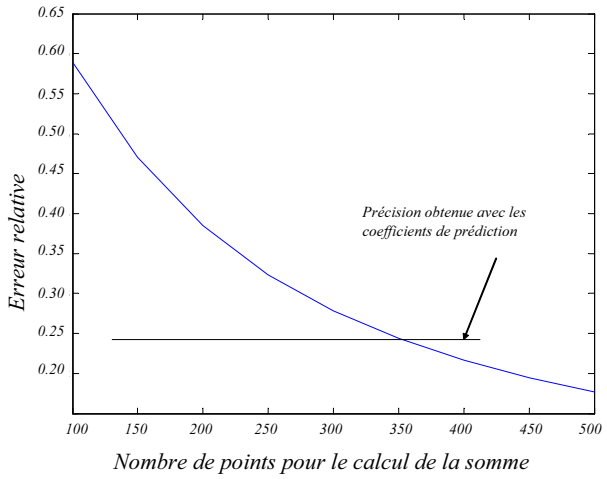

FIG. 3.36 – *Qualité de l'estimation des méthodes avec et sans les coefficients de prédiction en fonction du nombre de points de somme p*

Pour obtenir une erreur relative du même ordre que l'approche par prédiction, un nombre de point de somme d'environ 500 doit être utilisé. Considérons maintenant les Algorithmes de Projection Affine. Soit un APA de tailles $N = 16$ et $K = 8$. Des expérimentations identiques aux précédentes sont effectuées et remémorées dans le tableau 3.8. Les résultats obtenus par l'approche par coefficients sont très bons (inférieurs à 30%). Comme pour le LMS, un nombre de points p de somme proche de 500 permet d'aboutir aux mêmes résultats.

3.5.3 Comparaison avec les modèles dédiés

Dans cette partie, une comparaison entre la méthode générale et les modèles dédiés est effectuée en terme de précision. Sur la figure 3.37, les précisions de ces deux approches sont comparées pour un LMS de taille variant entre $N = 1$ et $N = 64$. L'estimation est moins bonne pour l'approche par prédiction comparée à l'approche par modèle dédié. Par exemple, pour le filtre de taille 8, l'erreur

Valeur de β	**Erreur relative**	**Nombre p equivalent**
0.05	24.65%	465
0.5	20.4%	515
0.95	26.32%	452

TAB. 3.8 – *Erreur relative commise par l'approche par prédiction et nombre de points de somme p aboutissant à la même erreur pour les APA*

relative est de 11% pour le modèle dédié alors qu'elle est de 22% pour le modèle général avec les coefficients linéaires. Pour obtenir un résultat équivalent avec le modèle général, la méthode utilisant directement la somme doit être utilisée. Avec cette méthode, pour obtenir 11% d'erreur relative, 700 points de somme doivent être pris en compte. Considérons le filtre LMS de taille 16. Dans le tableau 3.9, la précision obtenue par la méthode par prédiction et le modèle dédié est comparée pour différentes valeurs de corrélation des signaux d'entrée.

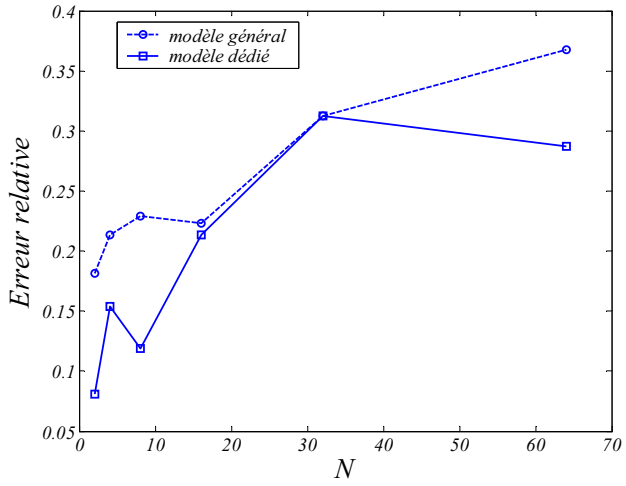

FIG. 3.37 – *Comparaison de l'erreur relative obtenue avec le modèle général et le modèle dédié pour un filtre LMS*

Les résultats obtenus sont très proches. La méthode par prédiction est même meilleure pour la cas où $\beta = 0.5$. L'approche par prédiction permet d'obtenir d'excellents résultats avec une complexité réduite. Considérons l'exemple des APA de tailles $N = 16$ et $K = 8$. Une expérimentation identique à la précédente est effectuée et les résultats sont fournis dans le tableau 3.10. L'approche par prédiction linéaire fournit une erreur relative supérieure à celle obtenue avec le modèle dédié.

Ainsi, la méthode par prédiction permet d'obtenir des résultats quasi équivalents à ceux des modèles dédiés. De plus, la complexité est réduite. Pour aboutir à de meilleurs résultats, la méthode directe de calcul de la somme doit être mise en œuvre. Néanmoins, cette approche est plus complexe.

3.6 Conclusion

Dans ce chapitre, une méthode générale d'évaluation automatique de la précision des systèmes en virgule fixe a été présentée. Après avoir défini les caractéristiques statistiques des bruits en

Valeur de β	**Méthode par prédiction**	**Modèle dédié**
0.05	17%	14.25%
0.5	5.31%	6.8%
0.95	5.83%	4.49%

TAB. 3.9 – *Erreur relative commise par l'approche par prédiction et le modèle dédié pour l'algorithme LMS de taille 16*

Valeur de β	**Méthode par prédiction**	**Modèle dédié**
0.05	24.65%	18.25%
0.5	20.4%	19.68%
0.95	26.32%	21.87%

TAB. 3.10 – *Erreur relative commise par l'approche par prédiction linéaire et le modèle dédié pour les APA*

entrée du système et leur propagation, la méthode générale a été développée. Les systèmes sont modélisés par leur pseudo fonction de transfert. Le calcul de la puissance du bruit découle de la pseudo réponse impulsionnelle du système. Pour réduire la complexité de ce calcul, une approche par prédiction linéaire a été définie. Dans une dernière partie, la qualité de l'approche est estimée par différentes expérimentations. La méthode générale permet d'obtenir une estimation précise de la puissance du bruit en sortie du système. Le biais introduit par la prédiction linéaire n'altère pas la qualité de l'estimateur.

Chapitre 4

Exploitation des modèles

Sommaire

La méthodologie d'évaluation automatique de la précision des systèmes en virgule fixe a été présentée dans les chapitres précédents. Celle-ci se base sur une approche analytique permettant une modélisation de la propagation des bruits dans le système. La précision est évaluée au travers du RSBQ de l'application traitée. La nécessité de déterminer la qualité de l'application en précision finie s'insert dans un processus plus global de conversion en virgule fixe. Cette conversion se fait à partir du code de l'application en précision infinie. L'objectif peut être d'optimisé au niveau arithmétique la génération d'un composant matériel ou logiciel spécifique au moyen d'IPs (Intellectual Properties). Dans ce cas, l'application est connue et son modèle dédié d'évaluation analytique de la précision peut être utilisé. Dans une première partie, l'exploitation des résultats des modèles dédiés définis dans le second chapitre est mise en avant au travers de l'étude d'un générateur d'IPs. Les exemples des filtres LMS et IIR sont étudiés plus en détails. Dans le cadre d'un outils général de conversion automatique, l'application n'est pas connue a priori et l'utilisateur fournit le code source. Dans ce cas, le modèle général présenté dans le troisième chapitre est mis en œuvre. L'outil d'évaluation automatique de la précision inséré dans cette méthodologie de conversion en virgule fixe, est présenté. Les temps de calcul de la méthode générale sont évalués et comparés à une approche basée sur des simulations virgule fixe.

4.1 Génération de composants virtuels optimisés

Les modèles dédiés présentés dans le second chapitre sont utilisés pour l'optimisation de la spécification virgule fixe de l'application considérée. Ainsi, ce type de modèle s'intègre au sein d'un générateur de composants matériels/logiciels dédiés. En effet, pour réduire le temps de mise sur le marché des applications, une approche de développement des systèmes embarqués basée sur la réutilisation de blocs IPs (Intellectual Properties) est mise en œuvre. L'utilisateur assemble les différents composants virtuels pour réaliser l'architecture associée à l'application. Les blocs

matériels dédiés sont associés à des processeurs pour accélérer certains traitements de l'application dans le cadre de systèmes sur puces (SoC). Dans le cadre de la conception d'ASIC ou le développement d'applications sur FPGA, ces blocs IP spécifiques sont utilisés afin d'accélérer le développement. Les applications traitées sont connues et, de ce fait, leur modèle dédié peut être appliqué. Ceux-ci s'insèrent dans un générateur d'IPs détaillé par la suite. Le générateur proposé optimise le coût du composant tout en satisfaisant la contrainte de précision. Le but de cette approche est de fournir à l'utilisateur un niveau d'abstraction plus important en optimisant automatiquement les aspects arithmétiques. L'utilisation d'un modèle analytique permet d'explorer l'espace de conception virgule fixe et d'optimiser la structure de l'algorithme en des temps raisonnables. Les générateurs existants permettent de fixer la largeur des données mais n'autorisent pas à fournir uniquement une contrainte de précision. En effet, les différents outils existants utilisent soit des simulations en virgule fixe [6, 113], soit n'explorent pas l'ensemble de l'espace de recherche [58] ou encore nécessitent des temps de calcul relativement importants [32, 90, 91]. L'approche analytique choisie permet d'obtenir la solution optimisée dans des temps raisonnables. Dans un premier temps, la méthode générale est présentée. Le flot de génération des composants virtuels optimisés est détaillé. Dans un second temps, les différentes contraintes fournies par l'utilisateur pour la conversion en virgule fixe sont explicitées. Elles sont déterminées en fonction des objectifs souhaités par l'utilisateur. Les modèles d'architecture sont définis dans la troisième partie. La quatrième partie présente la méthode de calcul du coût de l'architecture. Pour illustrer les concepts définis, l'exemple de l'algorithme LMS est utilisé. Enfin, dans une dernière partie, des expérimentations effectuées sur des algorithmes IIR et LMS sont présentées.

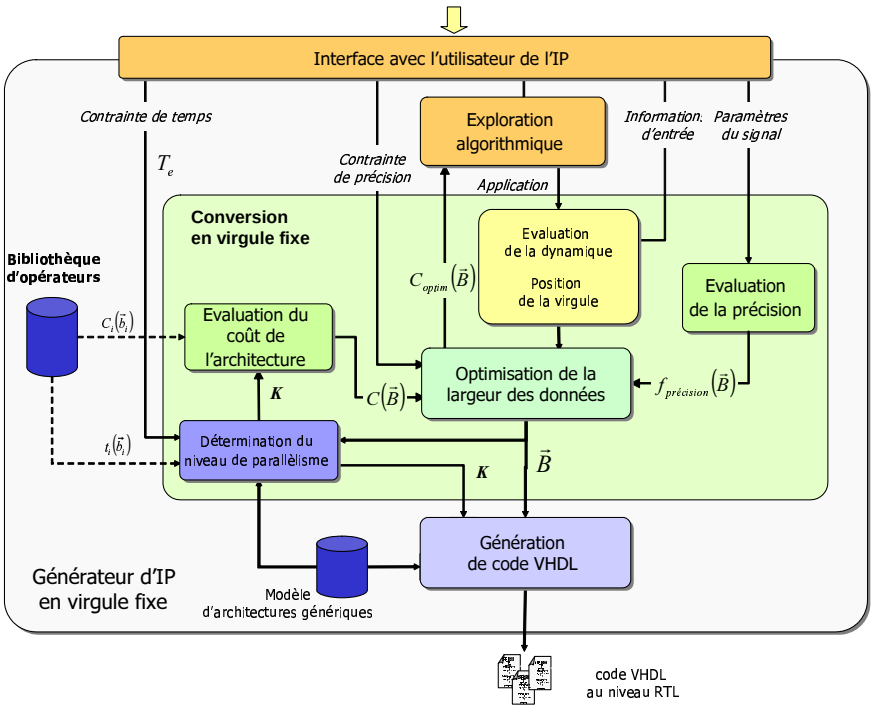

FIG. 4.1 – *Méthodologie de génération d'IPs en virgule fixe*

4.1.1 Flot de génération d'IP

Dans cette partie, la méthodologie de génération d'IP optimisée d'un point de vue arithmétique est présentée. L'objectif de notre générateur est de fournir un code VHDL au niveau RTL pour une IP dont le coût architectural est minimal. Ce coût correspond à la surface de l'architecture, la consommation d'énergie ou de puissance. Ce générateur d'IP, présenté sur la figure 4.1, est

composé de trois modules différents correspondant à l'exploration de l'algorithme, la conversion en virgule fixe et le back-end générant le code VHDL au niveau RTL.

L'objectif de la partie exploration de l'algorithme est de trouver la structure d'implantation minimisant le coût architectural pour une contrainte de précision donnée. Ce module teste les différentes structures d'une application donnée et choisit la meilleure en termes de coût. Pour chaque structure, le processus de conversion en virgule fixe recherche la spécification minimisant le coût $C(\overrightarrow{B})$ tout en satisfaisant une contrainte de précision où $\overrightarrow{B}$ représente le vecteur contenant les largeurs des opérandes d'entrée et de sortie des opérations. Le processus de conversion retourne le coût minimal $C_{min}(\overrightarrow{B})$ pour la structure choisie.

La partie principale du générateur d'IP correspond au processus de conversion en virgule fixe. Ce module doit explorer l'espace de recherche en virgule fixe pour trouver la spécification minimisant le coût architectural. La première partie correspond à la détermination de la dynamique des données. Ensuite, la position de la virgule est déduite de la dynamique de manière à assurer que toutes les données puissent être codées sans débordement. La troisième partie effectue l'optimisation de la largeur des données. L'algorithme d'optimisation a pour but de déterminer la largeur optimale des différentes données de façon à minimiser un coût architectural $C(\vec{B})$ (surface, energie, puissance) sous contrainte de précision comme exprimé par l'expression suivante :

$$\min(C(\vec{B})) \text{ avec } RSBQ(\vec{B}) \geq RSBQ_{min} \tag{4.1}$$

où $\vec{B}$ représente la largeur de toutes les données et $RSBQ_{min}$ la contrainte de précision. Le processus d'optimisation requiert l'évaluation du coût architectural $C(\vec{B})$ et de la contrainte de précision $RSBQ(\vec{B})$ définie au travers du Rapport Signal à Bruit de Quantification (RSBQ).

Le RSBQ de l'application est déterminé en utilisant son modèle dédié comme présenté dans le chapitre 2. Le coût de l'architecture dépend du niveau de parallèlisme au sein de l'application. Pour déterminer le niveau de parallèlisme K permettant de respecter la contrainte de temps, le temps d'exécution de l'architecture est évalué comme détaillé dans la section 4.1.2. Une fois les largeurs des différents opérateurs et le niveau de parallèlisme définis, le code VHDL représentant l'architecture au niveau RTL est généré.

Interface utilisateur

L'interface utilisateur permet de définir les paramètres de l'IP et les contraintes. L'utilisateur définit les différents paramètres associés à l'application. Par exemple, pour les filtres LTI numériques, l'utilisateur spécifie la fonction de transfert. Pour les filtres adaptatifs type LMS, l'utilisateur fournit la taille du filtre et le pas d'adaptation.

Pour une conversion en virgule fixe, l'évaluation de la dynamique ainsi que le calcul de la précision requièrent différentes informations sur le signal d'entrée. Par conséquent, l'utilisateur fournit la dynamique et des vecteurs de test des données d'entrée.

Pour générer l'architecture optimisée, l'utilisateur définit les contraintes de temps et de précision des calculs. La contrainte de temps détermine la fréquence des échantillons de sortie et est liée à la fréquence d'échantillonnage de l'application. Différentes contraintes de précision peuvent être considérées selon l'application. Par exemple, pour le LMS, le RSBQ en sortie est pris en compte. Pour les filtres LTI, trois contraintes sont définies correspondant à la déviation maximale de la réponse fréquentielle $|\Delta H_{max}(\omega)|$, à la valeur maximale du spectre pour le bruit de sortie $|B_{max}(\omega)|$ et au RSBQ minimal $RSBQ_{min}$.

Détermination de la position de la virgule

La position de la virgule est déterminée à partir de la dynamique des données. Plusieurs méthodes peuvent être utilisées pour évaluer la dynamique des données d'une application. Pour un système linéaire, la dynamique des données peut être calculée en utilisant les Normes L1 ou de Chebycheff. Ses normes permettent de déterminer les valeurs des dynamiques des différentes

données présentes dans le systèmes à partir de celles des données d'entrée et de la connaissance des différents coefficients du système. Soit $x(n)$ les données d'entrées et $y(n)$ celles de sortie. La norme $L1$ fournit la valeur maximale y_{max} de la sortie $y(n)$ définie par la relation suivante :

$$y_{max} = x_{max} \sum |h_i| \tag{4.2}$$

où x_{max} désigne la valeur maximale des données d'entrée $x(n)$ et $h(i)$ les termes de réponse impulsionnelle de la fonction de transfert $H(Z)$ entre l'entrée et la sortie. En utilisant la norme de Chebycheff, la valeur maximale de la sortie a pour valeur :

$$y_{max} = x_{max} max_w |H(w)| \tag{4.3}$$

Ces deux normes permettent de fournir la valeur maximale de la sortie $y(n)$ en fonction de la valeur maximale des données d'entrée $x(n)$. Ces méthodes sont appliquées à des systèmes pour lesquelles les fonctions de transfert existent. Ainsi, ces normes peuvent être utilisées uniquement pour des systèmes LTI. L'arithmétique d'intervalle peut être utilisée pour les systèmes non récursifs. Dans le cas où les systèmes sont non-LTI et présentant une récursion, une simulation en virgule flottante de l'application est nécessaire. Cette simulation permet de déterminer les différentes valeurs prises par les données à l'intérieur du système.

Les valeurs maximales des données présentes au sein du système sont alors déterminées. La position de la virgule dans les différents formats en est déduite. Le nombre de bits m_x pour le codage de la partie entière des données est déterminé à partir de leur valeur maximale permettant ainsi d'assurer l'absence de débordement dans le système. Sa valeur est donnée par la relation suivante :

$$m_x = \lceil log2(x_{max}) \rceil \tag{4.4}$$

Cette formule permet de définir l'ensemble des valeurs des nombres de bits nécessaires au codage des parties entières des données présentes dans le système. Les opérations de recadrage sont insérées dans l'application pour respecter les règles de l'arithmétique virgule fixe. Ainsi, les virgules des opérandes en entrée des additionneurs sont alignées.

4.1.2 Modèle d'architecture

Modèle d'architecture générique

Les performances de l'architecture dépendent de l'algorithme. Ainsi, un modèle d'architecture générique est défini pour chaque type de structures associées à l'algorithme ciblé. Ce modèle peut être configuré selon les paramètres fournis par l'utilisateur. Ce modèle d'architecture définit les unités de traitement et de contrôle, la mémoire et les interfaces d'entrée et de sortie. L'unité de traitement correspond à un ensemble d'opérateurs arithmétiques, de registres et de multiplexeurs interconnectés. Ces opérateurs et la mémoire sont extraits d'une librairie associée à la technologie utilisée. L'unité de contrôle génère les différents signaux de contrôle gérant l'unité de traitement, la mémoire et l'interface. Celle-ci est définie avec une machine d'états. Pour explorer l'espace de recherche en un temps raisonnable, des modèles analytiques sont utilisés pour évaluer le coût de l'architecture, sa latence ainsi que le niveau de parallèlisme.

Architecture LMS/DLMS Dans cette partie, l'architecture de l'IP LMS est détaillée. L'algorithme adaptatif LMS, présenté en section 2.1 page 36 est repris sur la figure 4.2. La mise à jour des coefficients se fait selon l'équation suivante :

$$W(n+1) = W(n) + \mu X(n) e(n-D) \quad \text{avec} \quad e(n) = y(n) - W(n)^t X(n) \tag{4.5}$$

où D correspond à un retard appliqué à l'erreur courante. Si ce retard est nul ($D = 0$), l'algorithme LMS est retrouvé. Sinon, l'algorithme Delayed-LMS (DLMS) est obtenu [66]. Au sein de

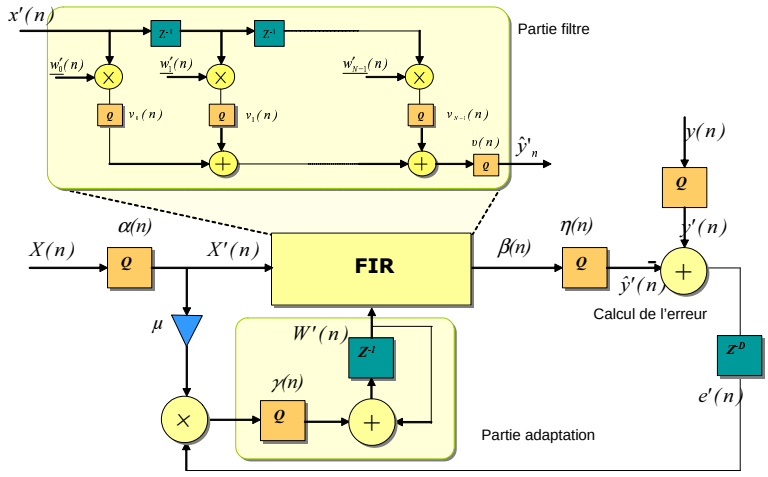

FIG. 4.2 – *Algorithme LMS/DLMS*

cet algorithme, l'erreur courante est retardée d'un certain nombre d'échantillons. Ceci permet de traiter séparément la mise à jour des coefficients et le calcul du filtre. Les parties mise à jour des coefficients et calcul du filtre peuvent être traitées en parallèle. Ainsi, la fréquence d'arrivée des échantillons en entrée peut être augmentée par rapport à l'algorithme LMS. En contrepartie, la convergence de cet algorithme est moins rapide. Le modèle de l'architecture, présenté sur la figure 4.3 est composé d'une partie filtrage et d'une partie adaptation. Pour respecter la contrainte de temps, ces deux parties doivent être parallélisées. Pour la partie filtre, K multiplications sont utilisées en parallèle et K MAD (Multiplications-Additions) pour la partie adaptation. Les différentes largeurs des données sont b_x pour les données d'entrée, b_m pour la sortie des multiplieurs du filtre, b_h pour les coefficients et b_e pour la sortie du filtre.

Pour accélérer le calcul, le processus est pipeliné et les opérateurs travaillent en parallèle. Soit T_{cycle} le temps de cycle correspondant à la période d'horloge. Celui-ci est égal à la valeur maximum entre le temps de latence d'un multplieur et d'un additionneur. La partie filtre est alors divisée en plusieurs étages pipelines. Le premier étage correspond à l'étape de multiplication. Pour additionner les différents résultats, un arbre d'addition est utilisé. Celui-ci est composé de $log_2(K)$ niveaux. Soit L_{ADD} le nombre d'additions pouvant être exécutées en un temps de cycle. Alors, le nombre d'étages pipelines pour l'addition globale est donné par la relation suivante :

$$M_{ADD} = \left\lceil \frac{log_2(K)}{L_{ADD}} \right\rceil \quad \text{avec} \quad L_{ADD} = \left\lceil \frac{T_{cycle}}{t_{ADD1}} \right\rceil \tag{4.6}$$

où t_{ADD1} est la latence d'un additionneur à deux entrées. Le dernier niveau de pipeline correspond à l'accumulation finale.

La partie adaptative est divisée en trois étages pipelines. Le premier désigne la soustraction entre le signal désiré et la sortie du filtre. Le deuxième est la multiplication entre l'erreur et le signal d'entrée, et l'addition finale entre le coefficient et son terme de mise à jour compose le dernier niveau.

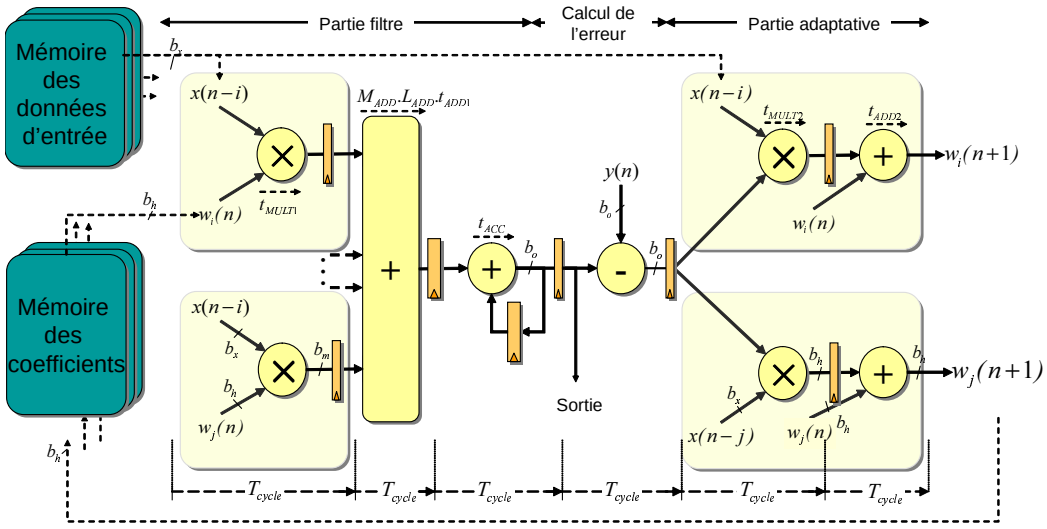

FIG. 4.3 – *Architecture générique de l'IP LMS/DLMS*

Détermination du niveau de parallélisme

Pour satisfaire la contrainte de temps spécifiée par l'utilisateur, plusieurs opérations doivent être exécutées en parallèle. Le niveau de parallélisme est déterminé tel que la latence du système soit inférieure à la contrainte de temps. Pour résoudre cette inégalité, la latence des opérateurs dépendant de la largeur des données doit être connue. La méthode à adopter est la suivante. Tout d'abord, la largeur des données est optimisée sans tenir compte du parallélisme. Les largeurs obtenues permettent d'établir la latence des opérateurs. Ensuite, le niveau de parallèlisme est calculé à partir de la contrainte de temps puis, les largeurs des données sont optimisées avec le vrai niveau de parallèlisme.

Contrainte temporelle Dans cette partie, les contraintes temporelles associées à l'architecture de l'IP LMS/DLMS sont détaillées. L'architecture est divisée en deux parties correspondant à la partie filtre et la partie adaptation. Le temps d'exécution de la partie filtre est donné par l'expression suivante :

$$T_{FIR} = \frac{N}{K} T_{cycle} + M_{ADD} T_{cycle} + T_{cycle} \tag{4.7}$$

Le temps d'exécution de la partie adaptative est donné par :

$$T_{Adapt} = T_{cycle} + \frac{N}{K}(T_{cycle}) + T_{cycle} \tag{4.8}$$

La contrainte de temps du système dépend de l'algorithme choisi. Pour le LMS, la contrainte de temps T_e doit satisfaire l'inégalité suivante :

$$T_{FIR} + T_{Adapt} < T_e \tag{4.9}$$

Le D-LMS possède un temps de convergence nettement plus lent que le LMS, cependant, les parties filtre et adaptation peuvent s'exécuter en parallèle puisque l'erreur prise en compte n'est

pas l'erreur courante mais une erreur retardée d'un délai D supérieur à une période d'échantillonnage. La contrainte devient alors :

$$T_{FIR} < T_e \quad \text{et} \quad T_{Adapt} < T_e \tag{4.10}$$

Le niveau de parallélisme est obtenu en résolvant les expressions (4.9) et (4.10).

Evaluation du coût de l'architecture

L'unité de traitement de l'IP est basée sur un ensemble d'opérateurs extraits d'une librairie. Cette dernière contient les opérateurs arithmétiques, les registres, les multiplexeurs et les mémoires pour les différentes largeurs de données possibles. Chaque élément de librairie l_i est automatiquement généré et caractérisé par sa surface Ar_i et sa consommation d'énergie En_i grâce à des scripts utilisant les outils Synopsys. La surface de l'IP (Ar_{IP}) est obtenue à partir de la somme de tous les éléments de base composant l'IP ainsi que la surface de la mémoire comme l'exprime l'équation (4.11). Soit IP_{archi} l'ensemble des éléments composants l'IP. La surface de l'architecture est donc égale à :

$$Ar_{IP}\left(\overrightarrow{B}\right) = \sum_{l_i \in IP_{archi}} Ar_i\left(b_i\right) \tag{4.11}$$

La consommation d'énergie de l'IP est la somme des consommations d'énergie de toutes les opérations effectuées comme l'exprime l'équation (4.12). Ces opérations inclues les opérations arithmétiques ainsi que les transferts de données entre l'unité de traitement et la mémoire. Soit IP_{ops}, l'ensemble des opérations exécutées pour calculer la sortie. La consommation d'énergie des différentes opérations En_j dépend de la largeur des données b_j.

$$En_{IP}\left(\overrightarrow{B}\right) = \sum_{l_j \in IP_{ops}} En_j\left(b_j\right) \tag{4.12}$$

La consommation d'énergie d'une opération est évaluée via des simulations avec l'outil Synopsys [109]. L'énergie moyenne est calculée à partir de l'énergie obtenue pour 10000 valeurs d'entrée aléatoires. Une fois l'expression du coût obtenu, cette dernière est optimisée via un algorithme d'optimisation [54] tout en respectant la contrainte de précision fournie.

4.1.3 Expérimentations et résultats

Des expérimentations ont été effectuées pour illustrer notre méthode. Pour ce faire, deux applications ont été étudiées. Il s'agit du filtre IIR d'ordre 8 et de l'algorithme LMS/DLMS à 128 coefficients. La librairie utilisée a été générée depuis une technologie CMOS $0.18\mu m$. Chaque élément de la librairie est automatiquement caractérisé en terme de surface et d'énergie en utilisant des scripts avec les outils Synopsys.

Filtre IIR

Dans cette partie, un filtre à Réponse Impulsionnelle Infinie est étudié. Soit N_{IIR} l'ordre du filtre. La fonction de transfert peut être factorisée afin d'implanter le filtre sous forme cascadée. Trois types de structures correspondant à la Forme Directe, le Forme Canonique Directe et la Forme Canonique Transposée [80] sont disponibles. L'ordre des cellules $Ncell$ peut évoluer entre deux et $\frac{N_{IIR}}{2}$ si N_{IIR} est pair où de deux à $\frac{N_{IIR}-1}{2}$ si N_{IIR} est impair. La complexité de ces différentes configurations est présentée dans le tableau 4.1 pour un filtre IIR d'ordre 8.

Pour une version cascadée du filtre IIR, la manière dont les différentes cellules sont organisées est importante. Ainsi, les différentes permutations des cellules doivent être testées. Pour des cellules d'ordre 4, trois différents couples de fonction de transfert peuvent être obtenus et pour chacun, 2 permutations sont possibles. Pour des cellules d'ordre 2, 24 permutations sont disponibles. Pour un filtre IIR d'ordre 8, les différents types de structures, les différents ordres des cellules,

Types de structure	Ordre des cellules	Nombre de cellules	Complexité du Filtre Addition	Multiplications	mises en mémoire	coefficients
Forme Directe	8	1	16	17	15	17
	4	2	16	18	15	18
	2	4	16	20	15	20
Forme Canonique Directe	8	1	16	17	12	17
	4	2	16	18	12	18
	2	4	16	20	12	20
Forme Canonique Transposée	8	1	16	17	12	17
	4	2	16	18	12	18
	2	4	16	20	12	20

TAB. 4.1 – *Complexité des différentes structures d'un filtre IIR d'ordre* 8

Ordre de la cellule	Largeur optimisée des coefficients
8	24
4	15
2	13

TAB. 4.2 – *Largeur des coefficients du filtre IIR*

les différentes factorisations et les différentes permutations doivent être testés. Cela conduit à 93 structures différentes pour une seule application. Ce nombre élevé de structures à tester montre l'intérêt d'une approche analytique pour obtenir des temps d'exécution raisonnables.

Optimisation de la largeur des coefficients Le processus d'optimisation pour un filtre IIR est effectué en deux étapes. Tout d'abord, la largeur des coefficients b_h est optimisée pour limiter la déviation de la réponse fréquentielle $|\Delta H(\omega)|$ due à la précision finie des coefficients comme l'exprime l'équation suivante :

$$min(b_h) \qquad \text{avec} \qquad |\Delta H(\omega)| \leq |\Delta H_{max}(\omega)| \tag{4.13}$$

La déviation maximale en fréquence $|\Delta H_{max}(\omega)|$ est définie de telle sorte que la réponse fréquentielle avec les coefficients en précision finie reste dans le gabarit. De plus, la stabilité du filtre est vérifiée avec les valeurs des coefficients en virgule fixe. Les résultats obtenus avec les versions cascadées et non-cascadées du filtre d'ordre 8 sont présentées dans le tableau 4.2. Pour une cellule d'ordre élevé, les coefficients ont une valeur plus grande, donc plus de bits sont nécessaires pour coder la partie entière. Ainsi, pour obtenir la même précision au niveau de la réponse fréquentielle, la largeur des coefficients doit être plus importante pour les cellules d'ordre élevé. Pour simplifier, la même largeur est adoptée pour tous les coefficients.

Optimisation de la largeur des signaux La seconde étape du processus d'optimisation correspond à l'optimisation de la largeur des signaux. Le but est de minimiser l'architecture sous contrainte de précision. Deux contraintes correspondant à la valeur maximale du spectre du bruit de quantification en sortie $|B_{max}(\omega)|$ et au RSBQ minimal $RSBQ_{min}$ sont prises en compte.

$$\min(C(\vec{B})) \qquad \text{avec} \qquad \begin{cases} RSBQ(\vec{B}) \geq RSBQ_{min} \\ |B(\omega)| \leq |B_{max}(\omega)| \end{cases} \tag{4.14}$$

La précision des calculs est évaluée à travers le RSBQ pour les 93 structures de façon à analyser les différences entre ces structures pour montrer l'influence de la structure d'implantation. La

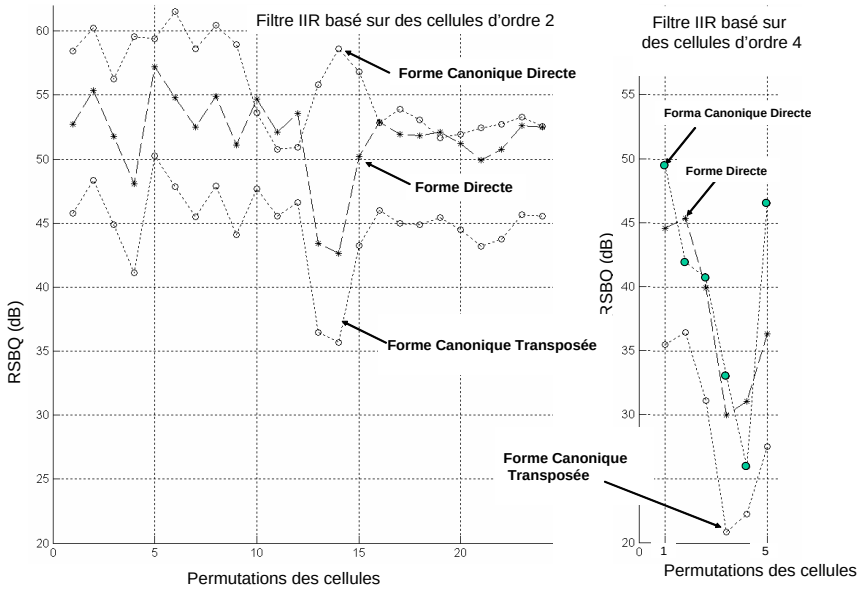

FIG. 4.4 – *Précision en virgule fixe en fonction de l'ordre des cellules et de leurs permutations*

précision est évaluée avec une implémentation classique basée sur des multiplications $16 \times 16 \rightarrow 32$-bits et des additions sur 32 bits. Pour le filtre non-cascadé, les bruits de quantification sont très importants et conduisent à un système non-stable. Les résultats sont présentés sur la figure 4.4 pour des filtres basés sur des cellules d'ordre 2 et 4.

Les résultats obtenus pour la Forme Canonique Transposée sont les mêmes que ceux obtenus pour la Forme Directe avec un offset. Celui-ci est égal à 7 dB pour le filtre basé sur des cellules d'ordre 2 et de 9 dB pour le filtre basé sur des cellules d'ordre 4. Ces deux types de filtres ont la même structure excepté le fait que les résultats des additions sont sauvegardés en mémoire pour la forme Transposée. Ces mémorisations ajoutent un bruit supplémentaire car la taille des données est réduite en mémoire.

Les résultats obtenus pour la Forme Directe et la Forme Canonique Directe indiquent qu'aucune des deux n'est meilleure. Les résultats dépendent de la permutation choisie. Dans le cas d'un filtre basé sur des cellules d'ordre 2, le RSBQ varie de 42 dB à 57 dB pour la Forme Directe et de 50 dB à 61 dB pour la Forme Canonique Directe. Pour les filtres basés sur les cellules d'ordre 4, le RSBQ varie de 30 dB à 45 dB pour la Forme Directe et de 26 dB à 49.5 dB pour la Forme Canonique Directe. Ainsi, le choix de la forme du filtre ne peut être fait initialement et toutes les structures doivent être testées.

La surface de l'architecture de l'IP et sa consommation d'énergie ont été optimisées pour différentes structures et pour des contraintes de précision de valeurs 40 dB et 90 dB. Les résultats sont présentés sur la figure 4.5 pour la consommation d'énergie et la figure 4.6 pour la surface de l'architecture. Pour souligner les variations de la surface de l'architecture dues aux changements de largeur des opérateurs, la contrainte de temps n'est pas prise en compte dans les expérimentations concernant le filtre IIR. Ainsi, le nombre d'opérateurs de l'architecture est identique pour chaque type de structure testée.

Comme montré sur la figure 4.4, les filtres basés sur des cellules d'ordre 4 aboutissent à une valeur de RSBQ plus faible que pour les structures basées sur des cellules d'ordre 2. Ainsi, ces filtres requièrent des opérateurs de plus grandes largeurs pour satisfaire la contrainte de précision. Ce phénomène augmente la surface de l'architecture de l'IP comme le montre la figure 4.6. Néanmoins, ces filtres nécessitent moins d'opérations pour le calcul de la sortie du système. Ceci réduit la consommation d'énergie comparé aux filtres basés sur des cellules d'ordre 2. La consommation d'énergie est plus importante pour les filtres composés de cellules d'ordre 4 comme le montre la

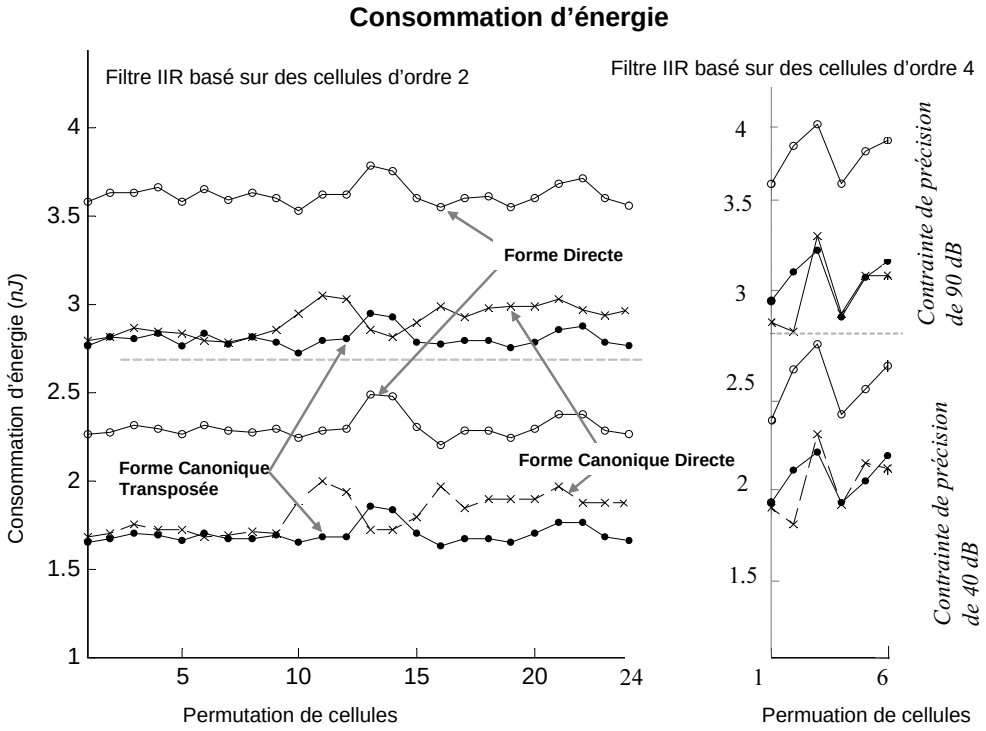

FIG. 4.5 – *Evolution de la consommation d'énergie en fonction des permutations et de l'ordre des cellules*

figure 4.5.

La consommation d'énergie est plus importante pour la Forme Directe car elle nécessite plus d'accès mémoire pour le calcul de la sortie du filtre. Pour les structures Canonique Directe et Canonique Transposée, les résultats sont proches. La meilleure solution est obtenue pour la Forme Canonique Transposée composée de cellules du 2^{nd} ordre et aboutit à une consommation d'énergie de 1.6 nJ pour une contrainte de précision de 40 dB et de 2.7 nJ pour une contrainte fixée à 90 dB. Comme le montre la figure 4.4, cette structure donne le plus faible RSBQ, donc, la largeur des opérateurs est plus grande que celle des autres structures. Néanmoins, cette forme consomme moins d'énergie car elle requiert moins d'accès mémoire que la Forme Canonique Directe. Pour la Forme Canonique Directe, les transferts en mémoire correspondent à la lecture des échantillons de signaux pour le calcul des produits avec les coefficients, et à l'écriture en mémoire de la mise à jour des échantillons retardés. Dans la Forme Canonique Transposée, les accès mémoire correspondent seulement à la mémorisation des sorties des additionneurs.

Par rapport à la meilleure solution, les autres structures basées sur des cellules du second ordre mènent à un surcoût maximal d'énergie de 36% pour une contrainte de précision de 40 dB et de 53% pour une contrainte de 90 dB. Pour les structures basées sur des cellules d'ordre 4, le surcoût maximum d'énergie est de 48% pour une contrainte de précision de 40 dB et de 71% pour une contrainte de 90 dB.
La surface de l'architecture est plus importante pour les filtres composés de cellules d'ordre 4. Comme expliqué précédemment, ces filtres conduisent à une valeur de RSBQ plus faible comparée à celles obtenues avec des structures composées de cellules du second ordre. Ainsi, elles requièrent des opérateurs avec une plus grande largeur pour satisfaire la contrainte de précision. La meilleure solution obtenue pour la Forme Canonique Directe avec des cellules du second ordre aboutit à une surface d'architecture de 0.3 mm^2 pour une contrainte de précision de 40 dB et à 0.12 mm^2 pour une contrainte de 90 dB. Comparées à la meilleure solution, les autres structures basées sur

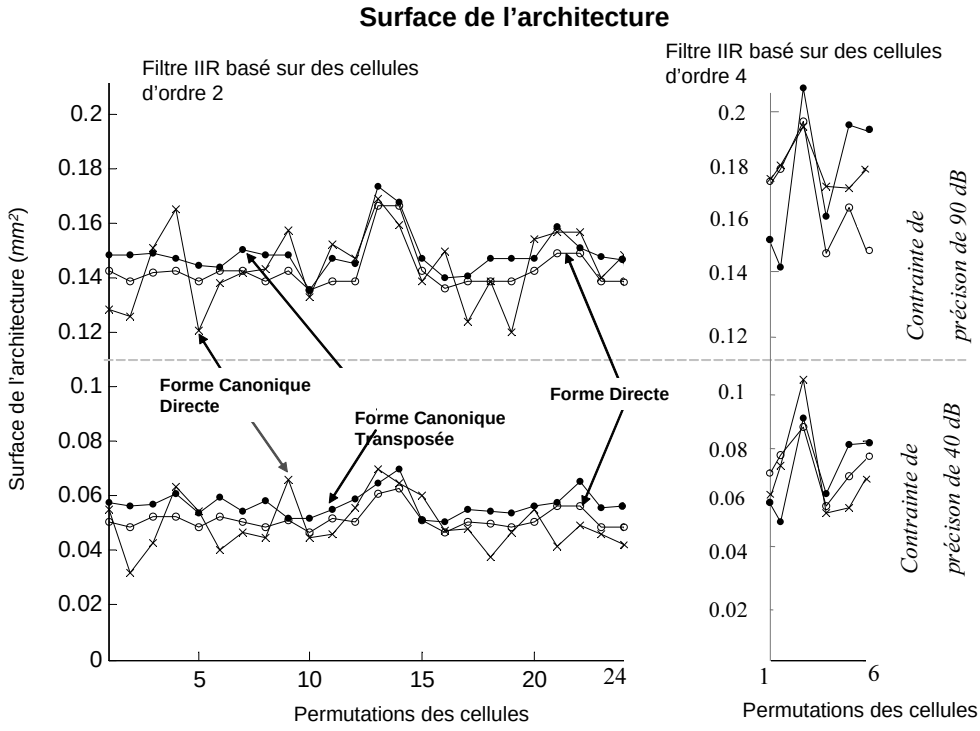

FIG. 4.6 – *Surface de l'architecture de l'IP en fonction des permutations des cellules et de leur ordre*

des cellules du second ordre mènent à un surcoût maximal de 100% pour une contrainte de 40 dB et de 40% pour une contrainte de 90 dB. Pour les structures composées de cellules d'ordre 4 le surcoût maximal est de 225% pour une contrainte de 40 dB et de 74% pour une contrainte de 90 dB.

La meilleure structure dépend du coût architectural défini. Les résultats sont différents pour la surface et la consommation d'énergie. Ces résultats soulignent les opportunités offertes par l'optimisation au niveau algorithme pour minimiser le coût de l'architecture et la nécessité de tester les différentes structures et de choisir la meilleure. En conséquence, des méthodes analytiques d'évaluation de la précision sont nécessaires pour avoir des temps d'optimisation raisonnables.

Pour une structure donnée, la surface de l'architecture de l'IP et la consommation d'énergie évoluent linéairement selon la contrainte de RSBQ. Entre les contraintes de 40 dB et 90 dB, l'énergie varie d'un facteur 1.68 et la surface d'un facteur 4. Ces résultats soulignent la nécessité de choisir la contrainte de précision adaptée de façon à ne pas gaspiller l'énergie et la surface.

Algorithmes LMS et DLMS

Les blocs IP LMS et DLMS sont utilisés dans différentes expérimentations pour souligner la nécessité d'optimiser les largeurs des opérateurs et des mémoires sous une contrainte de précision. Pour générer une architecture, la contrainte de temps T_e et la contrainte de précision $RSBQ_{min}$ doivent être définies.

Les blocs IP LMS et DLMS sont testés pour différentes valeurs de contraintes de temps T_e et de contraintes de précision $RSBQ_{min}$. Pour chaque valeur T_e et $RSBQ_{min}$, les largeurs des opérateurs et de la mémoire sont optimisées sous une contrainte de précision. Ensuite, l'architecture est générée. La surface de l'architecture, le niveau de parallèlisme et la consommation d'énergie sont mesurés et les résultats sont présentés respectivement sur les figures 4.7.a, 4.7.b et 4.7.c pour

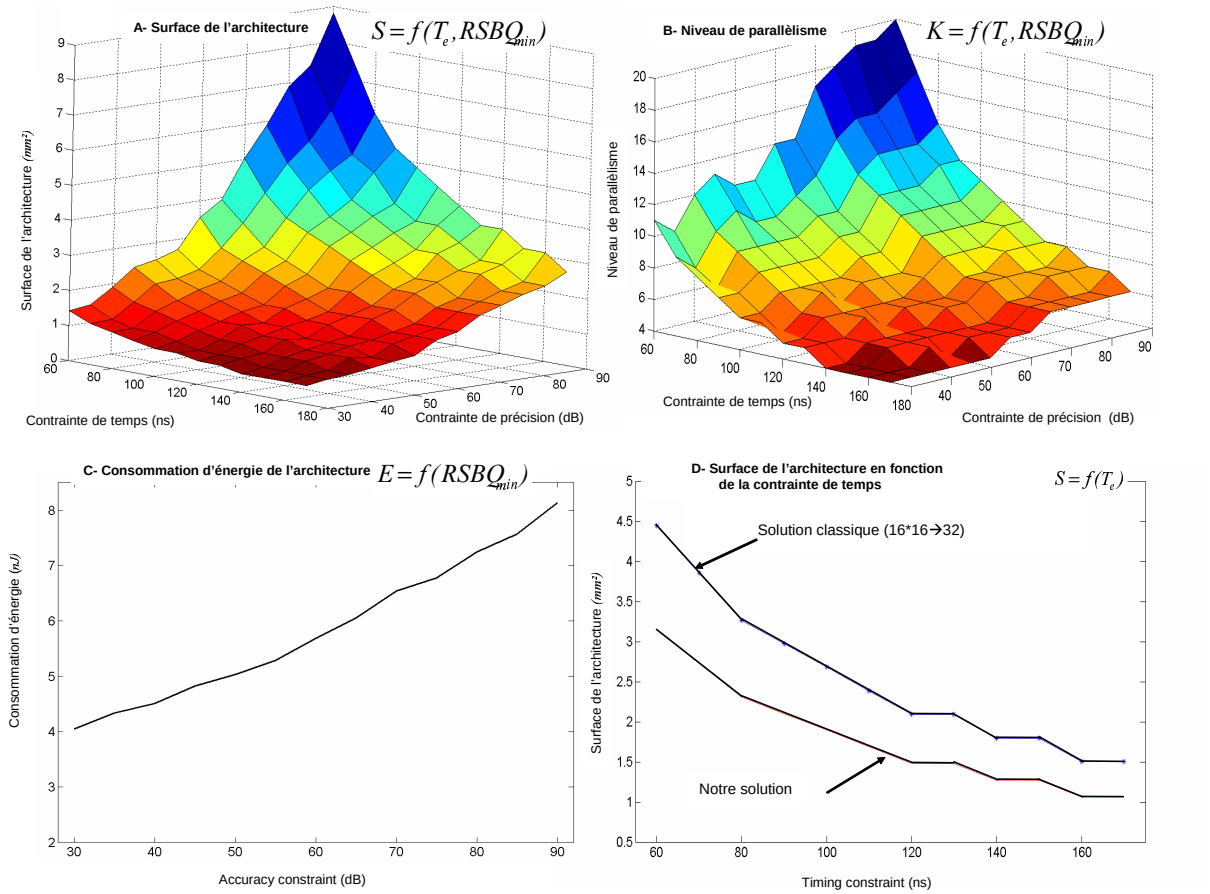

FIG. 4.7 – *Résultats d'expérimentations : surface de l'architecture, consommation d'énergie et niveau de parallèlisme pour différentes valeurs de contrainte de précision*

une contrainte de temps entre 60 ns et 170 ns et une contrainte de précision entre 30 dB et 90 dB. L'évolution de la consommation d'énergie selon la contrainte de précision est présentée sur la figure 4.7.c. Dans ce modèle, la consommation d'énergie est indépendante de la contrainte de temps et la puissance peut être estimée en divisant l'énergie par la latence. La consommation d'énergie varie de 4 nJ à 8.1 nJ pour une contrainte de précision allant de 30 dB à 90 dB. L'énergie est multipliée par deux entre ces deux valeurs de contrainte. Cette augmentation de la consommation d'énergie est seulement due à l'augmentation de la largeur des éléments de l'architecture. Pour satisfaire la contrainte de précision, la largeur des opérateurs doit être augmentée.
L'évolution de la surface l'architecture de l'IP selon les contraintes de précision et de latence est présentée sur les figures 4.7.a et 4.7.b. Pour des contraintes de précision de latence minimale, la surface de l'architecture est égale à 0.3 mm^2 avec un niveau de parallèlisme de $K = 4$. La surface de l'architecture monte à 9 mm^2 avec un niveau de parallèlisme de $K = 20$ pour les contraintes maximales. La surface augmente quand la contrainte de latence diminue. En effet, pour respecter cette contrainte, le niveau de parallèlisme K doit être plus important. Plus d'opérateurs sont nécessaires et, de ce fait, la surface de l'unité de traitement augmente. Des valeurs importantes de contrainte de précision nécessitent l'utilisation d'opérateurs possédant une plus grande largeur. Ainsi, la consommation d'énergie, la surface des unités de traitement et de mémoire augmentent tout comme la latence des opérateurs. Pour respecter la contrainte de temps, le niveau de parallèlisme K doit être augmenté et la surface de l'unité de traitement croît.
Nos résultats sont comparés à la solution classique basée sur des multiplications sur $16 \times 16 \rightarrow 32$-bits et des additions sur 32 bits. Cette solution fournit un RSBQ de 52 dB. Le coût est évalué pour notre approche avec une contrainte de précision de 52 dB avec différentes contraintes de temps. Les résultats sont présentés sur la figure 4.7.d. Pour la même qualité de calculs, notre approche

réduit le coût de l'architecture de 30% et la consommation d'énergie de 23%.

4.1.4 Conclusion

Dans cette partie, l'intérêt des modèles dédiés a été démontré en les intégrant dans un générateur d'IPs. Ce générateur minimise le coût de l'architecture de l'IP sous une ou plusieurs contraintes. Le niveau de parallèlisme de l'architecture est adapté aux contraintes de temps et de qualité. Le coût de l'architecture, et plus spécifiquement la qualité des calculs sont évalués analytiquement au moyen des modèles dédiés présentés dans le chapitre 2. Cette approche analytique permet de réduire de manière importante le temps d'évaluation. Cette technique permet d'explorer l'espace de recherche en virgule fixe et ainsi de trouver la spécification virgule fixe optimisant l'implémentation. De plus, cette approche analytique offre l'opportunité d'explorer l'espace de recherche au niveau algorithmique pour trouver la structure optimale. Les résultats présentés pour le filtre IIR d'ordre 8 soulignent l'intérêt d'une optimisation au niveau algorithmique. La meilleure structure peut réduire de manière significative la surface et la consommation en énergie de l'architecture de l'IP par rapport à une structure inadaptée. Pour le filtre LMS à 128 coefficients, la surface et la consommation d'énergie de l'architecture peuvent être réduite respectivement de 30% et 23% par rapport à la solution classique. Avec cette approche, l'utilisateur peut optimiser le compromis entre le coût de l'architecture, la précision et le temps d'exécution.

4.2 Méthode d'évaluation de la précision

Dans cette partie, la méthode d'évaluation automatique de la précision est présentée. Cette méthode se base sur le modèle d'évaluation analytique développé dans le chapitre 3. Cette approche s'intègre au flot général de conversion de virgule flottante en virgule fixe. Elle détermine automatiquement l'expression du RSBQ définissant la précision du système à partir de la description de celui-ci. La méthodologie se décompose en plusieurs parties telles que représentées à la figure 4.8. La représentation de l'application en entrée est définie par un Graphe Flot de Signal (GFS) S représentant le comportement de l'application spécifiée en virgule fixe. Cette représentation intègre les opérations de retard définissant le vieillissement des données. A chaque opération o_i du graphe est associée le type d'opération effectuée γ_i, les formats des entrées $(b_i^{e1},m_i^{e1},n_i^{e1})$, $(b_i^{e2},m_i^{e2},n_i^{e2})$, le format de la sortie (b_i^s,m_i^s,n_i^s) et le type de quantification Q_i utilisée en sortie de l'opération. Les termes $b_i = (b_i^{e1},b_i^{e2},b_i^s)$ et $m_i = (m_i^{e1},m_i^{e2},m_i^s)$ définissent la taille des données et la position de la virgule au niveau de chaque opération o_i. Soient B, m et Q les vecteurs définissant les paramètres b_i, m_i et Q_i des différentes opérations o_i. L'expression du RSBQ dépend donc de ces données. Ainsi, pour une application donnée, l'expression du RSBQ est déterminée automatiquement en fonction de B, m et Q sous la forme d'un code C décrivant le calcul du RSBQ.

La méthodologie se décompose en trois grandes parties. Tout d'abord, le GFS de données S est transformé en un GFS S' au niveau bruit incluant les bruits de quantification. Il permet de modéliser la manière dont les bruits se propagent au sein du système. Pour ce faire, chaque opération du système est modélisée au niveau bruit. Ensuite, par un parcours du GFS au niveau bruit S', les équations récurrentes du système définissant les relations entre les bruits d'entrée du système et les bruits de sortie sont déterminées. Ces dernières permettent de définir les pseudo fonctions de transfert modélisant les chemins parcourus par chaque source de bruit pour aboutir à une sortie. Ces deux premières parties utilisent la technique présentée dans [74]. Enfin, l'expression du RSBQ est calculée à partir des pseudo fonctions de transfert et des caractéristiques statistiques des sources de bruit. Cette dernière étape à été développée sous l'outil Matlab dans le cadre de ce travail de recherche .

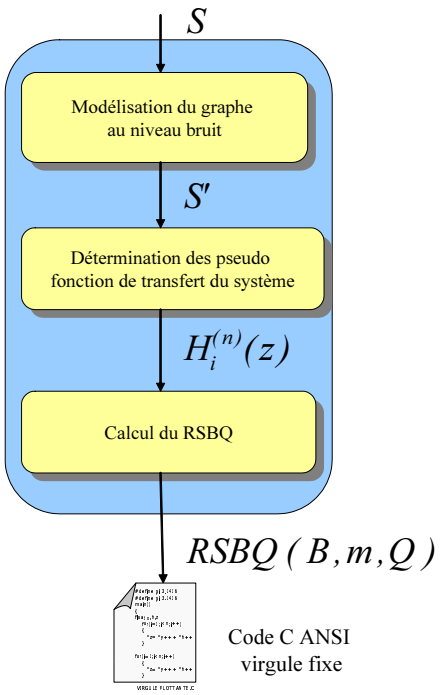

FIG. 4.8 – *Méthodologie d'évaluation de la précision*

4.2.1 Modélisation du système au niveau bruit

Le GFS noté $\mathcal{S}$ représentant le système en virgule fixe est modifié en un GFS noté $\mathcal{S}'$ modélisant le système au niveau bruit. Dans un premier temps, le graphe $\mathcal{S}$ est transformé en un graphe intermédiaire G_S incluant toutes les sources de bruit au moyen d'une transformation T_1. Ensuite, le graphe $\mathcal{S}'$ est obtenu en séparant la partie signal de la partie bruit. Le graphe $\mathcal{S}'$ modélise la propagation des différentes sources de bruit au sein du système.

La première étape désigne la représentation du graphe au niveau bruit. A chaque donnée d'entrée est associée une source de bruit. Ces caractéristiques statistiques dépendent de la provenance du signal d'entrée. Si celui-ci vient de la sortie d'un autre système, les statistiques du bruit associé sont, soit définies par notre modèle appliqué précédemment au système d'où proviennent les données, soit spécifiées par l'utilisateur. Sinon, les caractéristiques du bruits sont définies par les modèles classiques des bruits de quantification en fonction en fonction du nombre de bits éliminés, du format des données et de la loi de quantification utilisée (troncature, arrondi, ...).

Ensuite, chaque opération de la représentation intermédiaire est modélisée par ses formats comme le montre la figure 4.9. L'opération o_k possède deux entrées aux formats $(b_{e_1},m_{e_1},n_{e_1})$ et $(b_{e_2},m_{e_2},n_{e_2})$ et une sortie au format (b_s,m_s,n_s). Cette opération génère en sortie un bruit b_k défini par les différents formats de l'opération. Les paramètres statistiques de ces sources de bruit sont déterminés par le nombre de bits k éliminés, le nombre de bits n réservés au codage de la partie fractionnaire des données et de la loi de quantification Q. Les statistiques de ces bruits sont définies dans le tableau 1.1 page 20. Ainsi, la représentation intermédiaire G_S est obtenue. Elle représente l'application au sein de laquelle toutes les sources de bruit sont insérées. Une transformation de ce graphe peut être réalisée afin de réduire le nombre de sources de bruit en déplaçant et regroupant celles-ci.

La seconde phase de la transformation T_1 conduit à la modification du graphe G_S de façon à modéliser chaque opération o_i par son modèle de propagation au niveau bruit. Dans la section 3.1.2, les différentes opérations arithmétiques présentes dans les systèmes sont définies au niveau bruit. Cette modélisation permet d'exprimer la manière dont les bruits se propagent au sein de

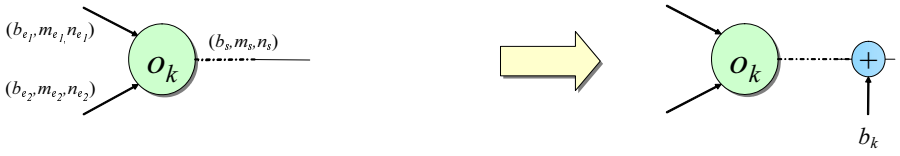

FIG. 4.9 – *Modèle d'insertion des bruits*

chaque opération présente dans le système. Ces opérations présentes dans le graphe G_S sont alors modifiées en les remplaçant par leur modélisation au niveau bruit. Chaque opération est séparée en une partie signal et une partie bruit. Les modèles utilisés pour les délais, la multiplication et l'addition sont définis dans la section 3.1.2.

Ainsi, Les entrées du graphe $\mathcal{S}'$ correspondent aux bruits générés au sein du système ainsi que les bruits associés aux données d'entrée. Les sorties de ce graphe S' sont définies par les bruits associés aux sorties du système initial $\mathcal{S}$. Ces bruits de sortie sont générés par la propagation des sources de bruit au sein du système.

4.2.2 Détermination des pseudo fonctions de transfert

La technique utilisée pour la détermination des pseudo fonctions de transfert est issue de la thèse de D. Menard [74]. Cette seconde étape de l'évaluation automatique des systèmes en virgule fixe notée T_2, correspond à la détermination des pseudo fonctions de transfert de chaque source de bruit vers une sortie du système $\mathcal{S}'$. Ces fonctions sont construites en parcourant le graphe des entrées vers les sorties. Mais cette technique n'est pas utilisable si des cycles sont présents dans le graphe comme dans le cas des structures récursives. Pour ces cas, le graphe est décomposé en plusieurs graphes acycliques. Les différentes étapes de T_2 sont illustrées sur la figure 4.10 et détaillées dans les paragraphes suivants.

Transformation en graphes acycliques ($T_{2.1}$): Le rôle de la transformation $T_{2.1}$ est de décomposer le graphe en plusieurs graphes acycliques dirigés si des circuits sont présents. La première étape consiste à détecter l'occurence d'un circuit dans le graphe pour décider si la transformation $T_{2.1}$ doit avoir lieu. Un algorithme de recherche de circuit basé sur une traversée en profondeur est utilisé. Cet algorithme permet de ne traiter chaque nœud qu'une seule fois. Lorsqu'un circuit est détecté, le ou les points de coupe du circuit doivent être déterminés pour séparer le graphe en portions acycliques. Le ou les points choisis sont ceux permettant de quitter le circuit vers une sortie du système lors du parcours du graphe orienté.

Expression des fonctions des sous-graphes ($T_{2.2}$): Les points de coupe p de l'étape précédente sont dupliqués en un nœud d'entrée p' et un nœud de sortie p. Un opérateur de retard doit se trouver entre p' et p sinon le système est en régime continu. Le graphe démantelé précédemment en graphes acycliques est transformé en un graphe équivalent G_{eq} spécifiant l'algorithme par un ensemble de fonctions. Les points de ce graphe équivalent sont les entrées et sorties de chaque graphe acyclique précédent et les arcs désignent des expressions pour passer du point source au point destination. La notation $f_{p_i p_j}$ représente la fonction permettant d'aller au nœud p_i depuis le nœud p_j. Des substitutions de variables sont effectuées pour conserver uniquement des circuits unitaires, c'est à dire ne partageant qu'un seul point avec le reste du graphe. Ainsi $f_{p_1 p_0}$ suivi de $f_{p_0 p_1}$ représentant le cycle p_0,p_1,p_0 est transformé en $f_{p_0 p_0}$. Ceci est la condition nécessaire pour pouvoir obtenir les pseudo fonctions de transfert par une simple transformée en Z.

Pseudo fonctions de transfert partielles ($T_{2.3}$): Du graphe G_{eq} est obtenu un graphe dirigé G_{H_i}, qui spécifie l'algorithme avec un ensemble de pseudo fonctions de transfert. Les nœuds sont

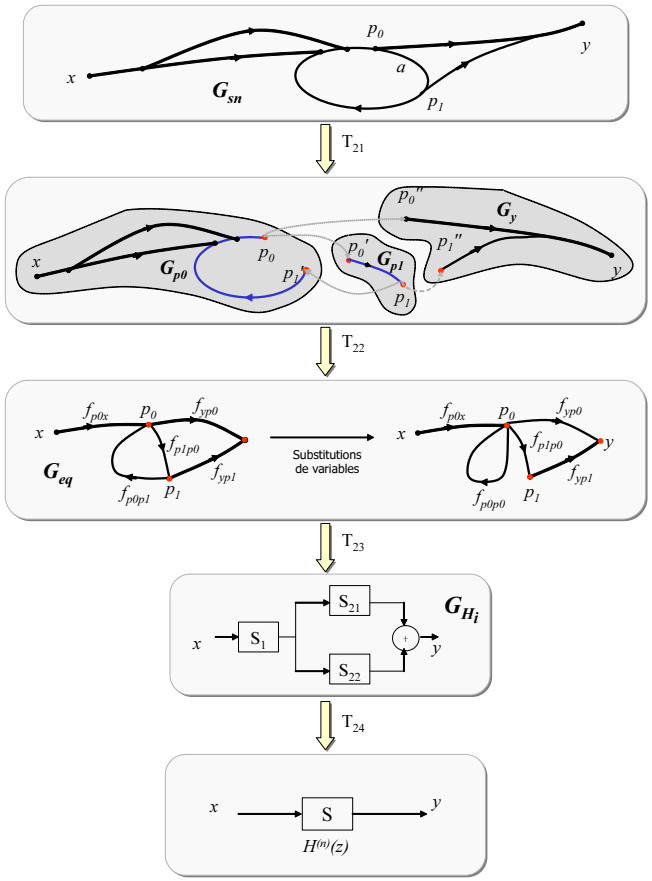

FIG. 4.10 – *Transformations* T_2

les mêmes que ceux du graphe G_{eq} et les arcs sont attachés à une pseudo fonction de transfert intermédiaire entre la source et la destination de l'arc orienté. Les circuits unitaires précédents n'apparaissent plus dans ce graphe, mais les autres arcs sont identiques. Les circuits unitaires se retrouvent dans les dénominateurs des pseudo fonctions de transfert. Les pseudo fonctions de transfert des sous-systèmes peuvent être calculées directement à partir des équations récurrentes de chaque sous-système.

Pseudo fonction de transfert ($T_{2.4}$): La pseudo fonction de transfert d'un point source à un point destination s'exprime en fonction des pseudo fonctions de transfert partielles situées sur le même chemin. Les chemins partiels parcourus étant en série ou en parallèle, la pseudo fonction de transfert finale est donc calculée en utilisant les méthodes de composition de systèmes présentées dans la section 3.2.2. Elle s'exprime sous la forme suivante :

$$H^{(n)}(z) = \frac{\sum_{l=0}^{Q} g_{(n-l)}^{(n)} z^{-l}}{1 - \sum_{i=1}^{P} f_{(n-i)}^{(n)} z^{-i}} \tag{4.15}$$

Au sein de cette expression, les termes $g_{(n-l)}^{(n)}$ et $f_{(n-i)}^{(n)}$ sont définis comme le produit de termes matriciels **A** et **D** pouvant évoluer au cours du temps. Cette modélisation est ensuite utilisée dans le calcul du RSBQ de l'application considérée.

4.2.3 Calcul de l'expression du RSBQ

La dernière étape de l'évaluation de la précision concerne le calcul de l'expression du RSBQ. Ce dernier niveau s'effectue grâce à l'outil Matlab et a été développé dans le cadre de ce travail de recherche. Pour ce faire, le calcul est mis en œuvre à partir des pseudo fonctions de transfert modélisant la propagation des sources de bruit dans le système et de leurs caractéristiques statistiques. Ensuite, l'outil implanté sous Matlab calcule les caractéristiques statistiques des des termes Ka et Km de l'équation (3.134) présentée à la page 110. L'autocorrélation, la covariance et les autres termes statistiques sont déterminés. A partir de l'autocorrélation du bruit de sortie, l'expression de la puissance est déduite. D'autre part, une simulation de l'application en virgule flottante permet de déterminer la puissance du signal, nécessaire pour calculer le RSBQ de l'application. Le code Matlab est divisé en 3 grandes parties comme le montre la figure 4.11.

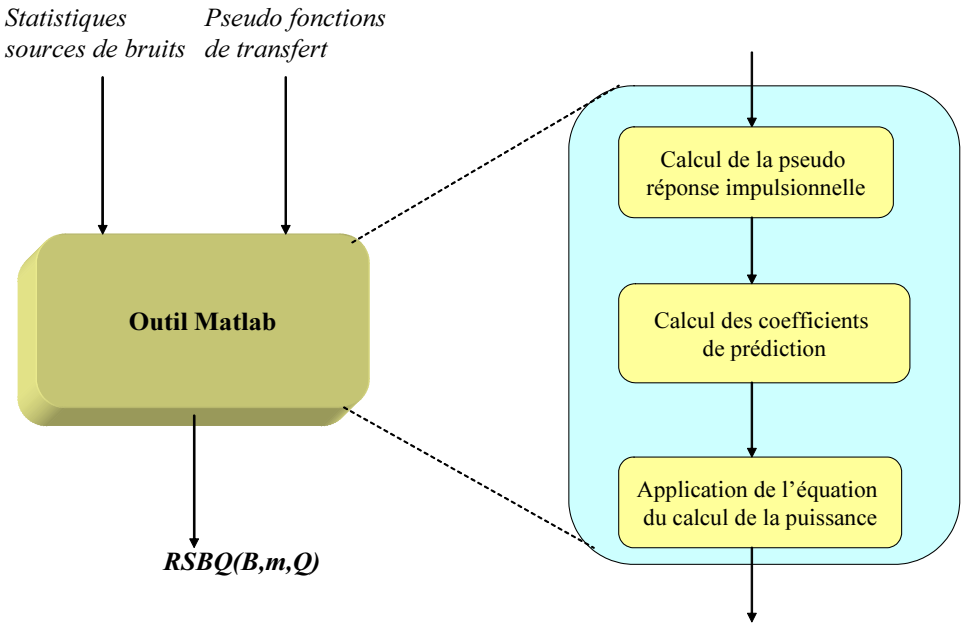

FIG. 4.11 – *Synoptique de l'outil de calcul du RSBQ*

Calcul de la pseudo réponse impulsionnelle

La première étape consiste à calculer les pseudo réponses impulsionnelles $h^{(n)}$ de chaque source de bruit. Ces termes permettent d'exprimer l'influence du bruit d'entrée et de ses échantillons précédents sur le bruit de sortie. Elle s'exprime selon la relation suivante :

$$y(n) = \sum_{k=0}^{n} h_{(k)}^{(n)} b(n-k) \tag{4.16}$$

Chacun de ces termes $h_{(k)}^{(n)}$ correspond à la multiplication par des termes matriciels $\mathbf{A}$ et $\mathbf{D}$ (ou/et $\mathbf{U}$ et $\mathbf{V}$ si le terme de bruit est transposé). Seuls les $(P+Q)$ premiers termes sont déterminés à partir de la pseudo fonction de transfert en utilisant la méthode de calcul de la pseudo réponse impulsionnelle définie dans la section 3.2.1. Les valeurs de P et Q désignant les ordres du numérateur et du dénominateur de la pseudo fonction de transfert $H^{(n)}(z)$.
Les termes suivants ne sont pas calculés. En effet, l'approche par prédiction linéaire permet de s'affranchir de cette étape. Pour chaque source de bruit $b_i(n)$, les premiers termes de sa pseudo réponse impulsionnelle sont ainsi obtenus.

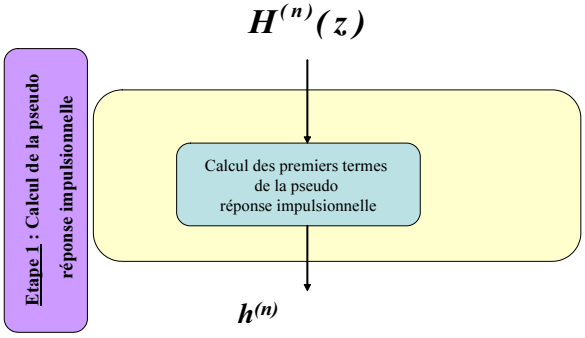

FIG. 4.12 – *Première étape de l'outil d'évaluation de la précision*

Calcul des coefficients de prédiction linéaire

Dans la deuxième étape, les coefficients de prédiction modélisant la propagation du bruit dans le système sont calculés à partir des premiers termes de la pseudo réponse impulsionnelle et de la pseudo fonction de transfert comme expliqué dans la section 3.3.3 (figure 4.13). La matrice **S** et le vecteur J nécessaires au calcul de ces coefficients sont tout d'abord établis. Pour ce faire, les $P+Q$ premiers termes de la pseudo réponse impulsionnelle sont utilisés ainsi que les termes du dénominateur de la pseudo fonction de transfert. Ces coefficients λ_{i_j} sont obtenus en inversant la matrice **S**. Cette inversion est effectuée sur l'outil Matlab facilitant les calculs matriciels. Ensuite, si les termes **A** et **D** sont présents ensemble au sein de la pseudo réponse impulsionnelle, les coefficients de prédiction sont calculés en multipliant ceux obtenus pour les termes en **A** et ceux obtenus pour les termes **D**. Ainsi, pour chacune des sources de bruit $b_i(n)$, les coefficients de prédiction sont déterminés. Ils permettent de réduire la complexité lors du calcul du RSBQ de l'application en supprimant les sommes infinies présentes dans l'expression.

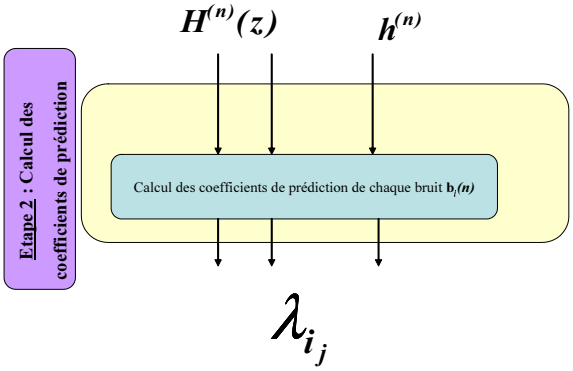

FIG. 4.13 – *Deuxième étape de l'outil d'évaluation de la précision*

Calcul de l'expression de la puissance du bruit

Finalement, la dernière étape consiste au calcul de l'expression du RSBQ. Pour ce faire, une simulation virgule flottante détermine la puissance du signal. La puissance du bruit s'exprime selon l'equation (3.127) dans laquelle sont insérés la pseudo réponse impulsionnelle ainsi que les coefficients de prédiction permettant de diminuer la complexité des calculs. Ces coefficients de prédiction permettent de remplacer les sommes infinies au moyen de l'expression (3.200). Les termes multipliant les statistiques des bruits d'entrée sont alors obtenus. En effet, pour des bruits

d'entrée blancs, représentant la quasi totalité des cas, la puissance du bruit de sortie s'exprime selon la relation suivante :

$$P_{b_y} = \sum_{i=1}^{Ne} \sigma_{b_i}^2 Ka_i + \sum_{i=1}^{Ne} \sum_{j=1}^{Ne} \mu_{b_i} \mu_{bj} Km_{ij} \tag{4.17}$$

Les coefficients de prédiction permettent de déterminer les valeurs des constantes Ka et Km dans des temps relativement faibles. Ainsi, l'expression analytique de la puissance du bruit est obtenue sous la forme d'un code C au sein de laquelle les termes Ka et Km ont été calculés. Ensuite, au sein du processus d'optimisation de la largeurs des données, l'expression précédente est évaluée pour différentes valeurs de μ et σ fixées par la largeur des données. Le temps requis est alors nettement moins important comparé à celui d'une approche par simulation. Si les bruits d'entrée ne sont pas des bruits de quantification, l'expression (3.127) est reprise. Cette dernière est plus complexe car des termes de covariance des bruits d'entrée apparaissent. Néanmoins, le temps requis est du même ordre.

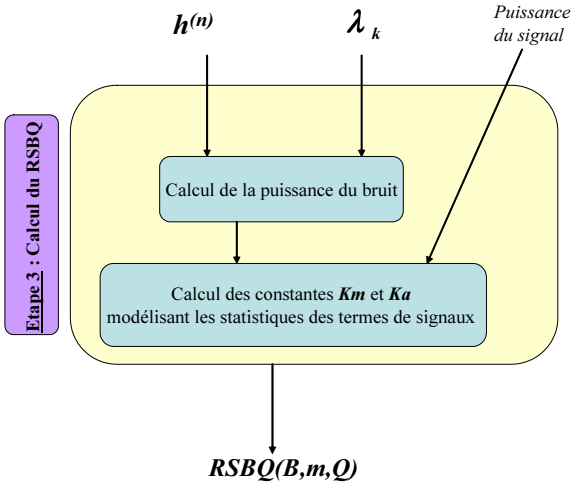

FIG. 4.14 – *Troisième étape de l'outil d'évaluation de la précision*

Conclusion

Dans cette partie, la méthodologie d'évaluation automatique de la précision des systèmes en virgule fixe a été présentée. Elle génère l'expression du RSBQ de l'application considérée à partir du graphe définissant le système en virgule fixe. Le graphe est déterminé au niveau bruit en y insérant les différentes sources puis les pseudo fonctions de transfert sont calculées pour chaque source de bruit. Enfin, le RSBQ est calculé en utilisant les pseudo réponses impulsionnelles déduites des pseudo fonctions de transfert.

4.2.4 Evaluation des temps de calcul de la méthodologie

Dans cette partie, les temps de calcul de la méthodologie générale sont évalués et plus particulièrement le temps d'exécution de la partie finale de l'outil. Tout d'abord, le temps requis pour le calcul des termes statistiques présents dans l'expression de la puissance du bruit est mesuré. Dans un second temps, notre approche analytique est comparée à une approche basée sur des simulations virgule fixe en terme de temps d'optimisation.

Temps de calcul

Le temps d'obtention de l'expression du RSBQ de l'application est mesuré dans ce paragraphe. Le temps de détermination des termes Ka et Km de l'equation (3.134) correspond au temps d'exécution de l'outil d'évaluation automatique de la précision en virgule fixe. Pour estimer le temps d'exécution de l'outil sur le filtre LMS, seule la partie finale est prise en compte car le temps requis pour cette partie est plus important que celui de la partie frontale. Une fois l'expression du RSBQ obtenue, celle-ci est évaluée à chaque itération du processus d'optimisation pour des formats de données spécifiés. Ce temps d'évaluation de l'expression correspond à l'application d'une relation mathématique dont le temps de calcul est négligeable.

Tout d'abord, le temps de calcul lié au LMS est présenté. Considérons un algorithme LMS de taille $N = 16$. Le temps requis pour calculer les termes statistiques de la méthode directe pour $p = 500$ et $q = 100$ est de 84 secondes. Ce temps est relativement faible mais pour des valeurs de N plus élevées, il peut devenir relativement important. La figure 4.15 montre l'évolution du temps de calcul en fonction de la taille du filtre avec les valeurs de p et q fixées précédemment. Alors que pour $N = 16$ le temps de calculs est de 84 secondes, celui-ci est de l'ordre de 21000 secondes pour $N = 128$ soit environ 5 heures. L'augmentation du temps de calcul est bien de l'ordre de N^2 comme calculé dans la partie complexité 3.3.2.

Pour réduire ces temps de calcul, la méthode d'approche par prédiction linéaire a été introduite. Pour un filtre de taille 128, le temps de calcul est de seulement 4 minutes. Le gain est proche d'un facteur 100. La figure 4.16 montre l'évolution du temps requis pour le calcul de la méthode par prédiction en fonction de la taille du filtre. Ces temps de calculs sont du même ordre que ceux obtenus pour les modèles dédiés aux applications. Le temps de calcul nécessaire avec la méthode basée sur les coefficients de prédiction est de 3.43 secondes pour $N = 16$. Ainsi, la méthode utilisant les coefficients de prédiction permet de réduire la complexité de manière importante sans pour autant affecter la qualité de l'estimation.

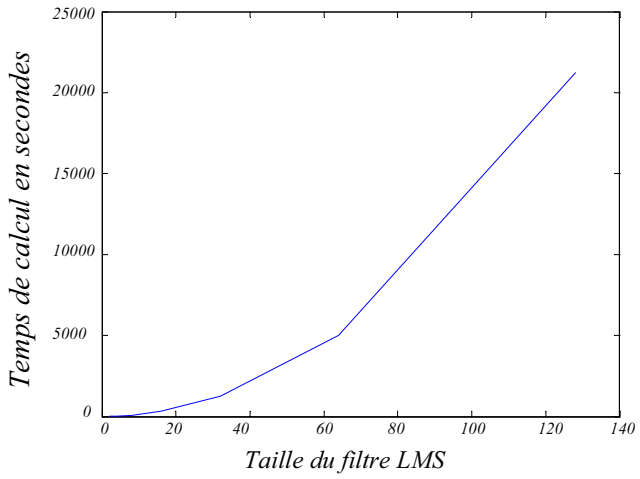

FIG. 4.15 – *Temps de calcul en fonction de la taille du filtre LMS*

Traitons maintenant le cas des filtres adaptatifs basés sur les algorithmes APA. La figure 4.17 montre le temps de calcul du modèle général appliqué aux APA pour des tailles de N allant de 2 à 10 et K variant de 1 à $N - 1$. Le temps de calcul varie de 10 secondes à 233 secondes (soit 4 minutes). Pour mieux voir l'évolution du temps de calcul, la figure 4.18 représente le temps de calcul pour la valeur de N fixée à 10 et K variant de 1 à 9. La valeur du temps de calcul varie de 84 à 232 secondes entre ces valeurs. Cependant, la valeur du temps de calcul semble linéaire

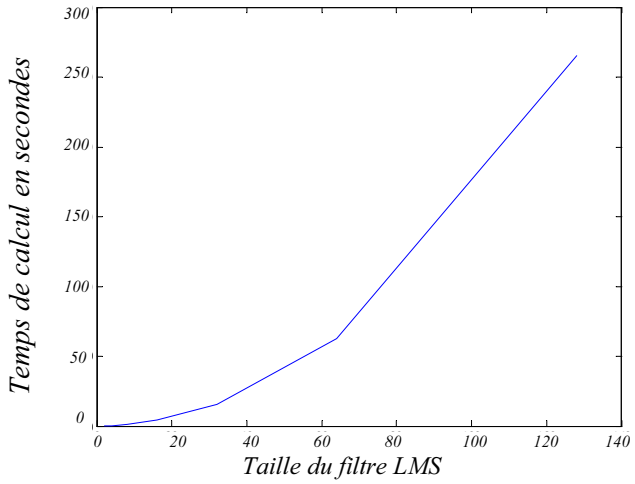

FIG. 4.16 – *Temps de calcul du filtre LMS avec la prédiction linéaire*

en fonction de K. En reprenant l'expression (3.160), page 114 du calcul de la complexité, celle-ci s'exprime en $\mathcal{O}(K)$ pour une valeur de N fixée. Ainsi pour une valeur de K multipliée par 9, le temps de calcul est multiplié par 3. Considérons maintenant, une valeur de K fixé à 1 et N varie de 2 à 10. La figure 4.19 représente les temps de calcul obtenus. Ils varient de 10 à 84 secondes. Comme précédemment, la croissance est, en théorie, linéaire. Pour une valeur de N multipliée par 5, le temps de calcul est multiplié par 8. Pour terminer, regardons comment évolue le temps de calcul pour des valeurs de K et N évoluants conjointement. La valeur de N varie de 2 à 10 et celle de K est égale à $N-1$. La figure 4.20 illustre les résultats obtenus. Le temps de calcul varie de 10 secondes à plus de 250 secondes. La croissance est plus importante que dans le cas précédent. En théorie, la croissance du temps de calcul est en $\mathcal{O}(N^2)$ pour ce cas présent. Le temps de calcul évolue de la même façon selon les différentes tailles du système. En effet, en regardant plus en détails les résultats obtenus, pour une valeur de $N+K$ constante, le temps de calcul est quasi constant. Les temps de calcul obtenus varient de 10 secondes à plus de 250 secondes. Pour réduire ces temps, l'approche par approximation linéaire peut être utilisée. Elle permet de diminuer le temps de calcul par 100 en moyenne. Pour les mêmes expérimentations, le temps de calcul varie de 0.04 à 1.26 secondes. La valeur maximale du gain obtenue est même de l'ordre de 200. Ce cas est obtenu pour les valeurs de $K=7$ et $N=9$. Les temps requis sont alors très faibles. En pratique, ces algorithmes sont utilisés avec un ordre de projection $K=12$ au maximum. De ce fait, le temps de calcul par la méthode par prédiction est présentée sur la figure 4.21 pour des tailles de filtres variant de 16 à 128. Le temps requis oscille alors entre 9 secondes pour $N=16$ et 310 secondes (environ 5 min) pour $N=128$. Les temps obtenus sont alors très faibles et raisonnables dans le cadre de la conception de systèmes embarqués.

Comparaison avec les méthodes basées sur la simulation

Dans cette section, les méthodes analytiques proposées sont comparées aux méthodes par simulation. Ces deux approches distinctes sont utilisées pour évaluer la précision d'un système lors du processus de conversion en virgule fixe. Cette conversion en virgule fixe s'effectue en optimisant un coût (énergie, puissance, surface de l'architecture) comme le montre l'expression suivante :

$$\min(C(\vec{B})) \text{ avec } RSBQ(\vec{B}) \geq RSBQ_{min} \tag{4.18}$$

où B désigne le vecteur contenant les largeurs des opérateurs. A chaque itération du processus d'optimisation, la précision de la spécification virgule fixe doit être évaluée. Ainsi, dans ce souci

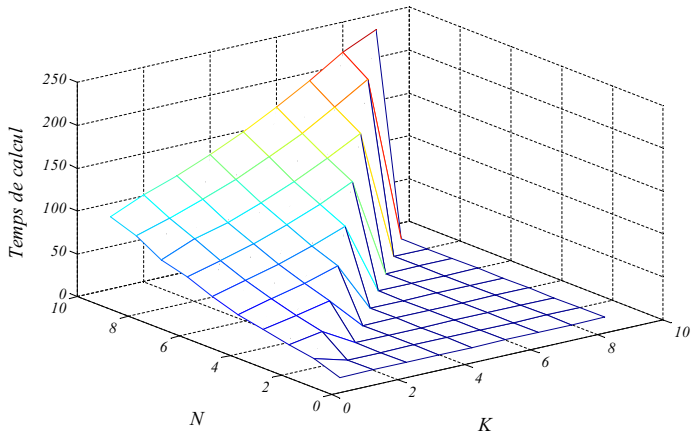

FIG. 4.17 – *Temps de calcul en fonction de la taille du filtre pour les APA*

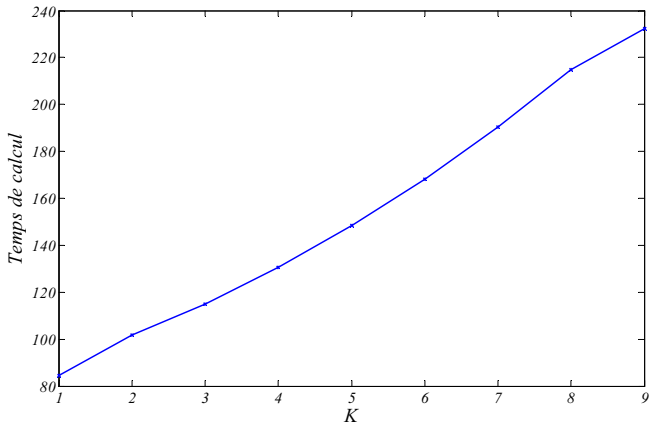

FIG. 4.18 – *Temps de calcul en fonction de la valeur de K pour les APA*

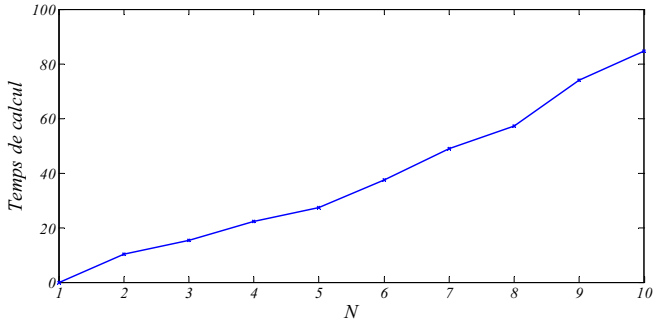

FIG. 4.19 – *Temps de calcul en fonction de la valeur de N pour les APA*

de gain de temps, notre approche est comparée à une approche par simulation en terme de temps d'optimisation. Pour ce faire, notre approche est estimée via l'outil Matlab. De la même façon, les approches basées sur des simulations virgule fixes sont évaluées au moyen de simulations Matlab. Pour obtenir des résultats cohérents, le temps de notre outil complet (parties frontale et finale) d'évaluation de la précision décrit dans le chapitre 4 est pris en considération. Pour des systèmes LTI, notre approche permet des gains de temps dès les premières itérations du processus d'optimisation. Ainsi, pour un filtre FIR de taille 16, le temps est moindre après seulement 3 itérations et après 14 itérations pour un IIR8.

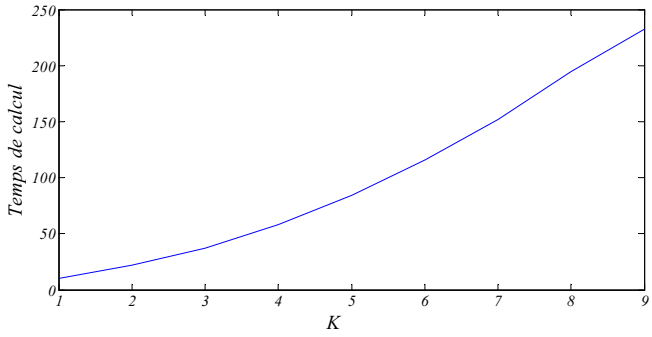

FIG. 4.20 – *Temps de calcul en fonction de la valeur de K et de $N = K + 1$ pour les APA*

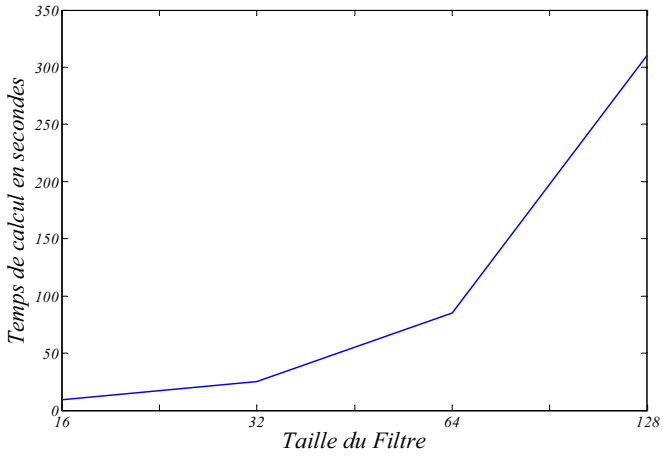

FIG. 4.21 – *Temps de calcul de la méthode par prédiction linéaire en fonction de N pour les APA*

Considérons maintenant les systèmes non-LTI comme les filtres adaptatifs. La figure 4.22 illustre le temps d'optimisation de l'algorithme LMS en fonction du nombre d'itérations. Le filtre LMS choisi est de taille 16. Le temps initial des méthodes analytiques désigne le calcul des différents termes statistiques présents dans l'expression analytique. La méthode directe par calcul de la somme aboutit à un meilleur temps d'optimisation après 88 évaluations de la précision. L'approche par prédiction linéaire permet d'aboutir à un meilleur résultat après seulement 6 itérations. En pratique, pour un problème d'optimisation de 30 à 33 variables, la précision est évaluée 50 000 à 100 000 fois [74]. Ainsi, l'approche analytique permet d'obtenir des gains de temps très importants. Sur la figure 4.23, le temps d'optimisation des APA est présenté. Le système choisi a pour dimensions $N = 11$ et $K = 8$. L'approche par calcul de la somme aboutit à un meilleur résultat après 105 itérations alors que la méthode par prédiction linéaire permet d'obtenir des gains de temps après 5 itérations seulement. Ainsi, cette partie illustre l'importance d'utiliser des approches analytiques comme celle développée ici pour réduire de manière importante les temps de conversion en virgule fixe et d'optimisation de la spécification virgule fixe.

4.3 Conclusion

Dans ce chapitre, deux approches ont été présentées au sein desquelles les modèles d'évaluation automatique de la précision ont été intégrés. D'une part, les modèles dédiés sont utilisés au sein d'un générateur de composants virtuels (IP). D'autre part, la méthode générale d'évaluation de la précision s'intègre au flot général de conversion de virgule flottante en virgule fixe. La méthodologie d'évaluation automatique de la précision utilisant le modèle général a été présentée et l'outil

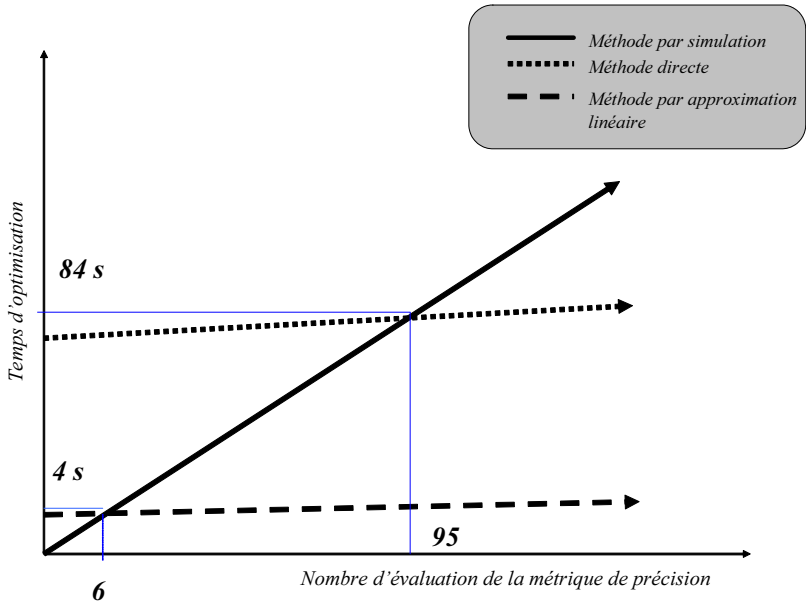

FIG. 4.22 – *Temps d'optimisation de l'algorithme LMS*

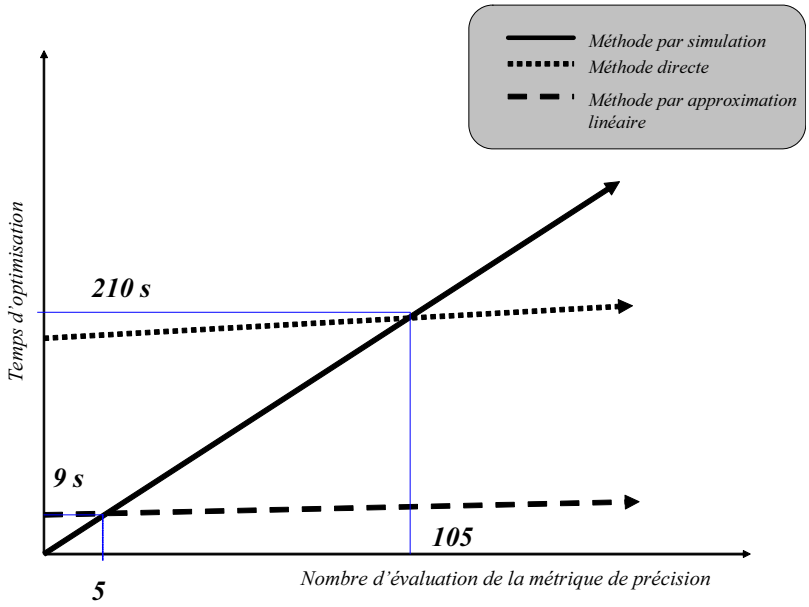

FIG. 4.23 – *Temps d'optimisation des APA*

développé sous Matlab, déterminant automatiquement le RSBQ d'une application a été détaillé. De plus, les temps d'optimisation de notre approche ont été évalués. Les gains de temps comparés à une approche basée sur des simulations virgule fixe apparaissent après quelques itérations seulement. Ainsi, ces résultats montrent l'intérêt de l'utilisation de méthodes analytiques.

Conclusion et Perspectives

L'implantation d'applications de traitement numérique du signal dans les systèmes embarqués requièrent l'utilisation de l'arithmétique virgule fixe pour satisfaire les contraintes de coût, de performance et de consommation d'énergie. Le codage manuel des données en virgule fixe est une tache fastidieuse, longue et source d'erreurs. De plus, la réduction du temps de mise sur le marché des applications nécessite l'utilisation d'outils de haut-niveau. Ainsi, des outils de conversion automatique de virgule flottante en virgule fixe sont nécessaires. La conversion a pour objectif de minimiser une fonction de coût (surface de l'architecture, consommation d'énergie, temps d'exécution du code) de l'implantation. Au sein de ce type d'outil, une étape importante est l'évaluation de la précision. En effet, le coût de l'implantation est minimisée sous contrainte de précision des calculs. A chaque itération du processus d'optimisation, la précision des calculs est évaluée. Ainsi, dans ce travail de recherche, une méthode d'évaluation automatique de la précision des systèmes en virgule fixe a été définie. La méthodologie développée détermine l'expression du Rapport Signal à Bruit de Quantification associée à une application. L'approche développée est analytique car les approches basées sur des simulations en virgule fixe conduisent à des temps d'optimisation prohibitifs.

Le travail s'articule autour de deux axes de recherche principaux. Dans un premier temps, des modèles analytiques d'évaluation de la précision ont été définis pour les filtres adaptatifs. Les méthodes d'évaluation automatique de la précision ne permettent pas de traiter ce type d'application non-LTI et présentant une récursion dans le graphe. Les modèles existants et dédiés à ce type d'application ne sont valides que pour une loi de quantification par arrondi convergent. Ainsi, le premier objectif de ce travail a été d'élaborer des modèles destinés aux filtres adaptatifs. Des modèles analytiques de calcul de la puissance du bruit ont été proposés pour les algorithmes du gradient et plus spécifiquement les algorithmes LMS, NLMS, Leaky-LMS et APA. Ces modèles ont été étendus pour définir une expression générale aux différents algorithmes du gradient. Les expressions obtenues sont valides pour toutes les lois de quantification existantes.

La seconde partie de ce travail concerne l'élaboration d'une méthode générale d'évaluation de la précision valide pour l'ensemble des systèmes utilisant des opérations arithmétiques. Cette méthode définit les caractéristiques statistiques du bruit en sortie du système considéré en fonction des statistiques des bruits d'entrée et d'une modélisation du système. Cette modélisation sous forme matricielle est obtenue en définissant chaque opération traitée par son modèle de propagation des bruits. Une expression analytique de la puissance du bruit en sortie du système est ainsi obtenue. D'autre part, pour réduire la complexité de cette expression, une approche par prédiction linéaire à été introduite. Celle-ci permet d'obtenir plus simplement l'expression de la puissance du bruit. Pour illustrer la méthode en détails, l'exemple de l'algorithme a été traité. De plus, dans le cas d'une application de l'approche générale sur des systèmes LTI, des résultats connus ont été retrouvés.

La qualité de ces deux approches a été mesurée au moyen de diverses expérimentations. L'erreur relative commise entre la valeur estimée par ces méthodes et la valeur réelle est en général inférieure à 35%. Cette erreur représente un écart maximal de 2 à 3 db entre la puissance du bruit réelle et celle estimée par nos approches. Pour obtenir de meilleurs résultats, le nombre de points

pris en compte dans le déroulement de la somme de la méthode générale doit être plus important. En contrepartie, le temps de calcul augmente pour cette méthode basée sur le déroulement de la récurrence. Pour réduire le temps de calcul la méthode par prédiction linéaire a été proposée et le biais introduit par cette approche a été mesuré et conduit à une erreur relative de l'ordre de 20%.

Les modèles dédiés ont été exploités dans le cadre d'un nouveau type de générateur de composants matériels virtuels. Le coût architectural du composant est optimisé sous une contrainte de précision fournie par l'utilisateur. Cette contrainte de précision est évaluée à chaque itération du processus d'optimisation à l'aide du modèle dédié propre à l'application traitée.

La méthode générale d'évaluation automatique de la précision proposée s'insère dans le flot de conversion de virgule flottante en virgule fixe. La conversion en virgule fixe a pour objectif d'obtenir une spécification optimisée d'un point de vue architectural (surface de l'architecture, consommation d'énergie). Au sein de cet outil, l'évaluation de la précision de la spécification virgule fixe est déterminée en appliquant la méthode générale. A chaque itération du processus d'optimisation de la specification virgule fixe, uniquement l'expression analytique de la métrique de précision est évaluée. Le temps requis pour cette évaluation est alors négligeable. La partie finale de l'outil d'évaluation de la précision a été développé sous Matlab et les temps de calcul ont été mesurés. L'approche par prédiction linéaire a été mise en œuvre et permet de réduire de manière importante le temps de calcul par rapport à la méthode générale. Une comparaison de cette méthode avec les approches basées sur des simulations virgule fixe a été effectuée. Notre approche permet d'obtenir des temps d'optimisation plus faibles par rapport aux approches basées sur la simulation après seulement quelques itérations. Ces résultats montrent l'intérêt de notre méthodologie pour réduire le temps de développement des systèmes en virgule fixe.

Perspectives

La méthode générale proposée permet de simplifier l'expression de la puissance du bruit en sortie des différentes transformées telles que la transformée de Fourier rapide ou en cosinus discret. L'utilisation des matrices permet d'obtenir une expression plus compacte de cette puissance de bruit comme présenté pour la FFT dans la section 3.2.2. De plus, les différentes matrices modélisant le parcours des sources de bruits se construisent de manière récursive. Ainsi, une perspective est de développer un modèle Matlab spécifique aux FFT. Ce modèle a pour but de définir la puissance du bruit de sortie pour une taille de FFT quelconque. L'objectif est d'utiliser la construction récursive des matrices pour obtenir une expression de la puissance du bruit de sortie. Ceci peut servir de base à la génération d'IPs pour les FFT.

La seconde perspective concerne l'interface des parties frontales et finales de l'outil d'évaluation de la précision. Les équations récurrentes du système sont déterminées par une méthode développée dans [74] et présentée dans le dernier chapitre. La partie développée sous Matlab dans le cadre de ce travail de recherche utilise les pseudo fonctions de transfert pour déterminer l'expression du RSBQ de l'application considérée. L'objectif est de définir une interface entre la détermination des équations récurrente et la partie calcul du RSBQ développée sous Matlab. Cette interface doit permettre de déterminer les pseudo fonctions de transfert sous Matlab à partir des équations récurrentes du système. Ainsi, cette interface permettra d'obtenir un outil complet d'évaluation analytique de la précision.

Une autre perspective de ce travail de recherche se situe au niveau des opérations traitées. L'objectif est de pouvoir intégrer de nouveaux opérateurs au sein de la méthodologie. Par exemple, les opérateurs décisionnels ne peuvent s'insérer directement aux modèles. Ces opérateurs sont caractérisés par le fait que la valeur de leur sortie dépend d'une condition vérifiée par le signal en entrée de l'opérateur. Les principaux exemples sont le signe ou le module. La sortie de l'opérateur signe est égale à 1 si le signal d'entrée est positif, 0 sinon. Pour ce type d'opérateurs, la propaga-

tion des bruits ne peut pas se modéliser par l'approche proposée dans ce travail de recherche. En effet, seule une approche statistique, peut être mise en œuvre. Les différentes valeurs pouvant être prises par le bruit en sortie sont déterminées ainsi que leur probabilité d'apparition. Si ces bruits s'insèrent dans un graphe contenant une récursion, les probabilités d'apparition d'un bruit en sortie d'un opérateur décisionnel dépendent des probabilités des instants précédents. L'utilisation des Chaines de Markov semble être une solution pour résoudre ce type de problème. Des recherches ont été faites dans le cadre de ce travail de recherche, mais des études plus approfondies sur ces opérateurs décisionnels sont encore nécessaires pour pouvoir généraliser le modèle développé à un ensemble plus important d'opérations.

Une dernière perspective concerne l'évaluation de la dynamique au sein de l'outil de conversion de virgule flottante en virgule fixe. La dynamique des données présentes dans le système est évaluée de manière analytique pour des systèmes LTI (normes L1 et Chebycheff) et pour des systèmes non-LTI non récursifs (Arithmétique d'Intervalle). Cependant, aucune méthode analytique n'est disponible pour traiter la dynamique des systèmes non-LTI présentant une récursion comme les filtres adaptatifs. L'objectif est de définir une méthode analytique d'évaluation de la dynamique des systèmes traités. Une première approche pourrait être de réutiliser les techniques développées dans le cadre de ce travail de recherche pour l'évaluation de la dynamique.

Annexe A

Calcul des caractéristiques des bruits de sortie

Dans cette partie, les calculs des statistiques du bruit de sortie sont démontrés. Le bruit de sortie $b_{yu}(n)$ est défini par la relation suivante

$$b_{yu}(n) = \sum_{i=1}^{Ne} \sum_{k=0}^{n} \underline{\mathbf{A}}_{ui}(k) b_i(k) \underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k) b_i^t(k) \underline{\mathbf{V}}_{ui}(k) \tag{A.1}$$

A.1 Moyenne

Pour calculer la moyenne de ce terme, l'hypothèse suivante est posée. Comme les bruits sont non-corrélés aux signaux, l'égalité suivante est vérifiée [19] : $E(AbD) = E(Am_bD)$ où E désigne l'espérance, A et D deux termes de signaux et b un bruit. La valeur de la moyenne du bruit de sortie $b_{yu}(n)$ est donnée par la relation suivante

$$m_{b_{yu}} = E(b_{yu}(n)) = \sum_{i=1}^{Ne} \sum_{k=0}^{n} E[\underline{\mathbf{A}}_{ui}(k) m_{b_i} \underline{\mathbf{B}_{ui}(k)} + \underline{\mathbf{U}}_{ui}(k) \mu_{b_i}^t \underline{\mathbf{V}}_{ui}(k)] \tag{A.2}$$

A.2 Auto-corrélation

Le terme $b_{yu}(n)$ est de taille M_{y_u}xN_{y_u}. Sa matrice d'autocorrélation $R_{b_{yu}b_{yu}^t}(\theta) = E[b_{yu}(n)b_{yu}^t(n-\theta)]$ de dimensions M_{y_u}xM_{y_u} est égale à :

$$
\begin{aligned}
\mathbf{R}_{b_{yu}b_{yu}}(\theta) &= E[b_{yu}(n)b_{yu}^t(n-\theta)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\{\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k) + \underline{\mathbf{U}}_{ui}(k)b_i^t(k)\underline{\mathbf{V}}_{ui}(k)\} \\
&.\{\underline{\mathbf{A}}_{uj}(m)b_j(m)\underline{\mathbf{D}}_{uj}(m) + \underline{\mathbf{U}}_{uj}(m)b_j^t(m)\underline{\mathbf{V}}_{uj}(m)\}^t] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m) \\
&+ \underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m) \\
&+ \underline{\mathbf{U}}_{ui}(k)b_i^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m) + \underline{\mathbf{U}}_{ui}(k)b_j^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m)\right] \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m)\right] \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left[\underline{\mathbf{U}}_{ui}(k)b_i^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m)\right] \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left[\underline{\mathbf{U}}_{ui}(k)b_j^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m)\right]
\end{aligned}
\tag{A.3}
$$

L'expression s'écrit comme la somme de quatre termes développés par la suite. Pour ce faire, deux cas sont différenciés. Premier cas, le bruit d'entrée est considéré comme vectoriel. Dans un second temps, le cas où le bruit d'entrée est matriciel est étudié

A.2.1 Bruit vectoriel

Premier terme

Pour la calcul, les notations suivantes sont introduites $\underline{\mathbf{A}}_{ui}(k) = A(k)$
$\underline{\mathbf{D}}_{ui}(k) = D(k)$

Le terme $E[A(k)b_i(k)D(k)D^t(m)b_j^t(m)A^t(m)]$ est à déterminer pour des valeurs de i,j,k et m fixées. Comme $b_{yu}(n)$ est de taille $M_{y_u}\text{x}N_{b_{yu}}$, $A(k)b_i(k)D(k)$ l'est également. D'autre part, $b_i(k)$ et $b_j(n)$ sont des vecteurs de taille M_{b_i}x1 et M_{b_j}x1, donc $A(k)$ et $D(k)$ ont pour dimensions $M_{y_u}\text{x}M_{b_i}$ et $1\text{x}N_{b_{yu}}$.
Soit la matrice M de taille $M_{y_u}\text{x}M_{y_u}$ définie par $M = E[A(k)b_i(k)D(k)D^t(m)b_j^t(m)A^t(m)]$. Pour déterminer M en détails, chacun de ces élément doit être déterminé. Pour ce faire, l'élément de la $a^{\text{ème}}$ ligne et $b^{\text{ème}}$ colonne de M est calculé. Celui-ci est exprimé en calculant un produit de six matrices

$$
(M)_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_{yu}}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)b_{i_{(c,1)}}(k)D_{(1,d)}(k)D_{(d,1)}^t(m)b_{j_{(1,e)}}^t(m)A_{(e,b)}^t(m)\right] \tag{A.4}
$$

ou encore, en réorganisant les termes

$$
(M)_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_{yu}}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)b_{i_{(c,1)}}(k)b_{j_{(1,e)}}^t(m)D_{(1,d)}(k)D_{(d,1)}^t(m)A_{(e,b)}^t(m)\right] \tag{A.5}
$$

Le terme $b_{i_{(c,1)}}(k)b^t_{j_{(1,e)}}(m)$ est retrouvé dans ce produit de matrices. Or l'espérance de ce terme correspond exactement à l'élément de la $c^{\text{ème}}$ ligne et $e^{\text{ème}}$ colonne de la matrice d'intercorrélation entre les bruits b_i et b_j

$$E(b_{i_{(c,1)}}(k)b^t_{j_{(1,e)}}(m)) = (\mathbf{R}_{b_ib_j}(m-k))_{(c,e)} \tag{A.6}$$

De plus, la matrice d'intercorrélation est égale à la somme de la matrice d'intercovariance et de la matrice produit des moyennes soit $\mathbf{R}_{b_ib_j}(m-k) = \mathbf{C}_{b_ib_j}(m-k) + m_{b_i}m^t_{b_j}$.

En réinjectant l'égalité précédente dans l'équation (A.7) et en utilisant l'hypothèse $E(AbD) = E(Am_bD)$, cela conduit à l'expression suivante :

$$\begin{aligned}
(M)_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_{yu}}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\mathbf{C}_{b_ib_j}(m-k)_{(c,e)}D_{(1,d)}(k)D^t_{(d,1)}(m)A^t_{(e,b)}(m)\right] \\
&+ E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_{yu}}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)m_{b_i}(k)_{(c,1)}D_{(1,d)}(k)D^t_{(d,1)}(m)m^t_{b_j}(m)_{(1,e)}A^t_{(e,b)}(m)\right] \\
&= E\left[\underbrace{\sum_{c=1}^{M_{b_i}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)\mathbf{C}_{b_ib_i}(m-k)_{(c,e)}A^t_{(e,b)}(m)}_{\left(A(k)\mathbf{C}_{b_ib_j}(m-k)A^t(m)\right)_{(a,b)}}\underbrace{\sum_{d=1}^{N_{b_{yu}}} D_{(1,d)}(k)D^t_{(d,1)}(m)}_{Tr(D(k)D^t(m))}\right] \\
&+ E\left[\left(A(k)m_{b_i}(k)D(k)D^t(m)m^t_{b_j}(m)A^t(m)\right)_{(a,b)}\right] \\
M &= E\left[A(k)\mathbf{C}_{b_ib_j}(m-k)A^t(m)Tr(D(k)D^t(m)) + A(k)m_{b_i}(k)B(k)B^t(m)m^t_{b_j}(m)A^t(m)\right]
\end{aligned} \tag{A.7}$$

Ainsi, le premier terme s'écrit sous la forme suivante :

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)b^t_j(m)\underline{\mathbf{A}}^t_{uj}(m)] \\
&= \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(Tr[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}^t_{uj}(m)\right)}_{\text{Terme de covariance}} \\
&+ \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)m^t_{b_j}\underline{\mathbf{A}}^t_{uj}(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.8}$$

Ce terme peut être analysé comme étant composée de deux parties :

– Une partie intégrant les termes de moyenne correspondant à la moyenne au carré du bruit de sortie $b_{yu}(n)$.

– Une partie composée des termes de covariance définissant la covariance du bruit de sortie.

Ainsi, la matrice d'autocorrélation est bien égale à la somme de la matrice de covariance et de la matrice de moyenne au carré.

Deuxième terme

Comme pour le calcul du second terme les notations utilisées sont les suivantes :
$\underline{\mathbf{A}}_{ui}(k) = A(k)$ et a pour dimensions M_{y_u}xM_{b_i}
$\underline{\mathbf{D}}_{ui}(k) = B(k)$ et a pour dimensions 1x$N_{b_{yu}}$
$\underline{\mathbf{U}}_{uj}(k) = U(k)$ et a pour dimensions M_{y_u}x1
$\underline{\mathbf{V}}_{uj}(k) = V(k)$ et a pour dimensions M_{b_j}xN_{y_u}

La matrice M définissant ce terme est égale à :

$$\begin{aligned}(M)_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)b_{i_{(c,1)}}(k)D_{(1,d)}(k)V^t_{(d,e)}(m)b_{j_{(e,1)}}(m)U^t_{(1,b)}(m)\right] \\ &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}}\sum_{g=1}^{N_{b_i}} A_{(a,c)}(k)b_{i_{(c,1)}}(k)b_{j_{(e,1)}}(m)D_{(1,d)}(k)V^t_{(d,e)}(m)U^t_{(1,b)}(m)\right] \\ &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}}\sum_{g=1}^{N_{b_i}} A_{(a,c)}(k)b_{i_{(c,1)}}(k)b^t_{j_{(1,e)}}(m)D_{(1,d)}(k)V^t_{(d,e)}(m)U^t_{(1,b)}(m)\right]\end{aligned} \tag{A.9}$$

Comme précédemment, l'intercorrélation des bruits d'entrée est retrouvée.

$$E(b_{i_{(c,1)}}(k)b^t_{j_{(1,e)}}(m)) = (\mathbf{C}_{b_ib_j}(m-k))_{(c,e)} + (m_{b_i}m_{b^t_j})_{(c,e)} \tag{A.10}$$

ceci conduit à l'expression suivante :

$$\begin{aligned}(M)_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)(\mathbf{C}_{b_ib_j}(m-k))_{(c,e)}V_{(e,d)}(m)D^t_{(d,1)}(k)U^t_{(1,b)}(m)\right] \\ &+ E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} A_{(a,c)}(k)m_{b_i}(k)_{(c,1)}D_{(1,d)}(k)V^t_{(d,e)}(m)m_{b_j}(m)_{(e,1)}U^t_{(1,b)}(m)\right] \\ M &= E\left[A(k)\mathbf{C}_{b_ib_j}(m-k)V(m)D^t(k)U^t(m)\right] + E\left[A(k)m_{b_i}(k)D(k)V^t(m)m_{b_j}(m)U^t(m)\right]\end{aligned} \tag{A.11}$$

Finalement, le deuxième terme s'écrit sous la forme :

$$\begin{aligned}&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}^t_{uj}(m)b_j(m)\underline{\mathbf{U}}^t_{uj}(m)] \\ &= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}^t_{ui}(k)\underline{\mathbf{U}}^t_{uj}(m)\right)}_{\text{Terme de covariance}} \\ &+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}^t_{uj}(m)m_{b_j}\underline{\mathbf{U}}^t_{uj}(m))}_{\text{Terme de Moyenne}}\end{aligned} \tag{A.12}$$

Comme pour le premier terme, celui-ci est composé d'une partie covariance et d'une partie moyenne.

Troisième terme

Ce troisième terme se développe comme le deuxième et conduit à l'équation finale suivante :

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{U}}_{ui}(k)b_i^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)b_j^t(m)\underline{\mathbf{A}}_{uj}^t(m)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right)}_{\text{Terme de covariance}} \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E(\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{A}}_{uj}^t(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.13}$$

Quatrième terme

Pour calculer ce dernier terme, les notations précédemment utilisées sont reprises.

$$\begin{aligned}
(M)_{(a,b)} &= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} U_{(a,1)}(k)b_{i_{(1,c)}}^t(k)V_{(c,d)}(k)V_{(d,e)}^t(m)b_{j_{(e,1)}}(m)U_{(1,b)}^t(m)\right] \\
&= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} U_{(a,1)}(k)b_{j_{(e,1)}}(k)b_{i_{(1,c)}}^t(m)V_{(c,d)}(k)V_{(d,e)}^t(m)U_{(1,b)}^t(m)\right] \\
&= E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} U_{(a,1)}(k)U_{(1,b)}^t(m)V_{(d,e)}^t(m)b_{j_{(e,1)}}(m)b_{i_{(1,c)}}^t(k)V_{(c,d)}(k)\right]
\end{aligned} \tag{A.14}$$

Or

$$E(b_{j_{(e,1)}}(k)b_{i_{(1,c)}}^t(m)) = (\mathbf{C}_{b_jb_i}(m-k))_{(e,c)} + (m_{b_i}m_{b_j^t})_{(e,c)} \tag{A.15}$$

Ceci conduit à l'équation suivante :

$$\begin{aligned}
(M)_{(a,b)} &= E\Bigg[U_{(a,1)}(k)U_{(1,b)}^t(m)\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{y_u}}\sum_{e=1}^{M_{b_j}} V_{(d,e)}^t(m)(\mathbf{C}_{b_jb_i}(k-m))_{(e,c)}V_{(c,d)}(k) \\
&\quad + U(k)m_{b_i}^t(k)V(k)V^t(m)m_{b_j}(m)U^t(m)\Big] \\
M &= E\left[U(k)U^t(m)Tr(V^t(m)\mathbf{C}_{b_jb_i}(k-m)V(k))\right]
\end{aligned} \tag{A.16}$$

Cela aboutit à l'équation finale suivante

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{U}}_{ui}(k)b_j^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\right)}_{\text{Terme de covariance}} \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{U}}_{uj}^t(m)\right)}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.17}$$

Calcul final

En introduisant les équations (A.30), (A.31), (A.32), (A.33) dans l'équation (A.3), la formule de l'autocorrélation du bruit de sortie est obtenue sous la forme suivante :

$$
\begin{aligned}
R_{b_{yu}b_{yu}^t}(\theta) &= E[b_{yu}(n)b_{yu}^t(n-\theta)] \qquad (A.18)\\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(Tr[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{A}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)m_{b_j}\underline{\mathbf{U}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{A}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)m_{b_j}\underline{\mathbf{U}}_{uj}^t(m)\right) \qquad (A.19)
\end{aligned}
$$

pouvant se mettre sous la forme suivante :

$$
\begin{aligned}
\mathbf{R}_{b_{yu}b_{yu}^t}(\theta) &= E[b_{yu}(n)b_{yu}^t(n-\theta)]\\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(Tr[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}_{uj}^t(m)]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}_{ui}^t(k)\underline{\mathbf{U}}_{uj}^t(m)\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}_{ui}^t(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}_{uj}^t(m)\right) \qquad (A.20)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\right)\\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left[(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)+\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k))\right.\\
&\left.\cdot\,(\underline{\mathbf{A}}_{uj}(m)m_{b_j}\underline{\mathbf{D}}_{uj}(m)+\underline{\mathbf{U}}_{uj}(m)m_{b_j}^t\underline{\mathbf{V}}_{uj}(m))^t\right]
\end{aligned}
$$

A.2.2 Cas matriciel

Dans cette partie, le cas où le bruit d'entrée est matriciel est traité. Les quatre termes de l'équation sont calculés comme dans le cas vectoriel.

Premier terme

Pour rappel, ce terme est égal à $\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)b^t_j(m)\underline{\mathbf{A}}^t_{uj}(m)]$. Comme dans le cas vectoriel, en reprenant les mêmes notations, l'élément de la $a^{\text{ème}}$ ligne et $b^{\text{ème}}$ colonne de M est exprimé en calculant un produit de six matrices

$$(M)_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_i}}\sum_{e=1}^{N_{y_u}}\sum_{f=1}^{N_{b_j}}\sum_{g=1}^{M_{b_j}} A_{(a,c)}(k)b_{i_{(c,d)}}(k)D_{(d,e)}(k)D^t_{(e,f)}(m)b^t_{j_{f,g)}}(m)A^t_{(g,b)}(m)\right] \quad \text{(A.21)}$$

ou encore

$$(M)_{(a,b)} = E\left[\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_i}}\sum_{e=1}^{N_{y_u}}\sum_{f=1}^{N_{b_j}}\sum_{g=1}^{M_{b_j}} A_{(a,c)}(k)b_{i_{(c,d)}}(k)b^t_{j_{f,g)}}(m)D_{(d,e)}(k)D^t_{(e,f)}(m)A^t_{(g,b)}(m)\right] \quad \text{(A.22)}$$

Le terme des bruits d'entrée est retrouvé sous la forme $b_{i_{(c,d)}}(k)b^t_{j_{(f,g)}}(m)$ a l'intérieur de ce produit de matrices. Or, l'intercorrélation des bruits b_i et b_j s'exprime selon l'expression suivante :

$$\sum_d b_{i_{(c,d)}}(k)b^t_{j_{(d,g)}}(m) \quad \text{(A.23)}$$

De ce fait, le terme $b_{i_{(c,d)}}(k)b^t_{j_{(f,g)}}(m)$ pour $f \neq d$ ne peut être exprimé en introduisant l'intercorrélation des deux bruits. Ainsi, une majoration de ces termes est effectuée.

Le terme posant problème s'écrit de la façon suivante :

$$\sum_d\sum_e\sum_f b_{i_{(c,d)}}(k)D_{(d,e)}(k)D^t_{(e,f)}(m)b^t_{j_{(f,g)}}(m) \quad \text{(A.24)}$$

ou encore

$$\sum_d\sum_f b_{i_{(c,d)}}(k)\left(D(k)D^t(m)\right)_{(d,f)}b^t_{j_{(f,g)}}(m) \quad \text{(A.25)}$$

La matrice $D(k)D^t(m)$ peut être diagonalisée. Dans son espace transposé, seuls apparaissent les termes de la diagonale. Cette matrice DD^t est positive car elle modélise la puissance des éléments contenus dans celle-ci. Dans ce cas, chaque valeur propre λ_i de cette matrice est majorée par sa trace

$$\lambda_i \leq \sum_j \lambda_j \quad \text{(A.26)}$$

Cette majoration permet d'écrire l'expression suivante sous la forme :

$$Tr(D(k)D^t(m))\sum_d\sum_f b_{i_{(c,d)}}(k)\mathbf{I}_{N_{(d,f)}}b^t_{j_{(f,g)}}(m) = Tr(D(k)D^t(m))\sum_d b_{i_{(c,d)}}(k)b^t_{j_{(d,g)}}(m) \quad \text{(A.27)}$$

Cela permet d'aboutir à la même expression que dans le cas vectoriel

$$(M)_{(a,b)} = E\left[Tr(D(k)D^t(m))\sum_{c=1}^{M_{b_i}}\sum_{d=1}^{N_{b_i}}\sum_{g=1}^{M_{b_i}} A_{(a,c)}(k)b_{i_{c,d)}}(k)b^t_{j_{(d,g)}}(m)A^t_{(g,b)}(m)\right] \tag{A.28}$$

Alors, l'intercorrélation des deux bruits apparaît, conduisant à une expression identique à celle développée dans le cas vectoriel.

$$M = E\left[A(k)\mathbf{C}_{b_ib_j}(m-k)A^t(m)Tr(D(k)D^t(m)) + A(k)m_{b_i}(k)D(k)D^t(m)m^t_{b_j}(m)A^t(m)\right] \tag{A.29}$$

Ainsi, le premier terme s'écrit selon l'équation suivante :

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)b^t_j(m)\underline{\mathbf{A}}^t_{uj}(m)] \\
&= \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(Tr[\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)]\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}^t_{uj}(m)\right)}_{\text{Terme de covariance}} \\
&+ \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)m^t_{b_j}\underline{\mathbf{A}}^t_{uj}(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.30}$$

Second terme

Avec les mêmes hypothèses que précédemment, cela aboutit à un terme identique à celui obtenu dans le cas vectoriel.

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{A}}_{ui}(k)b_i(k)\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}^t_{uj}(m)b_j(m)\underline{\mathbf{U}}^t_{uj}(m)] \\
&= \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{A}}_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{V}}_{uj}(m)\underline{\mathbf{D}}^t_{ui}(k)\underline{\mathbf{U}}^t_{uj}(m)\right)}_{\text{Terme de covariance}} \\
&+ \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E(\underline{\mathbf{A}}_{ui}(k)m_{b_i}\underline{\mathbf{D}}_{ui}(k)\underline{\mathbf{V}}^t_{uj}(m)m_{b_j}\underline{\mathbf{U}}^t_{uj}(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.31}$$

Troisième terme

Le troisième terme est également le même

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{U}}_{ui}(k)b^t_i(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)b^t_j(m)\underline{\mathbf{A}}^t_{uj}(m)] \\
&= \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{D}}_{uj}(m)\underline{\mathbf{V}}^t_{ui}(k)\mathbf{C}_{b_ib_j}(m-k)\underline{\mathbf{A}}^t_{uj}(m)\right)}_{\text{Terme de covariance}} \\
&+ \underbrace{\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E(\underline{\mathbf{U}}_{ui}(k)m^t_{b_i}\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{D}}^t_{uj}(m)m^t_{b_j}\underline{\mathbf{A}}^t_{uj}(m))}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.32}$$

Quatrième terme

Le quatrième n'évolue pas non-plus

$$\begin{aligned}
&\sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} E[\underline{\mathbf{U}}_{ui}(k)b_j^t(k)\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)b_j(m)\underline{\mathbf{U}}_{uj}^t(m)] \\
&= \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{U}}_{ui}(k)\underline{\mathbf{U}}_{uj}^t(m)Tr[\underline{\mathbf{V}}_{uj}^t(m)\mathbf{C}_{b_jb_i}(k-m)\underline{\mathbf{V}}_{ui}(k)]\right)}_{\text{Terme de covariance}} \\
&+ \sum_{i,j=1}^{Ne}\sum_{k=0}^{n}\sum_{m=0}^{n-\theta} \underbrace{E\left(\underline{\mathbf{U}}_{ui}(k)m_{b_i}^t\underline{\mathbf{V}}_{ui}(k)\underline{\mathbf{V}}_{uj}^t(m)m_{b_j}^t\underline{\mathbf{U}}_{uj}^t(m)\right)}_{\text{Terme de Moyenne}}
\end{aligned} \tag{A.33}$$

A.2.3 Conclusion

En distinguant ces deux cas, le même résultat est obtenu. Néanmoins, alors que les résultats sont exacts dans le cas où les bruits sont vectoriels, des majorations sont effectuées dans le cas où les bruits sont matriciels.

Annexe B

Réponse Impulsionnelle du filtre IIR

Dans cette partie, les calculs des termes de la réponse impulsionnelle pour un filtre IIR 2 défini dans la section 3.2.1 sont présentés. Avec le modèle développé dans la partie 3.2.1, les termes obtenus sont les suivants

$$\begin{aligned} h^{(n)}_{(n)} &= 1 \\ h^{(n)}_{(n-1)} &= a_1 \\ h^{(n)}_{(n-2)} &= a_1^2 + a_2 \\ h^{(n)}_{(n-3)} &= a_1^3 + 2a_2a_1 \\ h^{(n)}_{(n-4)} &= a_1^4 + 3a_1^2a_2 + a_2^2 \end{aligned} \tag{B.1}$$

L'objectif est de retrouver ces valeurs en utilisant les techniques adaptées aux systèmes linéaires invariants dans le temps (LTI). La fonction de transfert du système est la suivante :

$$H_1(z) = \frac{1}{1 - a_1z^{-1} - a_2z^{-2}} \tag{B.2}$$

Cette fonction de transfert peut se décomposer de la manière suivante :

$$\begin{aligned} H_1(z) &= \frac{1}{1 - a_1z^{-1} - a_2z^{-2}} \\ &= \frac{1}{(1 - z_1z^{-1})(1 - z_2z^{-1})} \end{aligned} \tag{B.3}$$

où z_1 et z_2 sont les deux pôles de $H_1(z)$. Ils sont supposés réels. Leur valeur est alors recherchée. Le discriminant du dénominateur est donné par l'expression suivante :

$$\Delta = a_1^2 + 4a_2 \tag{B.4}$$

Donc z_1 et z_2 sont égaux à

$$\begin{aligned} z_1 &= \frac{1}{2}(a_1 - \sqrt{a_1^2 + 4a_2}) \\ z_2 &= \frac{1}{2}(a_1 + \sqrt{a_1^2 + 4a_2}) \end{aligned} \tag{B.5}$$

De ce fait, l'égalité suivante est vérifiée :

$$\begin{aligned} H_1(z) &= \frac{1}{1-a_1z^{-1}-a_2z^{-2}} \\ H_1(z) &= \frac{1}{(1-z_1z^{-1})(1-z_2z^{-1})} \\ H_1(z) &= [\sum_{k=0}^{\infty}(z_1z^{-1})^k][\sum_{l=0}^{\infty}(z_2z^{-1})^l] \\ H_1(z) &= \sum_{m=0}^{\infty} c_m z^{-m} \end{aligned} \tag{B.6}$$

avec la relation $c_m = \sum_{j=0}^{m} z_1^j z_2^{m-j}$.

Maintenant, avec l'equation (B.6), $h_1(n)$ est parfaitement défini et est égale à :

$$h_1(n) = \sum_{m=0}^{\infty} c_m \delta(n-m) \tag{B.7}$$

où δ correspond à la fonction Dirac. Pour pouvoir déterminer entièrement $h1(n)$, les valeurs de c_m doivent être calculées.

$$\begin{aligned} c_m &= \sum_{j=0}^{m} z_1^j z_2^{m-j} \\ c_m &= \frac{z_2^{m+1}-z_1^{m+1}}{z_2-z_1} \end{aligned} \tag{B.8}$$

Or

$$z_2 - z_1 = \sqrt{a_1^2+4a_2} \tag{B.9}$$

De plus

$$\begin{aligned} z_2^{m+1} - z_1^{m+1} &= [\tfrac{1}{2}(a_1+\sqrt{a_1^2+4a_2})]^{m+1} - [\tfrac{1}{2}(a_1-\sqrt{a_1^2+4a_2})]^{m+1} \\ &= \frac{1}{2^{m+1}} \sum_{j=0}^{m+1} C_{m+1}^j a_1^{m+1-j} \left[(\sqrt{a_1^2+4a_2})^j - (-1)^j (-\sqrt{a_1^2+4a_2})^j \right] \\ &= \frac{1}{2^{m+1}} \sum_{j=0}^{m+1} C_{m+1}^j a_1^{m+1-j} \left[\underbrace{(\sqrt{a_1^2+4a_2})^j - (-1)^j (\sqrt{a_1^2+4a_2})^j}_{=0 \text{ si j est pair}} \right] \end{aligned} \tag{B.10}$$

(B.11)

Les termes précédents sont nuls si j est pair. De ce fait, en posant $j = 2p+1$, seuls les termes impairs sont conservés. Dans la somme, j varie de 0 à $m+1$. Donc p varie de 0 à $\lfloor \frac{m}{2} \rfloor$, où $\lfloor \rfloor$ désigne la partie entière. L'équation (B.10) devient alors :

$$\begin{aligned} z_2^{m+1} - z_1^{m+1} &= \frac{1}{2^{m+1}} \sum_{p=0}^{\lfloor \frac{m}{2} \rfloor} 2C_{m+1}^{2p+1} a_1^{m-2p} \left(\sqrt{a_1^2+4a_2} \right)^{2p+1} \\ &= \frac{\sqrt{a_1^2+4a_2}}{2^m} \sum_{p=0}^{\lfloor \frac{m}{2} \rfloor} C_{m+1}^{2p+1} a_1^{m-2p} (a_1^2+4a_2)^p \end{aligned} \tag{B.12}$$

Finalement, en intégrant les équations (B.9) et (B.12 dans l'équation (B.8), cela conduit à l'expression suivante :

$$b'(n) = \sum_{m=0}^{\infty} c_m b(n-m) \tag{B.13}$$

avec

$$\begin{aligned} c_m = & \quad \frac{z_2^{m+1} - z_1^{m+1}}{z_2 - z_1} \\ = & \quad \frac{1}{2^m} \sum_{p=0}^{\lfloor \frac{m}{2} \rfloor} C_{m+1}^{2p+1} a_1^{m-2p} (a_1^2 + 4a_2)^p \end{aligned} \tag{B.14}$$

$$\tag{B.15}$$

Ainsi, $h_1(n)$ est défini. Or avec notre modèle défini par l'équation (B.1), le bruit en sortie s'écrit selon la formule

$$b'(n) = \sum_{j=0}^{n} h_{(j)}^{(n)} b(j) \tag{B.16}$$

En comparant les équations (B.13) et (B.16), la validité de notre méthode est démontrée si les égalités suivantes sont vérifiées :

$$h_{(n-i)}^{(n)} = c_i \tag{B.17}$$

De plus, avec notre modèle, les premiers termes de la réponse impulsionnelle ont été calculés et sont résumés dans l'expression suivante :

$$\begin{aligned} h_{(n)}^{(n)} = & \quad 1 \\ h_{(n-1)}^{(n)} = & \quad a_1 \\ h_{(n-2)}^{(n)} = & \quad a_1^2 + a_2 \\ h_{(n-3)}^{(n)} = & \quad a_1^3 + 2a_2 a_1 \\ h_{(n-4)}^{(n)} = & \quad a_1^4 + 3a_1^2 a_2 + a_2^2 \end{aligned} \tag{B.18}$$

et c_i est égal à :

$$c_i = \frac{1}{2^i} \sum_{p=0}^{E(\frac{i}{2})} C_{i+1}^{2p+1} a_1^{i-2p} (a_1^2 + 4a_2)^p \tag{B.19}$$

Donc

$$\begin{aligned} c_0 = & \quad \frac{1}{2^0} \sum_{p=0}^{E(\frac{0}{2})} C_1^{2p+1} a_1^{0-2p} (a_1^2 + 4a_2)^p \\ = & \quad C_1^1 a_1^0 (a_1^2 + 4a_2)^0 \\ = & \quad 1 \end{aligned} \tag{B.20}$$

$$\begin{aligned} c_1 = & \quad \frac{1}{2^1} \sum_{p=0}^{E(\frac{1}{2})} C_2^{2p+1} a_1^{1-2p} (a_1^2 + 4a_2)^p \\ = & \quad \frac{1}{2} C_2^1 a_1^1 (a_1^2 + 4a_2)^0 \\ = & \quad a_1 \end{aligned} \tag{B.21}$$

$$\begin{aligned} c_2 = & \frac{1}{2^2}\sum_{p=0}^{E(\frac{2}{2})} C_3^{2p+1} a_1^{2-2p}(a_1^2+4a_2)^p \\ = & \frac{1}{4}[C_3^1 a_1^2(a_1^2+4a_2)^0 + C_3^3 a_1^0(a_1^2+4a_2)^1] \\ = & \frac{1}{4}[3a_1^2 + a_1^2 + 4a_2] \\ = & a_1^2 + a_2 \end{aligned} \tag{B.22}$$

$$\begin{aligned} c_3 = & \frac{1}{2^3}\sum_{p=0}^{E(\frac{3}{2})} C_4^{2p+1} a_1^{3-2p}(a_1^2+4a_2)^p \\ = & \frac{1}{8}[C_4^1 a_1^3(a_1^2+4a_2)^0 + C_4^3 a_1^1(a_1^2+4a_2)^1] \\ = & \frac{1}{8}[4a_1^3 + 4a_1^3 + 16a_1a_2] \\ = & a_1^3 + 2a_2 \end{aligned} \tag{B.23}$$

$$\begin{aligned} c_4 = & \frac{1}{2^4}\sum_{p=0}^{E(\frac{4}{2})} C_5^{2p+1} a_1^{4-2p}(a_1^2+4a_2)^p \\ = & \frac{1}{16}[C_5^1 a_1^4(a_1^2+4a_2)^0 + C_5^3 a_1^2(a_1^2+4a_2)^1 + C_5^5 a_1^0(a_1^2+4a_2)^2] \\ = & \frac{1}{16}[5a_1^4 + 10a_1^4 + 40a_1^2a_2 + a_1^4 + 16a_2^2 + 8a_1^2a_2] \\ = & a_1^4 + a_2 + 3a_1^2a_2 + a_2^2 \end{aligned} \tag{B.24}$$

Les égalités suivantes sont bien vérifiées

$$\begin{aligned} h_{(n)}^{(n)} = & \ c_0 \\ h_{(n-1)}^{(n)} = & \ c_1 \\ h_{(n-2)}^{(n)} = & \ c_2 \\ h_{(n-3)}^{(n)} = & \ c_3 \\ h_{(n-4)}^{(n)}) = & \ c_4 \end{aligned} \tag{B.25}$$

En continuant de la même façon, l'égalité de tous les autres termes est obtenue. De ce fait, le développement des calculs a permis de démontrer la validité de la méthode de calcul de la pseudo réponse impulsionnelle.

Bibliographie

[1] T. Adali and S.H. Ardalan. Convergence and error analysis of the Fixed-Point RLS Algorithm with correlated inputs. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP'90)*, pages 1479–1482, 1990.

[2] T. Adali and S.H. Ardalan. Error analysis of the Fixed-Point RLS Algorithm with correlated inputs. In *International Conference on Communications Control and Signal Processing (CCSP'90)*, pages 1291–1297, 1990.

[3] S.T. Alexander. Transient Weight Misadjustement Properties for the Finite Precision LMS Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35:1250–1258, 1987.

[4] S.H. Ardalan and S.T. Alexander. Fixed-Point Roundoff Error Analysis of the Exponentially Windowed RLS Algorithm for Time-Varying Systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 6:770–783, 1987.

[5] B. Pinçon. *Une introduction à Scilab.* ESIAL Nancy, April 2006.

[6] P. Banerjee, D. Bagchi, M. Haldar, A. Nayak, V. Kim, and R. Uribe. Automatic conversion of floating point matlab programs into fixed point fpga based hardware design. *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FFCM'03)*, 2003.

[7] M. Barberis and N. Shah. Migrating signal processing applications from floating-point to fixed-point. White paper, Catalytic Inc, Palo Alto, USA, November 2004.

[8] P. Belanovic and M. Rupp. Fixify : A Toolset for Automated Floating-point to Fixed-point Conversion. In *International Conference on Computing, Communications and Control Technologies (CCCT'04)*, pages 28–32, 2004.

[9] P. Belanovic and M. Rupp. Automated Floating-point to Fixed-point Conversion with the fixify Environment. In *IEEE Rapid System Prototyping (RSP'05)*, pages 172–178, 2005.

[10] M Bellanger. *Analyse des signaux et Filtrage Numérique Adaptatif*, volume 1 of *Collection technique et scientifique des télécommunications.* Masson, 1 edition, 1989.

[11] M. Bellanger. A survey of QR based fast least squares adaptive filters. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'91)*, pages 1833–1836, 1991.

[12] A. Benallal and A. Gilloire. Etude des effets de quantification dans les algorithmes MCR numériquement stables. In *Treizième colloque GRETSI*, 1989.

[13] N.J. Bershad. Comments on Comparaison of the Convergence of Two Algorithms for Adaptive FIR Digital Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 1604–1606, December 1985.

[14] N.J. Bershad. Analysis of the Normalized LMS Algorithm with Gaussian Inputs. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):793–806, August 1986.

[15] N.J. Bershad. On the Probability Density Function of the LMS Adaptive Filter Weights. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):43–55, 1989.

[16] N.J. Bershad and J.C.M. Bermudez. A Nonlinear Analytical Model for the Quantized LMS Algorithm-The Power-of-Two Step Size Case. *IEEE Transactions on Signal Processing*, 44(11):2895–2900, November 1996.

[17] N.J. Bershad and J.C.M. Bermudez. A Nonlinear Analytical Model for the Quantized LMS Algorithm-The Arbitrary Step Size Case. *IEEE Transactions on Signal Processing*, 44(5):1175–1183, May 1996.

[18] N.J. Bershad and J.C.M. Bermudez. New Insights on the Transient and Steady-State Behavior of the Quantized LMS Algorithm. *IEEE Transactions on Signal Processing*, 44(10):2623–2625, October 1996.

[19] C. Caraiscos and B.Liu. A Roundoff Error Analysis of the LMS Adaptive Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(1):34–41, February 1984.

[20] P.S. Chang and A.N. Willson Jr. A Roundoff Error Analysis of the Normalized LMS Algorithm. *IEEE Signal, Systems and Computers*, 2:1337–1341, 1996.

[21] Y.H. Chang, C.K. Tzou, and N.J. Bershad. Postmoothing for the LMS Algorithm and a Fixed Point Roundoff Error Analysis. *IEEE Transactions on signal processing*, 39:959–962, 1991.

[22] J.M. Cheneaux, L.S. Didier, and F. Rico. The fixed cadna library. *Real Number and Computers*, September 2003.

[23] J.M. Cheneaux and J.Vignes. Les fondements de l'arithmétique stochastique. *C.R. Académie des Sciences*, pages 1435–1440, 1992.

[24] J.M. Cioffi. Limited-Precision Effects in Adaptive Filtering. *IEEE Transactions on Circuits and Systems*, 7:821–333, 1987.

[25] J.M. Cioffi and T. Kailath. Fast, Recursive Least-Squares Transversal Filters for Adaptive Filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 304–337, April 1984.

[26] G. Constantinides. Perturbation Analysis for Word-length Optimization. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FFCM'03)*, 2003.

[27] G. Constantinides, P. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEEE Electronics Letters*, 35(23):2012–2014, November 1999.

[28] G. Constantinides, P. Cheung, and W. Luk. Roundoff-noise shaping in filter design. In *IEEE Symposium on Circuits and Systems (ISCAS'00)*, pages IV57–IV60, Geneva, Switzerland, May 28-31 2000. Institute of Electrical and Electronics Engineers.

[29] G. Constantinides, P.Y.K. Cheung, and W. Luk. The Multiple Wordlength Paradigm. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FFCM'01)*, March 2001.

[30] M. Coors, H. Keding, O. Luthje, and H. Meyr. Integer Code Generation For the TI TMS320C62x. In *International Conference on Acoustics, Speech, and Signal Processing 2001 (ICASSP'01)*, Sate Lake City, US, May 2001.

[31] M.H. Costa, J.C.M. Bermudez, and N.J. Bershad. Statistical Analysis of the LMS Algorithm with a zero-memory nonlinearity after the Adaptive Filter. In *IEEE Conference on Acoustic, Speech, and Signal Processing (ICASSP'99)*, pages 1661–1664, 1999.

[32] L. De Coster, M. Ade, R. Lauwereins, and J.A. Peperstraete. Code Generation for Compiled Bit-True Simulation of DSP Applications. In *Proceedings of the 11th International Symposium on System Synthesis (ISSS'98)*, Taiwan, December 1998.

[33] CoWare. *CoWare Signal Processing Designer*. CoWare, 2006.

[34] L.H. de Figueiredo and J. Stolfi. Affine Arithmetic : Concepts and Applications. *Numerical Algorithms*, pages 1–13, 2003.

[35] H. Dedieu and F. Castanie. Analyse des effets de quantification dans l'implantation de l'algorithme LMS. In *Onzième colloque GRETSI*, 1987.

[36] P.S. Diniz and M.G. Siqueira. Fixed-Point Error Analysis of the QR-Recursive Least Square Algorithm. *IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing*, 42(5):334–348, 1995.

[37] S.C. Douglas. Exact Expectation Analysis of the LMS Adaptive Filter for Correlated Gaussian Input Data. In *IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP'93)*, volume 3, pages 519–522, April 1993.

[38] S.C. Douglas. Performance Comparison of Two Implementations of the Leaky LMS Adaptive Filter. *IEEE Transactions on Signal Processing*, 45:2125–2129, August 1997.

[39] E. Eweda. Tracking Analysis of the Sign-Sign Algorithm for Nonstationnary Adaptive Filtering with Gaussian Data. *IEEE Transactions on Signal Processing*, pages 1375–1378, May 1997.

[40] E. Eweda. Transient and Tracking Performance Bounds of the Sign-Sign Algorithm. *IEEE Transactions on Signal Processing*, pages 2200–2210, August 1999.

[41] E. Eweda. Convergence Analysis of the Sign Algorithm Without the Independance and Gaussian Assumptions. *IEEE Transactions on Signal Processing*, pages 2535–2543, September 2000.

[42] H. Fan and X.Q. Liu. GAL and LSL Revisited : New Convergence Results. *IEEE Transactions on Signal Processing*, 41:55–64, 1993.

[43] C. Fang Fang, T. Chen, and R.A. Rutenbar. Floating Point Error Analysis based on Affine Arithmetic. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, pages 561–564, 2003.

[44] C. Fang Fang, T. Chen, and R.A. Rutenbar. Towards Efficient Static Analysis of Finite-Precision Effects in DSP Applications via Affine Arithmetic Modeling. In *Design Automation Conference (DAC'03)*, pages 496–501, 2003.

[45] A. Feuer and E. Weinstein. Convergence Analysis of LMS Filters with Uncorrelated Gaussian Data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(1):222–230, May 1985.

[46] S. Gazor and B. Farhang-Boroujeny. Quantization effects in Transform-Domain Normalized LMS Algorithm. *IEEE Transactions on Circuits and Systems*, 39:1–7, 1992.

[47] R.D. Gitlin, J.E. Mazo, and M.G. Taylor. On the design of Gradient Algorithm for Digitally Implemented Adaptive Filters. *IEEE Transactions on Circuit Theory*, 2:125–136, 1973.

[48] D.N. Godard. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Transactions on Communications*, pages 1867–1875, November 1980.

[49] M. Goel and N.R. Shanbhag. Finite-Precision Analysis of the Pipelined Strengh-Reduced Adaptive Filter. *IEEE Transactions on Signal Processing*, 46(6):1763–1769, June 1998.

[50] T. Grötker, E. Multhaup, and O.Mauss. Evaluation of HW/SW Tradeoffs Using Behavioral Synthesis. In *7th International Conference on Signal Processing Applications and Technology (ICSPAT'96)*, Boston, October 1996.

[51] R. Gupta and A.O. Hero. Transient Behaviour of Fixed-Point LMS Adaptation. In *IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP'00)*, pages 376–379, 2000.

[52] S. Haykin. *Adaptive Filter Theory*. Prentice Hall Inc, 2 edition, 1991.

[53] A. Hermánek, J. Schier, and P. A. Regalia. Architecture design for FPGA implementation of finite interval CMA. In *European Signal Processing Conference (EUSIPCO'04)*, number 2, pages 2039–2042, 2004.

[54] N. Hervé, D. Ménard, and O. Sentieys. Data wordlength optimization for fpga synthesis. In *IEEE International Workshop on Signal Processing Systems (SIPS'05)*, pages 623–628, Athens, Grece, November 2005.

[55] M.L. Honig and D.G. Messerschmidt. Convergence properties of an adaptive digital lattice filter. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 642–653, June 1981.

[56] L.L. Horowitz and K.D. Senne. Performance Advantage of Complex LMS for Controlling Narrow-Band Adaptive Arrays. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 3:722–736, 1981.

[57] K. Kalliojärvi and J. Astola. Roundoff Errors in Block-Floating Point Systems. *IEEE Transactions on Signal Processing*, 44(4):783–790, April 1996.

[58] H. Keding, M. Willems, M. Coors, and H. Meyr. FRIDGE: A Fixed-Point Design And Simulation Environment. In *Design, Automation and Test in Europe 1998 (DATE-98)*, 1998.

[59] S. Kim, K. Kum, and W. Sung. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. In *Workshop on VLSI and Signal Processing '95*, Osaka, November 1995.

[60] S. Kim and W. Sung. Fixed-Point-Simulation Utility for C and C++ Based Digital Signal Processing Programs. In *Twenty-eighth Annual Asilomar Conference on Signals, Systems, and Computer*, October 1994.

[61] S. Kim and W. Sung. Fixed-Point Error Analysis and Word Length Optimization of 8x8 IDCT Architectures. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8):935–940, December 1998.

[62] M. Kunt, M. Bellanger, F De Coulon, and C Guegen. *Techniques modernes de traitement numérique des signaux*, volume 1 of *Traitement de l'information*. Presses polytechniques et universitaires romandes et CNET-ENST, 1 edition, 1991.

[63] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee. *DSP Processor Fundamentals: Architectures and Features*. Berkeley Design Technology, Inc, Fremont, CA, 1996.

[64] H. Leung and S. Haykin. Error Bound Method and its Application to the LMS Algorithm. *IEEE Transactions on Signal Processing*, 39:354–358, 1991.

[65] A.P. Livias and P. A. Regalia. On the Numerical Stability and Accuracy of the Conventional Recursive Least Squares Algorithm. *IEEE Transactions on Signal Processing*, (1):88–96, January 1999.

[66] G. Long, F. Ling, and J.G. Proakis. The LMS algorithme with delayed coefficienst adaptation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 1397–1405, September 1989.

[67] E. Martin, J. Toureilles, and C. Nouet. Conception optimisée d'architectures en précision finie pour les applications de traitement du signal. *Traitement du Signal 2001*, 18(1):47–56, 2001.

[68] Mathworks. *System-Level Design Products for DSP and communications*, 2001.

[69] V.J. Matthews. Improved Convergence Analysis of Stochastic Gradient Adaptive Filters Using the Sign Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 450–454, April 1987.

[70] V.J. Matthews and Z. Xie. Fixed-Point Error Analysis of Stochastic Gradient Adaptive Lattice Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 70–80, January 1990.

[71] K. Mayyas and T. Aboulnasr. Leaky LMS Algorithm : MSE Analysis for Gaussian Data. *IEEE Transactions on Signal Processing*, 45:927–934, April 1997.

[72] J.E. Mazo. On the independance theory of equalizer convergence. *Bell Szst. Tech. J.*, 58:963–993, 1979.

[73] P. Meignen. Analyse statistique de systèmes en virgule fixe. Technical report, ENSSAT, Lannion, Septembre 2005.

[74] D. Menard. *Methodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe, sous contrainte de precision*. PhD thesis, Universite de Rennes I, Lannion, Décembre 2002.

[75] D. Menard, D. Chillet, and O. Sentieys. Floating-to-fixed-point conversion for digital signal processors. 2006:Article ID 96421, 25 pages, 2006.

[76] D. Menard, R. Rocher, P. Scalart, and O. Sentieys. SQNR determination in non-linear and non-recursive fixed-point systems. In *XII European Signal Processing Conference (EUSIPCO'04)*, pages 1349–1352, Vienna, Austria, September 2004.

[77] D. Menard and O. Sentieys. A methodology for evaluating the precision of fixed-point systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02)*, Orlando, May 2002.

[78] D. Menard and O. Sentieys. Automatic Evaluation of the Accuracy of Fixed-point Algorithms. In *Design, Automation and Test in Europe 2002 (DATE'02)*, Paris, March 2002.

[79] F. Michaut and M. Bellanger. *Filtrage Adaptatif*. Lavoisier, 2 edition, 2005.

[80] A. Oppenheim and R.W. Schafer. *Discrete Time Signal Processing*. Prentice All Signal Processing series. Prentice All, Upper Saddle River, New Jersey, 2 edition, 1999.

[81] K. Ozeki and T.Umeda. An adaptative filtering algorithm using an orthogonal projection to an affine subspace and its properties. *Electronics and Communications in Japan*, 67:19–27, 1984.

[82] E. Ozer, A.P. Nisbet, and D. Gregg. Stochastic Bitwidth Approximation Using Extreme Value Theory for Customizable Processors. Technical report, TrinityCollege, Dublin, Ireland, October 2003.

[83] S. Prakash and V.V. Rao. Fixed-Point Error Analysis of Radix-4 FFT. *Transactions on Signal Processing*, (2):123–133, 1981.

[84] K.J. Raghunath and K.K. Parhi. Finite-Precision Error Analysis of QRD-RLS and STAR-RLS Adaptive Filters. *IEEE Transactions on Signal Processing*, 45(5):1193–1209, 1997.

[85] N. Revol. Introduction à l'arithmétique par intervalles. *Algorithms and Computer Algebra*, 2004.

[86] R. Rocher, N. Herve, D. Menard, and O. Sentieys. Fixed-point configurable hardware components for adaptive filters. In *IEEE International Symposium on Circuits and Systems (ISCAS'06)*, volume 5, pages 57–60, Kos Island, Grèce, 2006.

[87] R. Rocher, D. Menard, N. Herve, and O. Sentieys. Fixed-point configurable hardware components. 2006:Article ID 23197, 13 pages, 2006.

[88] R. Rocher, D. Menard, P. Scalart, and O. Sentieys. Accuracy evaluation of fixed-point lms algorithm. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, pages 237–240, Montreal, Canada, May 2004.

[89] R. Rocher, D. Menard, O. Sentieys, and P. Scalart. Accuracy evaluation of fixed-point apa algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, pages 57–60, Philadelphie, Etats-Unis, 2005.

[90] S. Roy and P. Banerjee. An algorithm for converting floating-point computations to fixed-point in matlab based fpga design. In *Design Automation Conference (DAC'04)*, pages 484–487, June 2004.

[91] S. Roy and P. Banerjee. An algorithm for trading off quantization error with hardware resources for matlab-based fpga design. *IEEE Transactions on Computers*, pages 886–896, July 2005.

[92] C.G. Samson and V.U. Reddy. Fixed-Point Error Analysis of the Normalized Ladder Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 1177–1191, October 1983.

[93] S.D. Sankaran and A.A. Beex. Convergence Behavior of Affine Projection Algorithms. *IEEE Transactions Acoustic, Speech, and Signal Processing*, 4:1086–1096, 2000.

[94] E.H. Satorius and S.T. Alexander. Channel Equalization using adaptive lattice filters. *IEEE Transactions on Communications*, pages 899–905, June 1979.

[95] H.E. Satorius, S.W. Larisch, S.C. Lee, and L.J. Griffiths. Fixed-point implementation of adaptive digital filters. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'83)*, pages 33–36, 1983.

[96] A.H. Sayed and H.C. Shin. Transient behavior of Affine Projections Algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, volume 52, pages 90–102, 2004.

[97] R. Serizel. Implantation en virgule fixe d'un codeur audio. Master's thesis, ENSSAT, Lannion, Septembre 2006.

[98] M.J. Shensa. Recursive Least-Square Lattice Algorithms. *IEEE Transactions on Automatic Control*, pages 695–702, June 1981.

[99] D.T. Sherwood and N.J. Bershad. Quantization Effects in the Complex LMS Adaptive Algorithm-Linearization Using Dither-Applications. *IEEE Transactions on Circuits ans Systems*, 35:466–470, 1988.

[100] C. Shi. Statistical method for floating-point conversion. Master's thesis, University of California, Berkeley, 2002.

[101] C. Shi. *Floating-point to Fixed-point Conversion.* PhD thesis, University of California, Berkeley, Spring 2004.

[102] C. Shi and R.W. Bodersen. An automated floating-point to fixed-point conversion methodology. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, pages 529–532, 2003.

[103] C. Shi and R.W. Bodersen. A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs. In *IEEE International Symposium on Circuits and Systems (ISCAS'04)*, 2004.

[104] D.T.M Slock. On the Convergence Behaviour of the LMS and the Normalized LMS Algorithms. *IEEE Transactions on Signal Processing*, 41:2811–2825, 1993.

[105] G.R.L. Sohie and L.H. Sibul. Stochastic Convergence Properties of the Adaptive Gradient Lattice. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32:102–107, February 1984.

[106] V. Solo. The Limiting Behavior of LMS. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1909–1922, December 1989.

[107] V. Solo. The Error Variance of LMS Time-Varying Weights. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 40(4):803–813, April 1992.

[108] A. Sripad and D. L. Snyder. A Necessary and Sufficient Condition for Quantization Error to be Uniform and White. *IEEE Trans. ASSP.*, 25(5):442–448, Oct. 1977.

[109] Synopsys. *Converting ANSI-C into Fixed-Point using CoCentric Fixe-Point Designer.* Synopsys Inc., April 2000.

[110] M. Tarrab and A. Feuer. Convergence and performance analysis of normalized LMS Filters with Uncorrelated Gaussian Data. *IEEE Transactions on Information Theory*, 334(4):680–691, July 1988.

[111] L. Tizani. Implantation en virgule fixe d'un recepteur de radiocommunication WCDMA. Technical report, ENSSAT, Lannion, Juin 2006.

[112] J.M. Toureilles. *Conception d'architectures pour le traitement du signal en précision finie.* PhD thesis, Université de Rennes I, Janvier 1999.

[113] R. Uribe and T. Cesear. A methodology for exploring finite-precision effects when solving linear systems of equations with least-squares techniques in fixed-point hardware. In *Proceedings of the Ninth Annual Workshop on High Performance Embedded Computing (HPEC'05)*, September 2005.

[114] S. Wadekar and A. Parker. Accuracy Sensitive Word-Length Selection for Algorithm Optimization. *IEEE/ACM International Conference on Computer Design (ICCAD'98)*, pages 54–61, 1998.

[115] B. Widrow. A Study of Rough Amplitude Quantization by Means of Nyquist Sampling Theory. *RE Transactions on Circuit Theory*, CT-3(4):266–276, Dec. 1956.

[116] B. Widrow. Statistical Analysis of Amplitude Quantized Sampled-Data Systems. *Transactions AIEE, Part. II:Applications and Industry*, 79:555–568, 1960.

[117] B. Widrow. Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings IEEE*, 64:1151–1162, 1976.

[118] B. Widrow, I. Kollár, and M.-C. Liu. Statistical Theory of Quantization. *IEEE Transactions on Instrumentation and Measurement*, 45(2):353–361, Apr. 1996.

[119] N.R. Yousef and A.H. Sayed. Tracking Analysis of the LMF and LMMN Adaptive Algorithms. *Asilomar*, pages 786–790, October 1999.

[120] N.R. Yousef and A.H. Sayed. Fixed-point steady-state analysis of adaptive filters. *International Journal of Adaptive Control and Siganl Processing*, 17:237–258, 2003.

[121] X.H. Yu. A new numerically stable structure for fast RLS adaptive filtering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'90)*, pages 1409–1412, 1990.

[122] U. Zölzer. *Digital Audio Signal Processing.* John Wiley and Sons, August 1997.

Publications

Revues internationales

[1] R. Rocher, D. Menard, N. Herve and O. Sentieys. Fixed-Point Configurable Hardware Components. *Eurasip Journal on Embedded Systems (JES)*, pages : 1-13, 2006.

[2] D. Menard, R. Rocher and O. Sentieys. Automatic Accuracy Evaluation for DSP Fixed-Point Linear Systems. *Submitted to the Journal Theoretical Computer Science,* 2^{nd} *version.*

Conférences internationales

[3] R. Rocher, D. Menard, P. Scalart and O. Sentieys. Accuracy Evaluation of Fixed-Point LMS Algorithm. In *International Conference on Acoustics, Speech and Signal Processing 2004 (ICASSP 2004)*, pages : 237-240, Montreal, May 2004.

[4] D. Menard, R. Rocher, P. Scalart and O. Sentieys. SQNR determination in non-linear and non-recursive fixed-point systems. In *XII European Signal Processing Conference (EUSIPCO 2004)*, pages : 1349-1352, Vienna, September 2004.

[5] R. Rocher, D. Menard, O. Sentieys and P. Scalart. Accuracy Evaluation of Fixed-Point APA Algorithms. In *International Conference on Acoustics, Speech and Signal Processing 2005 (ICASSP 2005)*, pages : 57-60, Philadelphia, March 2005.

[6] R. Rocher, N. Herve, D. Menard and O. Sentieys, Fixed-Point Configurable Hardware Components for Adaptive Filters In *IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, pages : 57-60, Kos Island, March 2006,.

Conférence nationale

[7] R. Rocher, D. Menard, O. Sentieys and P. Scalart. Evaluation de la précision des Algorithmes de Projection Affine en Virgule Fixe. In $20^{ème}$ *colloque GRETSI sur le traitement du signal et des images*, Louvain, Septembre 2005.