UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

RÉSEAU DE NEURONES PROFOND POUR CALIBRER LA RÉCIPROCITÉ ET DÉTECTION SUR FPGA EN MIMO MASSIF DE LA 5G

MÉMOIRE PRÉSENTÉ
COMME EXIGENCE PARTIELLE DE LA

MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
SAMUEL SIROIS

SEPTEMBRE 2021

UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

MAÎTRISE EN GÉNIE ÉLECTRIQUE (M. Sc. A.)

Direction de recherche :

Daniel Massicotte	directeur de recherche
Messaoud Ahmed Ouameur	codirecteur de recherche

Jury d'évaluation

Sébastien Roy	ing. Ph.D. Université de Sherbrooke
Abdellah Chehri	ing. Ph.D. Université du Québec à Chicoutimi

Résumé

Ce travail introduit deux problèmes d'envergure rencontrés dans le contexte du MIMO (« Multiple-Input Multiple-Output ») massif du réseau mobile de 5^{e} génération (5G), à savoir, la détection des symboles transmis par les usagers et la calibration de réciprocité.

Dans un premier temps, un réseau de neurones ayant une architecture interne dédiée à la résolution de systèmes d'équations linéaires est présenté. Dans le contexte de ce travail, il est utilisé pour résoudre le problème de calibration de réciprocité qui consiste tout simplement à estimer de façon implicite les effets du canal de communication entre les antennes de la station de base et les usagers mobiles. Cette étape est cruciale dans la chaîne de télécommunications, car le filtrage spatial, qui permet d'envoyer l'information vers les usagers ciblés, est dépendant de la qualité de la solution obtenue au niveau de la calibration de réciprocité. Les résultats obtenus avec ce nouveau type de réseau neuronal démontrent le potentiel prometteur que cette technique pourrait avoir au niveau de la complexité et de la précision des calculs. Néanmoins, il reste encore beaucoup de chemin à parcourir avant d'obtenir un algorithme mature capable de généraliser et d'obtenir une solution fiable pour tous les scénarios possibles pouvant se présenter au niveau du problème à résoudre.

Dans un deuxième temps, un algorithme d'inversion de matrice particulièrement efficace lorsqu'appliqué au problème de détection du MIMO massif est implémenté sur FPGA (*Field-Programmable Gate Array*) à l'aide du logiciel Vivado HLS (HLS – *High Level Synthesis*). Les résultats obtenus démontrent qu'un débit très élevé de détection des données transmis par les usagers mobiles peut être atteint sous certaines conditions.

Ce mémoire présente donc les idées principales qui se retrouvent dans le développement théorique et dans l'implémentation matérielle d'algorithmes du MIMO massif de la 5G.

Avant-propos

Le simple fait de transmettre de l'information d'une façon qui transcende ce que nos sens peuvent détecter a toujours été fascinant pour moi. Nous vivons dans une société qui tient pour acquis les systèmes modernes de télécommunications dont elle est dépendante. Rien n'est plus normal que de transmettre un message de l'autre côté du globe en une fraction de seconde. Ce nouveau paradigme fait en sorte que la plupart des gens ne s'émerveillent plus devant la prouesse technologique que tout cela représente. J'ai donc choisi de travailler sur le réseau mobile de 5^{e} génération dans le cadre de ma maîtrise afin d'approfondir ma compréhension du domaine des télécommunications et de mieux apprécier la beauté qui se cache derrière cette science qui rend possible ce qui semblait inconcevable il y a de cela quelques décennies.

J'aimerais remercier Daniel Massicotte, mon directeur de maîtrise, et Messaoud Ahmed Ouameur, mon co-directeur de maîtrise, pour le support qu'ils m'ont offert tout le long de ces deux années de travail.

Table des matières

Liste des tableaux

Liste des figures

Liste des abréviations

AMP « Approximate Message Passing »

AP Apprentissage Profond

ASIC « Application Specific Integrated Circuit »

CSI « Channel State Information »

DetNet « Detection Network »

EM Espérance-Maximisation

FPGA « Field-Programmable Gate Array »

Gb/J Gigabits par Joule

GC Gradient Conjugué

GCD Gradient Conjugué Déroulé

GPU « Graphics Processing Unit »

GS Gauss-Seidel

IA Intelligence Artificielle

IoT « Internet of Things »

LISTA « Learned Iterative Shrinkage and Thresholding Algorithm »

LS « Least-Squares »

MAC Multiplication-Accumulation

MIMO « Multiple-Input Multiple-Output »

MMSE « Minimum Mean Square Error »

MRT « Maximum Ratio Transmission »

MSE « Mean Square Error »

NSE « Neumann Series Expansion »

OCD « Optimized Coordinate Gradient Descent »

OFDM « Orthogonal Frequency Division Multiplexing »

RF Radiofréquence

RGMIU « Recursive Gram Matrix Inversion Update »

RTL « Register Transfer Level »

SER « Symbol Error Rate »

SNR « Signal to Noise Ratio »

SVD « Singular Value Decomposition »

Tb/s Térabit par seconde

TDD « Time-Division Duplex »

TLS « Total Least Squares »

VHDL « Very High speed integrated circuit hardware Description Langage »

W Watt

WLS « Weighted Least Squares »

ZF « Zero Forcing »

Chapitre 1 - Introduction

La communication est la fondation même de toutes les civilisations humaines ayant existé au cours de l'histoire. L'ère moderne a introduit les télécommunications hertziennes qui ont permis l'émergence de la société contemporaine. Néanmoins, bien que les prouesses technologiques qui y sont reliées évoluent de façon exponentielle, il en est de même pour les besoins et les exigences des consommateurs. De nos jours, plusieurs domaines d'application comme l'internet des objets (IdO, *IOT – Internet of Things*), l'intelligence artificielle (IA) et le forage de données nécessitent une connexion au réseau internet rapide, fiable, efficace et à large bande passante. Dans bien des cas, les appareils de télécommunications sont physiquement situés à des endroits où l'accès direct à un réseau local n'est pas disponible. Heureusement, il existe des réseaux mobiles couvrant de très grandes portions de territoires et qui permettent à ces appareils de rester connecter en tout temps au monde entier.

Le réseau cellulaire de première génération (réseau 1G) a été conçu dans les années 1980 uniquement pour transmettre la voix des utilisateurs. Ainsi, la plupart des appareils sur ce réseau mobile étaient les premiers téléphones portables. Ensuite, dans les années 1990, la 2G a permis aux utilisateurs d'échanger des messages écrits, ce qui a constitué un changement de paradigme dans la mesure où les réseaux publics sans fil n'étaient plus uniquement utilisés pour la transmission de la voix. Ensuite, il a fallu attendre l'arrivée du réseau 3G au début des années 2000 pour que les usagers aient accès à internet via un réseau mobile sans fil. Au début des années 2010, le réseau 4G a permis la transmission d'un plus grand débit de

données afin de supporter l'arrivée massive des appareils connectés comme les tablettes électroniques et les téléphones intelligents. Avec la croissance du nombre d'objets connectés, un autre changement de paradigme a été nécessaire afin de supporter le trafic de données qui en résulte. Le réseau 5G a donc été pensé et conçu pour répondre à ces nouveaux besoins [1].

Ainsi, le réseau mobile de 5^{e} génération (5G) qui est actuellement en déploiement dans le monde entier repose principalement sur deux technologies clefs. La première est le MIMO massif qui se traduit par une quantité d'antennes beaucoup plus grande à la station de base que le nombre d'usagers mobiles servis en même temps. Cela permet, entre autres, d'augmenter le débit de données pouvant être échangé entre les parties prenantes, d'augmenter la fiabilité des communications, d'améliorer l'efficacité énergétique des transmissions et de réduire les interférences entre les utilisateurs du réseau [2]-[3].

La deuxième technologie utilisée est la modulation OFDM (« Orthogonal Frequency Division Multiplexing ») qui repose principalement sur des algorithmes de FFT (« Fast Fourier Transform ») et de IFFT (« Inverse Fast Fourier Transform ») afin d'encoder et de décoder de façon orthogonale les symboles échangés entre la station de base et les usagers. Cette technique de modulation permet l'utilisation efficace des bandes de fréquences puisque les canaux représentés par chaque sous-porteuse sont espacés selon l'inverse de la période d'un symbole OFDM [4].

Ces deux techniques combinées permettent donc la transmission fiable et efficace d'information à haut débit afin de soutenir, par exemple, les objets connectés de plus en plus nombreux dans la société moderne. Pour une introduction plus détaillée aux systèmes MIMO-OFDM, le lecteur peut se référer à [5]. Cela dit, dans le cadre de ce travail, deux problématiques reliées au MIMO massif du réseau 5G, à savoir, la détection des symboles

transmis par les usagers mobiles et la calibration de réciprocité, sont abordées. Bien qu'il y ait plusieurs perspectives d'approches quant à l'étude du réseau mobile de 5^{e} génération, ce mémoire introduit principalement des algorithmes présents au niveau de la couche physique des calculs.

Ainsi, dans un premier temps, le problème de calibration de réciprocité est présenté et résolu au chapitre 2 en se référant à un outil de calculs de plus en plus populaire et utilisé dans plusieurs domaines d'application, soit l'apprentissage profond (AP, *DL – Deep Learning*). Les contributions récentes de la littérature préconisent le potentiel de son utilisation pour la conception de systèmes de télécommunications [6]-[10]. Par exemple, certaines publications présentent des résultats sur la compression du signal [8] et le décodage des canaux [9] qui ont été obtenus grâce aux outils de pointe de l'AP. De plus, les architectures de traitement de données en parallèle, telles que les unités de traitement graphique (GPU – *Graphics Processing Unit*) se sont avérées très économes en énergie avec des capacités de calculs remarquables lorsqu'elles sont pleinement exploitées par des algorithmes massivement parallélisables [11].

Jusqu'à présent, l'objectif de l'introduction de l'AP dans les systèmes de télécommunications est, soit d'améliorer certaines parties des algorithmes existants, soit de les remplacer complètement de bout en bout [12]-[13]. À titre d'exemple, les auteurs de [6] ont discuté de plusieurs nouvelles applications prometteuses de l'AP au niveau de la couche physique des calculs. Aussi, deux architectures d'AP dédiées à la résolution du problème de détection pour les systèmes MIMO sont introduites dans [12] et reposent sur le déroulement profond des itérations d'un algorithme de descente du gradient projeté. Le déroulement

profond consiste à considérer chaque mise à jour d'un algorithme itératif comme étant une couche d'un réseau neuronal. Ce sujet est abordé en détail au chapitre 2.

Des méthodes de déroulement profond ont été développées avec succès dans [14] et [15] afin de résoudre un système d'équations linéaires. Dans [14], les auteurs ont fait le déroulement profond de l'algorithme itératif « approximate message passing » (AMP) pour résoudre le problème d'inversion linéaire mal conditionnée. Des données d'apprentissage étaient nécessaires pour ajuster les paramètres de chaque couche du réseau neuronal. De façon semblable à [12], les auteurs de [15] ont déroulé les itérations d'un algorithme de descente du gradient projeté et ont appliqué leur réseau de neurones au problème de détection du MIMO massif.

Avec l'avenue du réseau 6G dans plusieurs années, il semble être de plus en plus évident que l'AP y occupera une place centrale au niveau des algorithmes de traitement des données [16]. Certains croient que ce futur réseau mobile aura la capacité d'atteindre des débits de transferts de données atteignant parfois des pointes tournant aux alentours de 1 Tb/s [17]. Bien que ces débits d'échanges d'informations soient encore loin d'être atteints pour le moment, cela démontre l'importance qui doit être accordée à la résolution du problème de détection du MIMO massif de la 5G afin de préparer le terrain pour les réseaux mobiles qui suivront la marche. Ainsi, le chapitre 3 de ce mémoire présentera l'implémentation d'un algorithme de détection sur FPGA. Bien qu'il s'agisse d'une technique linéaire classique qui n'utilise pas l'AP, rien n'empêcherait l'élaboration de futurs travaux de recherches qui incorporeraient cet outil de calculs afin d'améliorer les performances obtenues.

Les méthodes de détection linéaires les plus connues dans la littérature sont le « maximum ratio transmission » (MRT), le « zero forcing » (ZF) et le « minimum mean square

error » (MMSE) [18]. Ces méthodes sont basées sur l'inversion explicite de la matrice de Gram qui présente un ordre de complexité cubique avec ses dimensions. La section 3.2 du chapitre 3 introduira plus en détail le concept de cette matrice. Pour accélérer les calculs, l'inversion matricielle implicite, qui consiste à trouver la solution optimale d'un système d'équations de façon itérative sans inversion directe de matrice, ou bien les techniques d'approximation d'inversion matricielle explicite sont souvent utilisées. La méthode « Neumann series expansion » (NSE) a été proposée dans [19] pour calculer une approximation de l'inverse de la matrice de Gram afin de résoudre le problème de détection. Cette méthode a été fortement utilisée pour sa complexité faible et prévisible puisque l'ordre de l'expansion de la série est généralement égal ou inférieur à trois. Néanmoins, l'algorithme d'inversion implicite de matrice Gauss-Seidel (GS) a été démontré dans [20] pour converger rapidement vers une solution précise avec une complexité de calculs moindre que la méthode NSE. Malheureusement, comme l'algorithme GS est une technique itérative possédant des dépendances inter itérations spéciales, les performances d'implémentations en termes de latences et de débits qui en résultent sont dégradées [21]-[24]. De plus, lorsque le système à résoudre devient plus grand, l'algorithme GS doit effectuer plus d'itérations pour obtenir des résultats précis, ce qui réduit encore davantage le débit maximal possible pour la détection des symboles transmis par les usagers. D'autre part, il a également été démontré que lorsque le nombre d'usagers augmente, l'algorithme NSE d'ordre trois n'est pas suffisant pour obtenir de bons résultats de détection [25].

Les faits décrits plus haut ont tendance à démonter qu'une inversion matricielle exacte et explicite est la meilleure façon de toujours obtenir des résultats précis et une complexité de calculs prévisible. Ce type d'opération est généralement basée sur la décomposition de

Cholesky qui est considérée comme étant la référence du point de vue des performances et des calculs.

Au niveau de l'implémentation matérielle, une conception sur circuit intégré à application spécifique, (ASIC – *Application Specific Integrated Circuit*), basée sur l'algorithme NSE a permis d'obtenir un débit de détection de données de 3.8 Gbit/s pour 128 antennes à la station de base et 8 usagers [26]. Ce même type de détecteur a également été implémenté sur un FPGA Xilinx Virtex-7 dans [19]. Cette conception a permis d'atteindre 600 Mbit/s pour une station de base de 128 antennes desservant encore une fois 8 utilisateurs. De plus, un autre détecteur basé sur l'algorithme du gradient conjugué (GC) proposé dans [27] a permis une réduction significative de la complexité des calculs. Cependant, pour accélérer davantage le taux de convergence et améliorer les performances, une technique hybride basé sur l'algorithme du GC et la méthode de Jacobi a été proposé dans [28]. Les résultats de l'implémentation d'un détecteur basé sur l'algorithme du GC sont présentés dans [21].

Pour des raisons qui sembleront plus évidentes au chapitre 3, l'implémentation sur FPGA de l'algorithme RGMIU (*Recursive Gram Matrix Inversion Update*) présenté dans [29] sera étudié dans le cadre de ce travail. Cette méthode permet d'inverser la matrice de Gram de manière itérative en ajoutant un usager aux calculs à chaque itération pour ainsi obtenir une inversion explicite exacte avec un nombre fini et connu de mises à jour.

Cela dit, d'autres approches récentes pour résoudre le problème de détection sont devenues populaires. Par exemple, l'AP est de plus en plus utilisé pour effectuer cette tâche. D'autres algorithmes dérivés de la descente du gradient sont également efficaces. Par exemple, les auteurs de [25] ont dérivé un algorithme implicite de descente selon le gradient

en coordonnées optimisées (OCD – *Optimized Coordinate Gradient Descent*), ayant des phases de prétraitement et de traitement peu complexes et ont pu atteindre une détection de données de 376 Mbit/s sur FPGA pour 128 antennes à la station de base et 8 usagers.

En résumé, dans le cadre de ce mémoire, le chapitre 2 introduit d'abord le problème de calibration de réciprocité et le processus de déroulement profond de l'algorithme du GC. Par la suite, le réseau de neurones résultant est appliqué au problème de calibration de réciprocité et les résultats obtenus sont comparés à ceux produits par des techniques connues de la littérature. Le chapitre 3 présente le problème de détection du MIMO massif de la 5G pour ensuite comparer les performances en simulations de plusieurs algorithmes connus, dont le RGMIU qui sera par la suite implémenté sur FPGA à l'aide de Vivado HLS (HLS – *High Level Synthesis*). Enfin, les performances potentielles d'implémentation de cette technique en termes de latences, de débits de données, d'utilisation des ressources matérielles et d'efficacité énergétique seront analysées.

Ce mémoire a permis d'apporter les contributions suivantes qui sont présentées en annexe:

S. Sirois, M. Ahmed Ouameur and D. Massicotte, "Deep Unfolded Extended Conjugate Gradient Method for Massive MIMO Processing with Application to Reciprocity Calibration," Journal of Signal Processing Systems, Springer, Feb. 2021, pp. 1-11.

S. Sirois, M. Ahmed Ouameur and D. Massicotte, "High Level Synthesis Strategies for Ultra Fast and Low Latency Matrix Inversion Implementation for Massive MIMO Processing," VLSI Journal, Elsevier, Jan. 2022, pp. 29-36

Chapitre 2 - Déroulement profond pour la calibration de réciprocité

2.1 Mise en contexte du déroulement profond

Le déroulement profond consiste à utiliser la structure d'un algorithme itératif connu, généralement dédié pour résoudre un système d'équations, et à considérer chaque itération comme étant une couche d'un réseau neuronal. Chaque paramètre de la méthode itérative qui était normalement mis à jour avec une règle déterministe est plutôt ajusté à l'aide du processus de rétropropagation pour produire des résultats optimaux dans la résolution du système d'équations. Certains paramètres peuvent être ajoutés ou modifiés lors du déroulement profond de l'algorithme afin de donner plus de degrés de liberté dans le processus d'apprentissage et ainsi permettre au réseau neuronal de prendre en compte certaines particularités du problème qui ne seraient pas considérées par la méthode itérative de départ. Cela peut augmenter la complexité d'une couche par rapport à une itération de l'algorithme d'origine, mais au final, le nombre de couches nécessaires pour résoudre le problème peut être considérablement réduit pour résulter en une complexité et une latence de calculs bien plus petites que ce qui serait obtenu avec la méthode itérative initiale. Ce constat sera présenté dans les résultats à la section 2.7.

En outre, un autre avantage de dérouler un algorithme itératif est qu'une fois la phase d'apprentissage terminée, le réseau neuronal finira par capturer tous les détails inhérents du problème et pourra généralement trouver une solution avec ses paramètres internes fixés,

tandis que ceux de la méthode itérative de départ devront être remis à jour chaque fois que le système à résoudre aura changé. En d'autres termes, une fois que l'apprentissage est fait pour une grande variété de données d'entrées, le réseau déroulé peut généraliser pour les autres données d'entrées. Au besoin, un réapprentissage partiel de certaines couches peut être fait de temps en temps. Néanmoins, le secret réside dans la stratégie d'apprentissage pour s'assurer que les paramètres ne sont pas surentraînés ou sous entraînés. De plus, des fonctions d'activations non linéaires peuvent être ajoutées aux couches de l'algorithme déroulé pour tenir compte des non-linéarités du problème lorsqu'applicables.

Il a été observé que de façon générale, les algorithmes déroulés dans la littérature et utilisés pour résoudre des systèmes d'équations ont besoin d'avoir accès à des données explicites d'apprentissage afin d'être efficaces. Un paramètre d'erreur résiduelle dans l'algorithme du GC peut être exploité pour créer un réseau neuronal pouvant s'entraîner sans nécessiter de données d'entraînement connues. Cela peut être utile dans de nombreuses applications réelles telles que la calibration de réciprocité et la détection qui sont rencontrées dans les systèmes MIMO massifs du réseau 5G. Une architecture du GC déroulée (GCD – Gradient Congugué Déroulé) sera donc présentée dans ce chapitre afin de démontrer l'utilité de cette propriété intéressante (paramètre d'erreur résiduelle) appliquée au problème de calibration de réciprocité.

L'algorithme du GC peut être utilisé pour résoudre itérativement de larges systèmes d'équations linéaires et il est reconnu pour converger rapidement, et ce, même lorsque le système à résoudre est dit mal conditionné. De plus, sans la présence de perturbations externes, cette méthode est assurée d'atteindre une solution en un nombre d'itérations inférieur ou égal au nombre d'inconnus [30]. Il est à noter que le paramètre d'erreur résiduelle

calculée à chaque itération ne dépend pas directement de la solution explicite du problème. Ce fait est exploité dans l'architecture de déroulement profond qui sera proposée plus loin afin de former un réseau de neurones qui ne nécessite pas de données d'apprentissage pour ajuster les valeurs de ses coefficients internes.

2.2 Mise en contexte du problème de calibration de réciprocité

À titre de rappel, le MIMO massif est une des technologies clefs pour la nouvelle génération de réseau mobile également connu sous le nom de réseau 5G [31]-[32]. Dans le contexte de la 5G, il est défini comme étant un système multiusager classique avec chaque station de base possédant un nombre d'antennes beaucoup plus grand que le nombre d'usagers/utilisateurs servis en même temps [33]-[34]. La technologie du MIMO massif fonctionne en mode « Time-Division Duplex » (TDD) et utilise l'information d'état du canal, « Channel State Information » (CSI) en anglais, en liaison montante pour faire la détection des symboles des usagers et pour effectuer du filtrage spatial en liaison descendante, « beamforming » en anglais [35]-[36]. Malheureusement, le canal en liaison montante n'est pas réciproque au canal en liaison descendante en raison des chaines radiofréquences (RF) des émetteurs-récepteurs qui introduisent des effets aléatoires sur le gain et la phase des coefficients de canaux. Cela implique que si le CSI en liaison montante est utilisé pour effectuer du filtrage spatial en liaison descendante, l'efficacité de la transmission des symboles aux usagers s'en verrait grandement affectée [37]-[38]. Le CSI explicite en liaison descendante ne peut malheureusement pas être déterminé directement, car cela nécessiterait que chaque antenne à la station de base envoie des signaux pilotes connus aux usagers, que ces derniers leur renvoient l'information recueillie pour qu'ensuite le CSI en liaison descendante entre chaque antenne de la station de base et chaque usager soit calculé. De plus,

ce processus doit être fait chaque fois que le CSI entre les usagers et la station de base change. Cette manière de procéder serait beaucoup trop gourmande en termes de latences et de temps de calcul pour les applications prévues du réseau 5G. Une façon beaucoup plus rapide de faire consiste à évaluer le CSI en liaison montante et à partir de ces données et d'une calibration interne, il est possible de déduire le CSI en liaison descendante. L'estimation du canal en liaison montante ne nécessite que l'envoi de signaux pilotes connus de la part des usagers qui sont beaucoup moins nombreux que le nombre d'antennes à la station de base. Cette méthode est connue sous le nom de calibration de réciprocité et elle permet donc de calculer implicitement le CSI des canaux en liaison descendante d'un système MIMO massif à partir du CSI en liaison montante [37].

Les détails de l'impact des canaux non réciproques et les solutions possibles pour contrer ce problème sont également présentés dans [37]. Aussi, une introduction plus détaillée au problème de calibration de réciprocité est présentée dans [39] tandis qu'une solution pour la calibration est implémentée sur un banc d'essai réel dans [40].

L'idée principale derrière l'approche la plus simple pour résoudre le problème de calibration de réciprocité est de choisir arbitrairement une antenne de référence à la station de base qui enverra et recevra des signaux pilotes connus à/de chacune des autres antennes présentes à cette même station de base. Ensuite, basée sur la collecte de tous les signaux reçus par chaque antenne, la calibration de réciprocité peut être faite en résolvant un simple problème d'optimisation. Malheureusement, les performances de cette méthode varient énormément en fonction du positionnement de l'antenne de référence et sont très sensibles face aux perturbations externes.

Pour éliminer le problème de positionnement de l'antenne de référence, les auteurs de [38] ont proposé plusieurs estimateurs « least-squares » (LS) généralisés basés sur le couplage mutuel entre chaque paire d'antennes à la station de base. En d'autres termes, chaque antenne peut être vue à tour de rôle comme étant une antenne de référence qui envoie et reçoit des signaux pilotes connus à/de chacune des autres antennes. Avec la collecte de tous ces signaux, un système d'équations linéaires peut être construit à partir de l'estimateur LS et résolu pour déterminer des coefficients de calibration qui serviront à corriger les coefficients du canal en liaison montante afin d'obtenir une estimation implicite du CSI en liaison descendante.

Les auteurs de [41] ont donc proposé une nouvelle façon itérative pour inverser rapidement et efficacement de grandes matrices afin de résoudre le problème du LS relié à la calibration de réciprocité. D'un autre point de vue, les auteurs de [42] ont formulé un algorithme d'espérance-maximisation (EM – *Expectation–Maximization*) qui améliore de façon itérative les résultats obtenus avec d'autres méthodes comme celle du LS généralisé.

La formulation de l'estimateur LS dans le contexte de la calibration de réciprocité d'un système MIMO massif implique l'inversion d'une matrice de dimensions $(M-1)\times(M-1)$ afin de résoudre un système d'équations linéaires, où M représente le nombre d'antennes à la station de base. Même si, en théorie, la calibration doit être faite approximativement chaque heure pour un système MIMO centralisé (même horloge de référence pour tout le système) [37], il en est tout autrement pour les systèmes MIMO distribués où les horloges de références ne sont pas parfaitement synchronisées entre elles de sorte qu'une calibration doit être faite aux quelques secondes pour éviter des erreurs de phases lors des calculs [43]. Un algorithme de calibration rapide et efficace est donc nécessaire pour répondre aux besoins

des systèmes MIMO distribués. Cela dit, une méthode basée sur le GCD semblerait appropriée, car une fois le processus d'apprentissage terminé, le réseau neuronal ainsi créé peut théoriquement généraliser pour d'autres entrées et arriver très rapidement à une solution pour le processus de calibration.

Il existe plusieurs techniques itératives efficaces afin de résoudre des systèmes d'équations linéaires. Ces méthodes sont généralement très pratiques lorsque des perturbations sont présentes d'un seul côté du système d'équations. Dans la formulation de l'estimateur LS inhérent à la calibration de réciprocité, un bruit parasite est présent de chaque côté du signe d'égalité du système d'équations à résoudre. Ainsi, les algorithmes de résolutions classiques comme ceux de GS et de NSE requièrent beaucoup d'itérations afin de converger vers une bonne solution [41].

La raison pour laquelle l'algorithme du GC a été choisi pour être déroulé à la place d'une autre méthode itérative en particulier est dû en partie au fait qu'à chaque itération, un paramètre d'erreur résiduelle peut facilement être utilisé pour calculer une fonction de perte indépendante de la solution explicite du problème. Un seuil peut être ajusté sur ce paramètre afin d'indiquer au réseau neuronal ainsi déroulé quand arrêter son processus d'apprentissage. Cette propriété peut être très utile dans certains problèmes comme celui de la calibration de réciprocité où des données explicites d'entraînement ne sont pas disponibles. De plus, l'algorithme du GC est très populaire pour résoudre de larges systèmes d'équations linéaires dû à sa convergence rapide et fiable vers une bonne solution.

Cela dit, un réseau de neurones standard pourrait également être utilisé pour résoudre le problème de la calibration de réciprocité, mais l'objectif principal recherché ici est de voir si le GCD, qui possède une architecture interne dédiée à la résolution de systèmes d'équations

linéaires, peut offrir des performances acceptables, voir au-delà des attentes, pour résoudre des problèmes plus spécifiques reliés au MIMO massif de la 5G par exemple. Il est très important de comprendre que les performances obtenues avec le déroulement profond du GC ne seront pas comparées à celles qui auraient été obtenues à l'aide d'un réseau de neurones standard. En effet, comme il sera vu plus loin, le GCD présente des résultats prometteurs, mais il y a encore beaucoup de chemin à faire avant d'obtenir une technique mature et fiable. En d'autres termes, les résultats qui seront présentés dans ce mémoire ne sont que la pointe de l'iceberg dans la recherche pour l'optimisation et l'utilisation du déroulement profond d'algorithmes itératifs dédiés à la résolution de problèmes spécifiques et il serait injuste de comparer immédiatement les résultats observés avec ceux qui seraient obtenus avec un réseau de neurones classique dont le développement théorique et pratique est assez complet.

Certains peuvent également argumenter que l'avantage indéniable d'un réseau de neurones standard est l'utilisation de fonctions d'activations non linéaires à chaque couche qui permettent de capturer les détails parfois complexes du problème à résoudre, mais rien n'empêcherait d'ajouter de telles fonctions à chaque itération lors du processus de déroulement profond d'un algorithme itératif. Néanmoins, dans un souci de simplicité, aucune fonction d'activation non linéaire n'a été utilisée dans le réseau qui sera présenté aux sections suivantes.

2.3 Déroulement profond du gradient conjugué

Dans le contexte de la calibration de réciprocité, considérons la résolution d'un système d'équations linéaires à valeurs réelles de la forme $\mathbf{Ax} = \mathbf{y}$ avec x contenant de l'information à propos des coefficients de calibration et avec A et y étant dérivés d'un estimateur LS. $\mathbf{A}$ est une matrice carré symétrique définie positive de dimensions $(2M-2)\times(2M-2)$, où M

représente le nombre d'antennes à la station de base dans le contexte du MIMO massif de la 5G et où le facteur multiplicatif de deux provient du fait que les parties réelles et imaginaires des nombres complexes impliqués dans les calculs sont séparées afin d'obtenir un système à valeurs réelles uniquement. La formulation mathématique du problème de calibration de réciprocité sera vue en détails à la section 2.5. Afin de résoudre ce système d'équations, l'algorithme itératif du GC requiert d'abord les initialisations suivantes [30] :

$$\hat{\mathbf{x}}_1 = \textit{Estimation Initiale} \tag{2.1}$$

$$\mathbf{r}_1 = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}_1 \tag{2.2}$$

$$\mathbf{s}_1 = \mathbf{A}\mathbf{r}_1 \tag{2.3}$$

$$\mathbf{p}_1 = \mathbf{s}_1 \tag{2.4}$$

$$\gamma_1 = \|\mathbf{s}_1\|^2 \tag{2.5}$$

où $\hat{\mathbf{x}}_1$ est la meilleure estimation possible, *a priori*, de la solution pour $\mathbf{x}$. Une fois ces initialisations terminées, l'algorithme du GC pour la $k^{ième}$ itération est défini comme suit pour k=1, 2, ... :

$$\mathbf{q}_k = \mathbf{A}\mathbf{p}_k \tag{2.6}$$

$$\alpha_k = \frac{\gamma_k}{\|\mathbf{q}_k\|^2} \tag{2.7}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k \tag{2.8}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k \tag{2.9}$$

$$\mathbf{s}_{k+1} = \mathbf{A}\mathbf{r}_{k+1} \tag{2.10}$$

$$\gamma_{k+1} = \|\mathbf{s}_{k+1}\|^2 \tag{2.11}$$

$$\mathbf{p}_{k+1} = \mathbf{s}_{k+1} + \frac{\gamma_{k+1}}{\gamma_k}\mathbf{p}_k \tag{2.12}$$

En appliquant le déroulement profond à la méthode du GC, les équations (2.6) à (2.12) peuvent être utilisées pour créer l'architecture d'une couche d'un réseau neuronal. En analysant davantage les étapes d'une itération, il est facile de réaliser que les équations (2.8) et (2.9) sont indépendantes. De plus, lorsque la norme du vecteur de l'erreur résiduelle $\mathbf{r}$ tend vers zéro, la valeur de $\hat{\mathbf{x}}$ tend vers la solution $\mathbf{x}$ du problème $\mathbf{Ax} = \mathbf{y}$ [30]. Ces deux propriétés intéressantes permettent de dérouler l'algorithme du GC tout en omettant l'équation (2.8). L'apprentissage du réseau de neurones se fera en minimisant une fonction d'erreur basée sur la norme du vecteur d'erreur résiduelle $\mathbf{r}$ de la dernière couche. Une fois le processus d'entraînement terminé, l'équation (2.8) peut être utilisée pour trouver la solution finale au problème.

Cela dit, les paramètres qui seront ajustés durant le processus d'apprentissage sont α_k et $\boldsymbol{\lambda}_k \triangleq \gamma_{k+1}/\gamma_k$, noter que $\boldsymbol{\lambda}_k$ est une matrice qui sera expliquée plus loin. Ainsi, les équations (2.7) et (2.11) peuvent être retirées lors du déroulement profond puisque ces deux paramètres (α_k et $\boldsymbol{\lambda}_k$) sont ajustés à l'aide du processus de rétropropagation à la place d'être calculés de façon explicite par une équation déterministe.

Avant de continuer, certaines conventions utilisées dans le problème de calibration de réciprocité doivent être présentées. Le problème original implique, comme il sera vu plus loin, la résolution d'un système d'équations linéaires à valeurs complexes de la forme :

$$\overline{\mathbf{A}}\,\overline{\mathbf{x}} = \overline{\mathbf{y}} \tag{2.13}$$

avec $\bar{\mathbf{A}}$ étant une matrice de dimensions $(M)\times(M-1)$ composée d'éléments à valeur complexe. L'astuce suivante sera utilisée pour transformer ce problème à valeurs complexes en un problème ne possédant que des valeurs réelles :

$$\breve{\mathbf{y}} = \begin{pmatrix} \mathrm{Re}(\bar{\mathbf{y}}) \\ \mathrm{Im}(\bar{\mathbf{y}}) \end{pmatrix} \in \mathbb{R}^{2M\times 1} \tag{2.14}$$

$$\breve{\mathbf{A}} = \begin{pmatrix} \mathrm{Re}(\bar{\mathbf{A}}) & -\mathrm{Im}(\bar{\mathbf{A}}) \\ \mathrm{Im}(\bar{\mathbf{A}}) & \mathrm{Re}(\bar{\mathbf{A}}) \end{pmatrix} \in \mathbb{R}^{2M\times(2M-2)} \tag{2.15}$$

$$\mathbf{x} = \begin{pmatrix} \mathrm{Re}(\bar{\mathbf{x}}) \\ \mathrm{Im}(\bar{\mathbf{x}}) \end{pmatrix} \in \mathbb{R}^{(2M-2)\times 1} \tag{2.16}$$

où Re(·) et Im(·) correspondent respectivement à la partie réelle et à la partie imaginaire de l'expression entre parenthèses. Puisque l'algorithme du gradient conjugué ne peut résoudre que les systèmes d'équations ayant une matrice carrée symétrique définie positive, les manipulations suivantes doivent être effectuées :

$$\mathbf{A} = \breve{\mathbf{A}}^T\breve{\mathbf{A}} \in \mathbb{R}^{(2M-2)\times(2M-2)} \tag{2.17}$$

$$\mathbf{y} = \breve{\mathbf{A}}^T\breve{\mathbf{y}} \in \mathbb{R}^{(2M-2)\times 1} \tag{2.18}$$

Avec ces conventions d'établies, il est possible d'écrire :

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{2.19}$$

Par la suite, il ne suffit que de résoudre pour $\mathbf{x}$ afin de retrouver la forme complexe $\bar{\mathbf{x}}$ à l'aide de l'équation (2.16).

De façon plus formelle, pour dérouler l'algorithme du GC, les équations (2.6) et (2.9) sont combinées, les équations (2.8) et (2.9) sont utilisées sans modification et le terme

scalaire $\frac{\gamma_{k+1}}{\gamma_k}$ dans l'équation (2.12) est remplacé par la matrice $\boldsymbol{\lambda}_k$ qui sera expiquée au prochain paragraphe. Cela donne les équations suivantes qui décrivent l'architecture de la $k^{ième}$ couche du réseau neuronal ainsi créé :

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k \tag{2.20}$$

$$\mathbf{s}_k = \mathbf{A}\mathbf{r}_{k+1} \tag{2.21}$$

$$\mathbf{p}_{k+1} = \mathbf{s}_k + \boldsymbol{\lambda}_k \mathbf{p}_k \tag{2.22}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k \tag{2.23}$$

où $\mathbf{r}$ est l'erreur résiduelle de dimensions $(2M-2)\times 1$ sur l'estimation du paramètre inconnu, $\hat{\mathbf{x}}$, $\mathbf{p}$ et $\mathbf{s}$ sont des variables intermédiaires de dimensions $(2M-2)\times 1$, α est un paramètre scalaire et $\boldsymbol{\lambda}$ est une matrice de dimensions $(2M-2)\times(2M-2)$. Les valeurs des variables α et $\boldsymbol{\lambda}$ sont les éléments qui devront être déterminés durant le processus d'apprentissage du réseau de neurones. Normalement, $\boldsymbol{\lambda}$ est représenté par une valeur scalaire dans l'algorithme classique du GC, mais le fait de le transformer en une matrice permet d'augmenter le nombre de degrés de liberté durant le processus d'entraînement de sorte que le réseau neuronal résultant pourra capturer beaucoup plus de détails inhérents au problème à résoudre afin de converger plus rapidement vers un résultat acceptable. Bien entendu, tel que mentionné précédemment, cela a comme inconvénient d'augmenter la complexité d'une couche par rapport à une itération normale de l'algorithme initial, mais au final, ce nombre total de couches peut se voir drastiquement réduit pour donner une complexité globale très faible. Les conditions initiales du GCD sont définies par :

$$\mathbf{r}_1 = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}_1 \tag{2.24}$$

$$\mathbf{p}_1 = \mathbf{A}\mathbf{r}_1 \tag{2.25}$$

avec l'estimation initiale $\hat{\mathbf{x}}_1$ qui peut être le vecteur nul.

La figure 1 à la page suivante présente l'arbre de dépendance pour l'algorithme du GCD. La fonction de perte e est basée sur la norme L_2 appliquée sur $\mathbf{r}_{n+1}$, tel que présentée ci-dessous :

$$e = \text{Loss}(\Theta; \mathbf{A}, \mathbf{x}) = \|\mathbf{r}_{n+1}\|_2^2 \tag{2.26}$$

$$\Theta = \{\alpha_k, \boldsymbol{\lambda}_k\}_{k=1}^{n \text{ pour } \alpha \text{ et } (n-1) \text{ pour } \lambda} \tag{2.27}$$

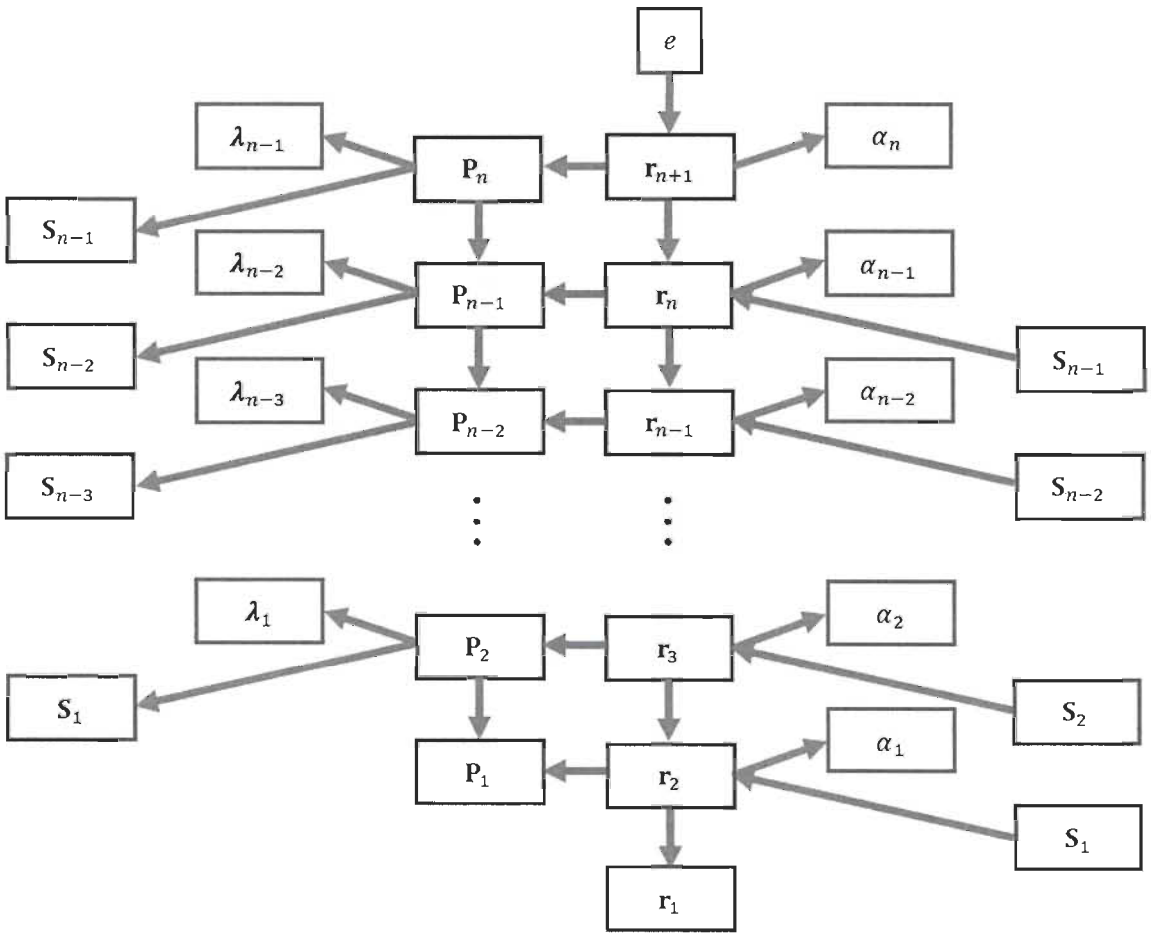


Figure 1. Arbre de dépendance du GCD

La fonction de perte est donc minimisée par les paramètres $\Theta = \{\alpha_k, \boldsymbol{\lambda}_k\}_{k=1}^{n \text{ pour } \alpha \text{ et } (n-1) \text{ pour } \lambda}$. Lorsque l'erreur résiduelle $\mathbf{r}_{n+1}$ tend vers le vecteur nul et donc que la fonction de perte approche zéro, l'erreur sur l'estimation du vecteur $\mathbf{x}$ devient pratiquement inexistante [30]. Ce fait est le concept clef de l'algorithme du GC qui a fait en sorte qu'il a été choisi pour être déroulé. Ainsi, comme il sera vu plus loin, cette astuce permet à l'algorithme d'apprentissage de décider quand arrêter l'entraînement du réseau neuronal en se basant sur un seuil limite appliqué à la fonction de perte. En plus, puisque cette dernière ne dépend pas explicitement de $\mathbf{x}$, cela rend l'architecture proposée aveugle. La dérivation du gradient de la fonction de perte par rapport à α et $\boldsymbol{\lambda}$ ainsi que la méthode d'entrainement utilisée pour le réseau neuronal sont présentées à la section suivante.

2.4 Dérivation du gradient de la fonction de perte du GCD

Le lecteur doit se référer à l'arbre de dépendance présenté à la figure 1 et aux équations (2.20), (2.21) et (2.22) pour comprendre le raisonnement qui suivra. Les formes explicites du gradient de la fonction de perte e par rapport à α_k et $\boldsymbol{\lambda}_k$ sont :

$$\overline{\frac{\partial e}{\partial \alpha_k}} = 2\mathbf{r}_{n+1}^T \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_k}} \tag{2.28}$$

$$\overline{\frac{\partial e}{\partial \boldsymbol{\lambda}_k}} = 2\sum_{z=1}^{2M} \mathbf{r}_{n+1}^{(z)} \overline{\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}} \tag{2.29}$$

où $\overline{\frac{\partial(\cdot)}{\partial(\cdot)}}$ représente la dérivée totale et $\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}$ représente la dérivée totale du z^{ieme} élément de $\mathbf{r}_{n+1}$ par rapport à $\boldsymbol{\lambda}_k$, où $k \in [1:n]$ pour l'équation (A.1) et $k \in [1:n-1]$ pour l'équation

(A.2). Dans un premier temps, La formule qui décrit la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à α_n est tout simplement :

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_n}} = \frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_n} \tag{2.30}$$

puisque le seul chemin possible allant de $\mathbf{r}_{n+1}$ vers α_n sur la figure 1 est celui qui relie directement les deux éléments. Il est important de mentionner que les chemins valides correspondent à ceux qui peuvent relier deux nœuds tout en suivant le sens des flèches sur la figure 1. Selon cette logique, il est possible de déduire que :

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_k}} = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{k+1}}} \frac{\partial \mathbf{r}_{k+1}}{\partial \alpha_k} \tag{2.31}$$

où la règle de dérivation en chaîne est appliquée entre la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{r}_{k+1}$, qui dépend de tous les chemins possibles reliant ces deux éléments et de la dérivée correspondant au chemin unique allant de $\mathbf{r}_{k+1}$ à α_k avec $k \in [1 : n-1]$. De façon similaire avec $\boldsymbol{\lambda}_k$, il est simple d'obtenir :

$$\overline{\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}}(i,j) = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_{k+1}}}(i,z) \frac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_k}(i,j) \tag{2.32}$$

où les deux indices entre parenthèses à côté de chaque terme représentent la ligne et la colonne de la matrice correspondante avec $\mathrm{i,j,z} \in [1 : 2\mathrm{M}]$ et $k \in [1 : n-1]$. Le côté gauche de l'équation (A.5) montre qu'il y a une dérivée distincte pour chaque élément présent dans le vecteur $\mathbf{r}_{n+1}$ par rapport à chaque élément (i,j) se retrouvant dans la matrice $\boldsymbol{\lambda}_k$. La raison pour laquelle la forme de l'équation (A.5) est légèrement différente de celle retrouvée dans

l'équation (A.4) est due au fait que $\boldsymbol{\lambda}_k$ représente une matrice à la place d'être un simple scalaire. Il est ensuite possible de continuer avec la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{p}_n$:

$$\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{p}_n} = \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n} \tag{2.33}$$

qui correspond au seul chemin possible entre ces deux éléments. Cela se complique légèrement pour la forme générale de la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{p}_k$ avec $k \in [2 : n-1]$:

$$\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{p}_k} = \frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{p}_{k+1}} \left(\frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{p}_k} + \frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{s}_k} \frac{\partial \mathbf{s}_k}{\partial \mathbf{r}_{k+1}} \frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k} \right) + \frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{k+1}} \frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k} \tag{2.34}$$

où, dans un premier temps, la règle de dérivation en chaîne est appliquée entre la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{p}_{k+1}$ et le seul chemin possible entre $\mathbf{p}_{k+1}$ et $\mathbf{p}_k$. Ensuite de façon similaire, la règle de dérivation en chaîne est appliquée entre la même dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{p}_{k+1}$ et le chemin passant, dans l'ordre, par $\mathbf{p}_{k+1}$, $\mathbf{s}_k$, $\mathbf{r}_{k+1}$ et $\mathbf{p}_k$. Finalement, le même processus est réalisé entre la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{r}_{k+1}$ et le chemin unique entre $\mathbf{r}_{k+1}$ et $\mathbf{p}_k$. À ce stade, le seul inconnu restant correspond à la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à n'importe quel autre $\mathbf{r}$. En ce sens, il est d'abord simple de trouver :

$$\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_n} = \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_n} + \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n} \frac{\partial \mathbf{p}_n}{\partial \mathbf{s}_{n-1}} \frac{\partial \mathbf{s}_{n-1}}{\partial \mathbf{r}_n} \tag{2.35}$$

qui peut ensuite être insérée dans l'équation décrivant la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{r}_{n-1}$:

$$\begin{aligned}\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{n-1}} &= \frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{n}}\left(\frac{\partial \mathbf{r}_{n}}{\partial \mathbf{r}_{n-1}}+\frac{\partial \mathbf{r}_{n}}{\partial \mathbf{p}_{n-1}}\frac{\partial \mathbf{p}_{n-1}}{\partial \mathbf{s}_{n-2}}\frac{\partial \mathbf{s}_{n-2}}{\partial \mathbf{r}_{n-1}}\right) \\ &+\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_{n}}\frac{\partial \mathbf{p}_{n}}{\partial \mathbf{p}_{n-1}}\frac{\partial \mathbf{p}_{n-1}}{\partial \mathbf{s}_{n-2}}\frac{\partial \mathbf{s}_{n-2}}{\partial \mathbf{r}_{n-1}}\end{aligned} \tag{2.36}$$

où les trois chemins possibles entre $\mathbf{r}_{n+1}$ et $\mathbf{r}_{n-1}$ sont considérés. Finalement, il devient possible de trouver par simple récurrence la dérivée totale de $\mathbf{r}_{n+1}$ par rapport à $\mathbf{r}_k$ avec $k \in [2 : n-2]$:

$$\begin{aligned}\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{k}} &= \frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{k+1}}\left(\frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{r}_{k}}+\frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_{k}}\frac{\partial \mathbf{p}_{k}}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_{k}}\right) \\ &+\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_{n}}\left(\prod_{i=n}^{k+1}\frac{\partial \mathbf{p}_{i}}{\partial \mathbf{p}_{i-1}}\right)\frac{\partial \mathbf{p}_{k}}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_{k}} \\ &+\sum_{j=1}^{n-k-1}\left(\frac{\overline{\partial \mathbf{r}_{n+1}}}{\partial \mathbf{r}_{n-j+1}}\frac{\partial \mathbf{r}_{n-j+1}}{\partial \mathbf{p}_{n-j}}\prod_{z=n-j}^{k+1}\left(\frac{\partial \mathbf{p}_{z}}{\partial \mathbf{p}_{z-1}}\right)\right)\frac{\partial \mathbf{p}_{k}}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_{k}}\end{aligned} \tag{2.37}$$

où tous les chemins possibles allant de $\mathbf{r}_{n+1}$ à $\mathbf{r}_k$ sont pris en compte. En prenant les dérivées partielles des équations (2.20), (2.21) et (2.22), chaque dérivée symbolique dans les équations (A.1) à (A.10) peut être substituée par sa vraie valeur :

$$\begin{cases}\dfrac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{r}_{k}} = \mathbf{I}, \dfrac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_{k}} = -\alpha_k \mathbf{A} \\ \dfrac{\partial \mathbf{r}_{k+1}}{\partial \alpha_{k}} = -\mathbf{A}\mathbf{p}_k, \dfrac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_{k}} = \mathbf{A} \\ \dfrac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{p}_{k}} = \boldsymbol{\lambda}_k, \dfrac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{s}_{k}} = \mathbf{I} \\ \dfrac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_{k}} = \begin{pmatrix} \mathbf{p}_k^{(1)} & \cdots & \mathbf{p}_k^{(2M)} \\ \vdots & \vdots & \vdots \\ \mathbf{p}_k^{(1)} & \cdots & \mathbf{p}_k^{(2M)} \end{pmatrix} \in \mathbb{R}^{2M \times 2M}\end{cases} \tag{2.38}$$

où $\mathbf{I}$ est une matrice identité de dimensions $2M \times 2M$ et $\frac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_k}$ est une matrice dont la i^{eme} colonne contient une valeur constante correspondante au i^{eme} élément de $\mathbf{p}_k$.

Afin d'obtenir de bons résultats qui convergent rapidement durant le processus de rétropropagation, les pas d'adaptation pour α_k et $\boldsymbol{\lambda}_k$ sont ajustés à différentes valeurs et la technique du momentum est utilisée sur $\boldsymbol{\lambda}_k$:

$$\boldsymbol{\lambda}_k(t+1) = \boldsymbol{\lambda}_k(t) - \mu_1 \overline{\frac{\partial e}{\partial \boldsymbol{\lambda}_k(t)}} + \beta\left(\boldsymbol{\lambda}_k(t) - \boldsymbol{\lambda}_k(t-1)\right) \tag{2.39}$$

$$\alpha_k(t+1) = \alpha_k(t) - \mu_2 \overline{\frac{\partial e}{\partial \alpha_k(t)}} \tag{2.40}$$

Où μ_1 et μ_2 représentent les pas d'adaptations fixes, β est le coefficient de momentum pouvant être ajusté entre 0 et 1 et t correspond à l'itération courante du processus de rétropropagation.

2.5 Formulation mathématique de la calibration de réciprocité

Cette section résume la formulation du problème de la calibration de réciprocité. Dans le contexte de la 5G, le but recherché est de calculer avec le plus de précision possible les coefficients de calibration qui permettront d'estimer les effets du canal en liaison descendante (station de base vers usagers) à partir des coefficients connus du canal en liaison montante (usagers vers station de base). En d'autres termes, il s'agit d'une estimation implicite des coefficients du canal en liaison descendante. La formulation du problème à résoudre se fait via un estimateur LS. Pour plus de détails sur ce sujet, le lecteur peut se référer à [38], [40] et [41]. Les coefficients de calibration $b^{(m)}$, où $m = 1, \cdots, M$, sont estimés avec chaque

antenne de la station de base qui, à tour de rôle, envoie un signal pilote $s_p = +1$ aux autres antennes voisines. On dénote $y^{(m,l)}$ le signal reçu par l'antenne m et transmis par l'antenne l. Lorsque groupés par paires correspondantes, les signaux échangés entre deux antennes peuvent s'écrire [13] :

$$\begin{pmatrix} y^{(l,m)} \\ y^{(m,l)} \end{pmatrix} = \alpha^{(l,m)} \begin{pmatrix} b^{(l)} \\ b^{(m)} \end{pmatrix} + \begin{pmatrix} n^{(l,m)} \\ n^{(m,l)} \end{pmatrix} \tag{2.41}$$

où $\alpha^{(l,m)} = t_{SB}^{(l)} t_{SB}^{(m)} \tilde{h}^{(l,m)} = t_{SB}^{(m)} t_{SB}^{(l)} \tilde{h}^{(m,l)}$, $t_{SB}^{(m)}$ et $t_{SB}^{(l)}$ sont les coefficients à valeurs complexes (phase et gain) qui modélisent les transmetteurs RF des antennes m et l respectivement, $\tilde{h}^{(l,m)} = \tilde{h}^{(m,l)}$ représentent les coefficients complexes du canal de propagation réciproque entre les antennes m et l et finalement, $\left(n^{(l,m)} \quad n^{(m,l)}\right)^T$ est un vecteur de bruit Gaussien complexe circulairement symétrique de moyenne nulle et de variance N_0. Il est important de prendre note que les coefficients complexes $r_{SB}^{(m)}$ et $r_{SB}^{(l)}$ qui modélisent les récepteurs RF des antennes m et l respectivement sont intégrés dans les coefficients de calibration, d'où le fait qu'ils n'apparaissent pas explicitement dans l'équation (2.41).

Afin de modéliser le canal réciproque $\tilde{h}^{(l,m)} = \tilde{h}^{(m,l)}$, les auteurs de [38] ont déterminé de façon empirique une équation basée sur des mesures faites dans une chambre anéchoïque. Les effets du canal entre les antennes m et l peuvent donc être approximés par :

$$\tilde{h}^{(l,m)} = \beta^{(l,m)} \exp\left(j\phi^{(l,m)}\right) + w^{(l,m)} \tag{2.42}$$

où $\beta^{(l,m)} = 0.03\left(d^{(l,m)}\right)^{-3.7}$, la distance $d^{(l,m)}$ entre les antennes est en multiples de demi-longueur d'onde, la phase $\phi^{(l,m)}$ est distribuée uniformément sur $[0, 2\pi[$ et la partie $w^{(l,m)}$ est

un bruit Gaussien complexe circulairement symétrique de moyenne nulle et de variance N_w qui sert à modéliser l'effet de diffusion due aux éléments lointains dans l'environnement.

Une estimation simple des coefficients de calibration $\hat{\mathbf{b}} = \left(\hat{b}^{(0)}, \quad \hat{b}^{(1)}, \ldots, \quad \hat{b}^{(M-1)}\right)^T$ peut être calculée en utilisant la méthode du chemin direct. Pour se faire, il suffit de choisir une antenne de référence à la station de base, dénommée *ref*, et de calculer le quotient suivant :

$$\hat{b}^{(m)} = y^{(ref,m)} / y^{(m,ref)} \tag{2.43}$$

tout en assumant $\hat{b}^{(ref)} = 1$. Ce résultat est dérivé d'un estimateur LS basé sur l'équation (2.41). Le fait de fixer $\hat{b}^{(ref)} = 1$ n'impacte en rien la qualité des coefficients de calibration, car l'estimation de ces valeurs peut se faire à une constante près sans affecter le filtrage spatial qui dépend de l'estimation implicite du canal en liaison descendante pour envoyer l'information vers les usagers ciblés [38].

Cela dit, la méthode du chemin direct peut être généralisée avec chaque antenne à la station de base qui agit à tour de rôle comme antenne de référence afin d'augmenter la diversité des calculs et d'avoir un meilleur estimateur LS. En d'autres termes, toutes les combinaisons possibles de l'équation (2.41) sont utilisées afin de minimiser la fonction de coût suivante :

$$J\left(\hat{\mathbf{b}}\right) = \sum_{m,l \neq m} \left\| \hat{b}^{(m)} y^{(m,l)} - \hat{b}^{(l)} y^{(l,m)} \right\|^2 \tag{2.44}$$

qui mène à la solution explicite :

$$\hat{\mathbf{b}} = \left(\hat{b}^{(1)}, \quad \hat{b}^{(2)}, \ldots, \quad \hat{b}^{(M-1)}\right)^T = -\left(\tilde{\mathbf{A}}_1^H \tilde{\mathbf{A}}_1\right)^{-1} \tilde{\mathbf{A}}_1^H \tilde{\mathbf{a}}_1 \tag{2.45}$$

où $\tilde{\mathbf{A}} \triangleq \left[\tilde{\mathbf{a}}_1 \quad \tilde{\mathbf{A}}_1\right]$ et $\hat{b}^{(0)}$ est imposé à 1 pour éviter la solution $\hat{\mathbf{b}} = \mathbf{0}$. La matrice $\tilde{\mathbf{A}}$ est définie par :

$$\tilde{\mathbf{A}}^{(m,l)} = \begin{cases} \sum_{l=0}^{M-1} \left| y^{(m,l)} \right|^2, & m = l \\ -y^{*(m,l)} y^{(l,m)}, & m \neq l \end{cases} \tag{2.46}$$

D'autres types d'estimateurs comme celui du « weighted least squares » (WLS) sont également présentés dans [13].

En manipulant l'équation (2.45), il est possible d'arriver à $\tilde{\mathbf{A}}_1 \hat{\mathbf{b}} = -\tilde{\mathbf{a}}_1$ qui possède la même forme et les mêmes dimensions que l'équation (2.13). C'est donc à partir de cette expression que la convention des équations (2.14) à (2.19) sera appliquée afin de dériver l'algorithme du GCD. Le but de convertir un problème à valeurs complexes vers un problème à valeurs réelles est de faciliter le développement du gradient de la fonction de perte qui est nécessaire pour le processus de rétropropagation.

Lorsqu'un bruit Gaussien est présent seulement dans le vecteur $\tilde{\mathbf{a}}_1$, l'estimateur LS est connu pour donner une solution optimale pour $\hat{\mathbf{b}}$ [44]. Malheureusement, dans le contexte du problème de calibration de réciprocité, du bruit est présent dans $\tilde{\mathbf{a}}_1$ et $\tilde{\mathbf{A}}_1$. Une technique beaucoup plus robuste dans ce genre de situation est la méthode du « total least squares » (TLS) spécialement conçue pour ce genre de problème d'estimation [45]. Pour faire simple, l'estimateur TLS permet de trouver une valeur de $\hat{\mathbf{b}}$ de telle sorte que la quantité :

$$\left\| \Delta \tilde{\mathbf{A}}_1 \right\|^2 + \left\| \Delta \tilde{\mathbf{a}}_1 \right\|^2 \tag{2.47}$$

sujette à la contrainte :

$$\left(\tilde{\mathbf{A}}_1 - \Delta\tilde{\mathbf{A}}_1\right)\tilde{\mathbf{b}} = \left(-\tilde{\mathbf{a}}_1 - \Delta\tilde{\mathbf{a}}_1\right) \tag{2.48}$$

est minimisée. $\Delta\tilde{\mathbf{A}}_1$ et $\Delta\tilde{\mathbf{a}}_1$ sont définis comme étant des perturbations présentes sur $\tilde{\mathbf{A}}_1$ et $\tilde{\mathbf{a}}_1$ respectivement. Pour obtenir la solution du TLS, une décomposition en valeurs singulières, (SVD – *Singular Value Decomposition*), de la matrice augmentée $\left(\tilde{\mathbf{A}}_1 \quad \tilde{\mathbf{a}}_1\right)$ doit être faite afin d'en extraire le vecteur singulier de droite $\mathbf{v} = \left(v^{(0,M-1)}, \quad v^{(1,M-1)}, \ldots, \quad v^{(M-1,M-1)}\right)^T$ correspondant à la plus petite valeur singulière de la décomposition. Afin de s'assurer que $\hat{b}^{(0)} = 1$, l'estimation des coefficients de calibration est donnée sous la forme :

$$\hat{\mathbf{b}} = \frac{1}{v^{(0,M-1)}}\mathbf{v} \tag{2.49}$$

Il sera démontré plus loin que les estimateurs LS et TLS performent de façon équivalente lorsque le rapport signal sur bruit, (SNR – *Signal to Noise Ratio*) est élevé, car les perturbations deviennent moins importantes dans le système d'équations. Malheureusement, l'estimateur TLS requière une très grande complexité de calculs dû à la SVD, de sorte qu'il devient difficile de réaliser une implémentation de cet algorithme pour des applications nécessitant des performances en temps réel.

2.6 Configuration des simulations

La configuration des simulations est fortement inspirée de [38] et de [41] avec un réseau d'antennes « patch » de 10 par 10 et la variance N_w du canal de propagation est ajustée à -50 dB. L'antenne centrale est définie comme étant l'antenne de référence dans le contexte de la méthode du chemin direct. Les chaînes RF sont modélisées avec les paramètres dans [43] où les coefficients des transmetteurs et des récepteurs ont une phase uniformément

distribuée sur $[-\pi,\pi[$ et une magnitude également distribuée uniformément sur la plage $[1-\delta,1+\delta]$, où δ est choisi de telle sorte que :

$$\sqrt{E\left(\left(\left|t_{SB}^{(m)}\right|-1\right)^2\right)}=\sqrt{E\left(\left(\left|r_{SB}^{(m)}\right|-1\right)^2\right)}=0.1 \tag{2.50}$$

où $E(\cdot)$ est l'opérateur d'espérance. Le réseau du GCD possède sept couches ($n = 7$) avec les paramètres α et $\boldsymbol{\lambda}$ qui sont initialisés avec des valeurs non nulles près de zéro afin de s'assurer que l'algorithme ne diverge pas. Le ratio μ_1 / μ_2 est approximativement de 10^5 et le coefficient de momentum β est ajusté le plus près possible de 1 afin d'éviter de diverger durant l'entraînement du réseau neuronal. Le lecteur doit se référer à la section 2.4 pour une meilleure compréhension.

2.7 Résultats obtenus avec le GCD

Les figures 2, 3 et 4 aux deux pages suivantes présentent l'erreur quadratique moyenne (MSE – *Mean Square Error*), des coefficients de calibration en fonction du SNR présent entre deux antennes voisines pour différentes techniques de calibration de réciprocité. Puisque la résolution de l'équation (2.45) implique l'inversion d'une matrice complexe de dimensions $(M-1)\times(M-1)$, des méthodes itératives comme celles de GS ou de NSE peuvent être utilisées pour trouver une estimation $\hat{\mathbf{b}}$ [41]. En effet, ces algorithmes ont été utilisés avec succès dans le contexte du ZF et du MMSE que l'on retrouve dans la résolution du problème de détection du MIMO massif de la 5G. Malheureusement, comme il a également été démontré dans [41], de telles méthodes vont nécessiter un très grand nombre d'itérations pour converger vers une solution acceptable dans le contexte d'un estimateur LS appliqué à la calibration de réciprocité.

Tel qu'attendu, l'estimateur TLS présente de très bons résultats à de très bas SNR comparé à celui du LS à cause des perturbations présentes dans $\tilde{\mathbf{A}}_1$ et $\tilde{\mathbf{a}}_1$. Dans le contexte des simulations, la méthode LS est résolue à l'aide de l'algorithme du GC standard, ce qui donne, avec un nombre suffisant d'itérations, les mêmes résultats qu'une inversion directe de matrice pour résoudre l'équation (2.45). De plus, lorsque le SNR augmente, le LS et le TLS convergent vers les mêmes performances puisque les perturbations deviennent de plus en plus négligeables.

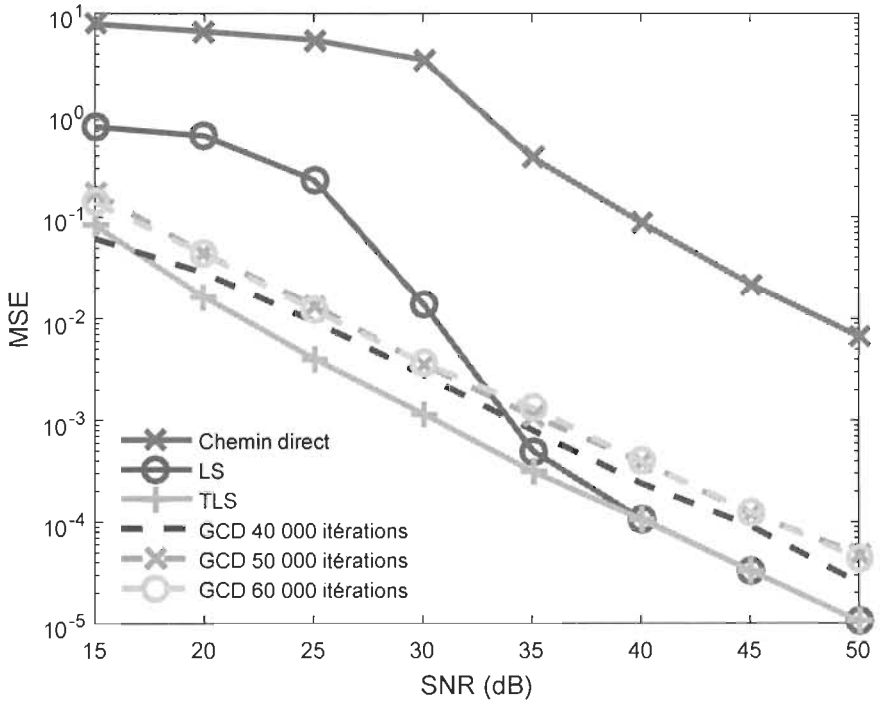


Figure 2. GCD : apprentissage à 60 dB, seuil à 10^{-14} et itérations variables

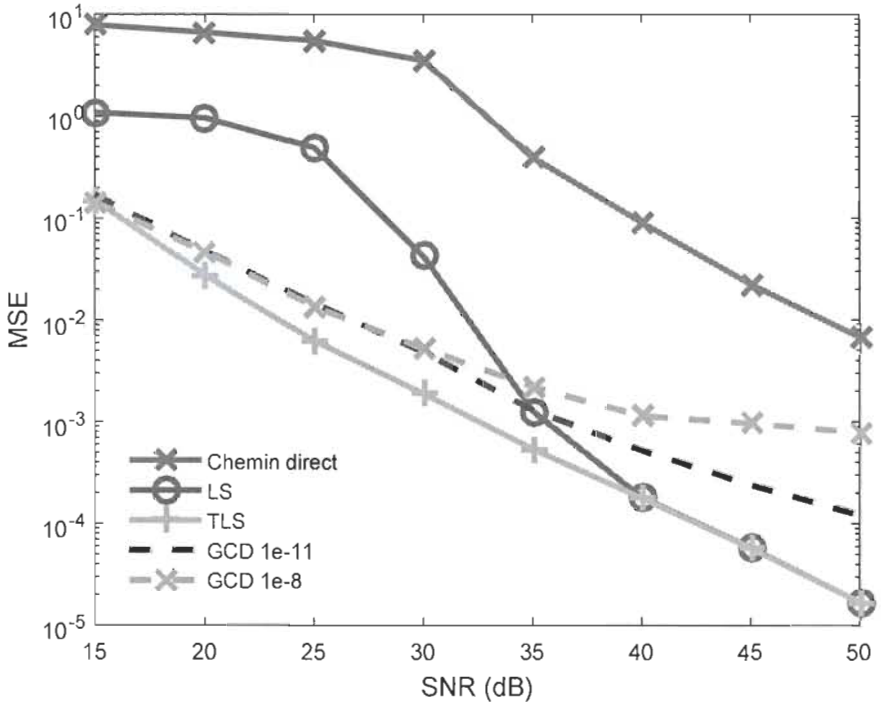


Figure 3. GCD : apprentissage à 45 dB, itérations à 40000 et seuils variables

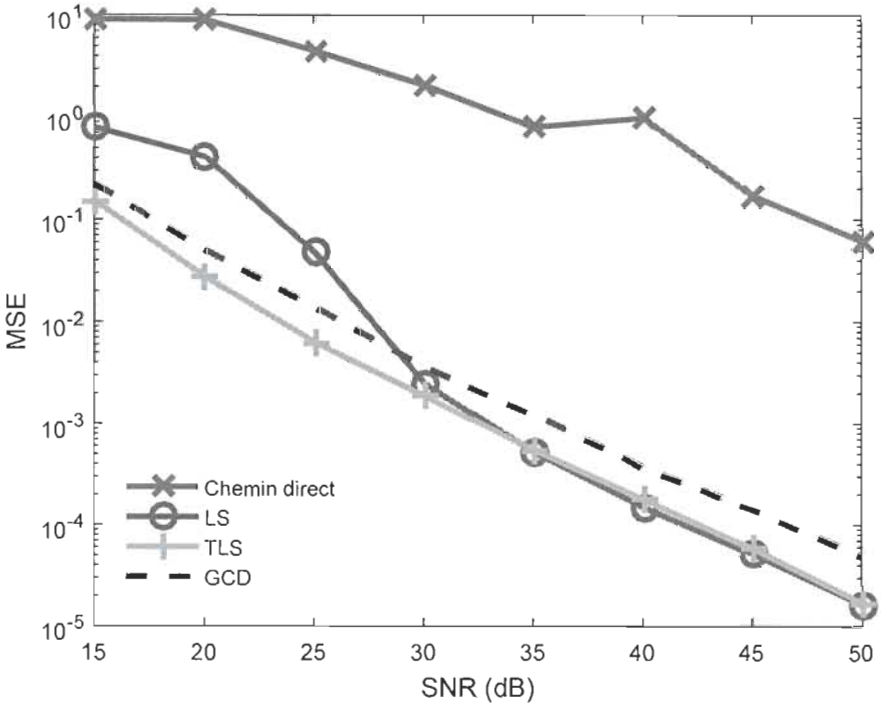


Figure 4. GCD avec 50 antennes : apprentissage à 60 dB, itérations à 40000 et seuil à 10^{-14}

D'un intérêt particulier, la figure 2 présente les résultats du GCD lorsque le processus de rétropropagation est effectué 40000, 50000 et 60000 fois par réalisation de système d'équations $\tilde{\mathbf{A}}_l\tilde{\mathbf{b}} = -\tilde{\mathbf{a}}$. Il est simple de constater que le nombre optimal d'itérations est 40000. D'un autre côté, il a été réalisé que les simulations avec moins de 35000 itérations par système d'équations ne convergeaient jamais. La figure 3 montre les performances du GCD lorsque le SNR d'entraînement entre deux antennes voisines est de 45 dB et que le nombre optimal d'itérations pour le processus de rétropropagation est choisi selon deux seuils ajustés à 10^{-8} et 10^{-11} sur la fonction de coût. L'algorithme d'apprentissage arrête la phase d'entraînement du réseau neuronal lorsque la valeur de cette dernière devient inférieure au seuil ajusté. S'il est trop haut, l'entraînement du réseau neuronal ne sera pas optimal. En revanche, s'il est trop bas, il y a toujours le risque que la fonction de coût n'atteigne jamais la valeur de ce seuil. Il est intéressant de noter que lorsque le SNR d'entraînement augmente, le seuil peut être ajusté plus bas. De façon intuitive, cela est dû au fait que les plus faibles perturbations durant l'entraînement permettent à la fonction de coûts d'atteindre de plus petites valeurs. Cela dit, lorsque ce paramètre est bien choisi, il a été observé que le MSE des coefficients de calibration est excellent pour les valeurs de SNR se situant sous la valeur du SNR d'entraînement. Cela ne cause aucun problème, car avec la méthode générale du LS appliquée dans le cas d'une centaine d'antennes à la station de base, les erreurs de transmissions en liaison descendante sont négligeables avec un SNR de calibration supérieur à 20 dB pour la technique de précodage MRT et supérieur à 35 dB pour la technique de précodage du ZF [37]-[38]. Ainsi, tant que le processus d'apprentissage se fait dans une région du SNR où les performances du GCD sont meilleures que celles du LS à 20 ou 35 dB, il ne devrait pas y avoir de problème. Sinon, à titre de comparaison, la figure 4 montre les

performances du GCD lorsqu'il y a 50 antennes à la station de base. Celles-ci semblent indépendantes du nombre d'antennes tandis que celles du LS donnent un meilleur MSE à bas SNR lorsque ce nombre diminue. Il vaut également la peine de mentionner que le SNR d'entraînement du GCD peut varier de ±5 dB d'un système d'équations à l'autre sans affecter les performances de l'algorithme.

Il est intéressant de noter que la méthode du GCD proposée ne nécessite que sept couches pour obtenir de meilleurs résultats que la méthode classique du LS à bas SNR. Cependant, la quantité de calculs du GCD par couche/itération devient plus importante que celle du GC parce que la multiplication d'un scalaire avec un vecteur dans l'équation (2.12) est remplacée par la multiplication d'une matrice avec un vecteur. Les tableaux 1, 2, 3, 4, 5 et 6 aux pages suivantes comparent la complexité du GC et du GCD en termes d'additions et de multiplications réelles selon le nombre d'antennes M à la station de base. Il est important de mentionner que la complexité du GC a été calculée avec l'hypothèse que les équations (2.14), (2.15) et (2.16) ne sont pas utilisées. En d'autres termes, l'algorithme du GC résout directement $\tilde{\mathbf{A}}_1^H \tilde{\mathbf{A}}_1 \tilde{\mathbf{b}} = \tilde{\mathbf{A}}_1^H (-\tilde{\mathbf{a}})$. Cela explique pourquoi les quantités de calculs pour rendre le système symétrique défini positif ne sont pas la même pour les deux algorithmes. La raison pour laquelle la convention des équations (2.14), (2.15) et (2.16) a été utilisée est qu'il était plus simple de dériver les équations du gradient de la fonction de coût ainsi. Cela dit, le GCD aurait pu être développé en utilisant les dérivées de Wirtinger pour obtenir la même complexité que le GC au niveau du prétraitement du système d'équations.

L'algorithme original du GC appliqué au LS requiert approximativement 250 itérations pour converger vers une solution acceptable comparativement à sept couches pour le GCD. Ces améliorations sont principalement dues au fait que $\boldsymbol{\lambda}$ a été transformé en matrice. En

effet, selon des résultats de simulations préliminaires du GCD avec λ agissant comme un scalaire, des résultats pratiquement identiques à ceux du GC standard ont été obtenus.

Tableau 1. Complexité pour rendre la matrice du système symétrique définie positive

Algorithme	Additions réelles	Multiplications réelles
GC ($\tilde{\mathbf{A}}_1^H \tilde{\mathbf{A}}_1$ et $\tilde{\mathbf{A}}_1^H \tilde{\mathbf{a}}_1$)	$2M^3 + M^2 - 5M + 2$	$2M^3 + 2M^2 - 4M$
GCD (Eq. (2.17) et (2.18))	$4M^3 - 4M^2 - M + 1$	$4M^3 - 2M^2 - 2M$

Tableau 2. Complexité pour l'initialisation du GC

Eq.	Additions réelles	Multiplications réelles
(2.2)	$2(M-1)^2 + 2(M-1)(M-2) + 2(M-1)$	$4(M-1)^2$
(2.3)	$2(M-1)^2 + 2(M-1)(M-2)$	$4(M-1)^2$
(2.5)	$(M-1) + (M-2)$	$2(M-1)$
Total	$8M^2 - 16M + 8$	$8M^2 - 14M + 6$

Tableau 3. Complexité d'une itération du GC

Eq.	Additions réelles	Multiplications réelles	Divisions réelles
(2.6)	$2(M-1)^2 + 2(M-1)(M-2)$	$4(M-1)^2$	0
(2.7)	$(M-1) + (M-2)$	$2(M-1)$	1
(2.8)	$2(M-1)$	$2(M-1)$	0
(2.9)	$2(M-1)$	$2(M-1)$	0
(2.10)	$2(M-1)^2 + 2(M-1)(M-2)$	$4(M-1)^2$	0
(2.11)	$(M-1) + (M-2)$	$2(M-1)$	0
(2.12)	$2(M-1)$	$2(M-1)$	1
Total	$8M^2 - 10M$	$8M^2 - 6M - 2$	2

Tableau 4. Complexité pour l'initialisation du GCD

Eq.	Additions réelles	Multiplications réelles
(2.24)	$(2M-2)(2M-3)+(2M-2)$	$2(M-2)^2$
(2.25)	$(2M-2)(2M-3)$	$2(M-2)^2$
Total	$8M^2-18M+10$	$8M^2-16M+8$

Tableau 5. Complexité d'une couche du GCD

Eq.	Additions réelles	Multiplications réelles
(2.20)	$(2M-2)(2M-3)+(2M-2)$	$2(M-2)^2+2(M-2)$
(2.21)	$(2M-2)(2M-3)$	$2(M-2)^2$
(2.22)	$(2M-2)(2M-3)+(2M-2)$	$2(M-2)^2$
(2.23)	$(2M-2)$	$(2M-2)$
Total	$12M^2-24M+12$	$12M^2-20M+8$

Tableau 6. Complexité totale

Algorithme	Additions réelles	Multiplications réelles
GC (250 itérations)	$2M^3+2021M^2-2543M+19$	$2M^3+2006M^2-1516M-492$
GCD (sept couches)	$4M^3+80M^2-169M+85$	$4M^3+82M^2-142M+56$

Pour 100 antennes, le GCD requiert environ 4,6 fois moins d'additions réelles et de multiplications réelles que le GC ordinaire. Bien entendu, cela exclut la phase d'apprentissage du GCD. Dépendamment du seuil ajusté sur la fonction de perte, l'entraînement du GCD pour une valeur fixe des coefficients de calibration nécessite de cinq à quinze réalisations du système d'équations $\tilde{\mathbf{A}}_l\tilde{\mathbf{b}}=-\tilde{\mathbf{a}}$ avec des perturbations différentes chaque fois. Cela représente un très gros avantage pour les systèmes de calculs hétérogènes, car une interconnexion à faible bande passante peut être utilisée pour transmettre des

données. Par exemple, le processus d'apprentissage du GCD pourrait être fait sur un GPU tandis que son application pourrait se faire sur FPGA.

Bien que les simulations se soient arrêtées à une seule valeur particulière des coefficients de calibration pour l'apprentissage, il reste tout à fait possible de faire un réentraînement partiel d'une ou plusieurs couches du GCD lorsqu'une calibration doit être faite à nouveau. Éventuellement, le réseau neuronal serait capable de reconnaitre les effets des faibles variations des coefficients de calibration dues à différents facteurs externes (température, humidité, dérive d'horloge, etc.) sans avoir besoin d'un réentraînement partiel. Cela implique donc un sacrifice dans la quantité de calculs de la phase d'apprentissage du GCD, mais une fois qu'elle est terminée, la complexité de l'algorithme se traduit littéralement par ce qui est décrit au tableau 6.

La beauté dans tout cela est le fait que l'algorithme du GCD a la capacité de décider s'il doit faire un réentraînement partiel ou non en se basant sur la valeur de la fonction de coût, et ce, même s'il n'a pas accès aux valeurs explicites des coefficients de calibration. Autrement dit, aucune donnée d'apprentissage n'est nécessaire pour que le réseau neuronal puisse adapter de façon optimale α et $\boldsymbol{\lambda}$. Il s'agit d'un énorme avantage par rapport à d'autres algorithmes déroulés tels que le « learned iterative shrinkage and thresholding algorithm» (LISTA) [11] et le « Detection Network » (DetNet) [15] qui nécessitent des données explicites d'apprentissage pour adapter leurs coefficients sur chaque couche.

À titre d'information, il aurait également pu être possible d'ajouter des fonctions d'activations non linéaires dans l'architecture du GCD afin de capturer les non-linéarités souvent présentes en pratique et d'obtenir de meilleurs résultats. Bien que l'analyse du GCD se soit arrêtée à des valeurs fixes de coefficients de calibration, l'étude du comportement et

de la robustesse de ce réseau neuronal face aux changements graduels de ces valeurs avec le temps est réservée pour des travaux futurs.

2.8 Récapitulatif du chapitre 2

Pour résumer, le déroulement profond consiste à considérer chaque itération d'un algorithme itératif comme étant une couche d'un réseau neuronal. Les paramètres de chaque itération qui sont normalement déterminés de façon explicite sont plutôt ajustés à l'aide d'un processus d'apprentissage dérivé de la rétropropagation. L'algorithme itératif du GC a été choisi pour être déroulé principalement en raison d'un paramètre d'erreur résiduelle présent à chaque itération qui permet de faire l'entraînement du réseau neuronal sans avoir accès à des données explicites d'apprentissage. De plus, un paramètre scalaire a été remplacé par un paramètre matriciel pour augmenter les degrés de libertés dans le processus d'entrainement afin d'obtenir de meilleurs résultats. Un seuil limite dérivé du paramètre d'erreur résiduelle peut être ajusté sur la fonction de coût du GCD afin d'indiquer à l'algorithme d'apprentissage quand arrêter l'ajustement du réseau neuronal.

Dans plusieurs problèmes du monde réel, il est très difficile d'avoir accès à des données explicites d'apprentissage, d'où l'utilité du GCD. Par exemple, dans le contexte du MIMO massif, la calibration de réciprocité requiert la résolution d'un grand système d'équations linéaires dérivé d'un estimateur LS sans avoir accès aux valeurs réelles des coefficients de calibration pour l'ajustement du réseau neuronal. Il a été démontré que le GCD peut résoudre ce genre de problème avec beaucoup moins de couches que d'itérations requises avec l'algorithme original du GC. Finalement, à bas SNR, les performances du GCD se rapprochent de celles de la méthode du TLS qui est considérée comme étant optimale.

Chapitre 3 - Détection MIMO massif et stratégies sur FPGA

3.1 Mise en contexte du problème de détection

Le problème de détection dans le contexte du MIMO massif du réseau 5G se résume à estimer la valeur des symboles envoyés simultanément par les usagers vers la station de base. Tel que mentionné au chapitre 2, le fonctionnement de ce type de système se fait généralement en mode TDD, ce qui signifie que les émetteurs-récepteurs des usagers et de la station de base utilisent les mêmes bandes de fréquences. Ainsi, lorsqu'un groupe envoie des données, l'autre est en mode écoute et vice-versa. Puisque la station de base comporte beaucoup plus d'antennes que le nombre d'usagers servis en même temps, une certaine diversité est créée au niveau des signaux reçus par chaque antenne, ce qui facilite le décodage efficace des symboles envoyés par chaque utilisateur. Cette diversité est due en grande partie au fait que le CSI des canaux de communication est différent entre chaque usager et chaque antenne à la station de base. Il est alors dit que les canaux de communications tendent à devenir orthogonaux.

Ainsi, comme il sera montré plus loin, pour une fenêtre de temps et une bande de fréquence (sous-porteuse) données, il est possible de créer un système d'équations linéaires surdéterminé où le nombre d'équations correspond au nombre d'antennes à la station de base et où le nombre d'inconnus est égal au nombre d'usagers. Il est ensuite possible de résoudre ce système afin de retrouver les symboles envoyés par les usagers durant cette fenêtre de

temps. Dans le cadre du réseau 5G où les latences de calculs sont critiques, l'algorithme de résolution choisi doit avoir une faible complexité de calculs et d'implémentation et une précision acceptable.

Deux approches principales sont employées dans la littérature pour résoudre les systèmes d'équations surdéterminés inhérents à la tâche de détection du MIMO massif de la 5G. Elles utilisent généralement toutes deux une méthode de prétraitement qui consiste à calculer la matrice de Gram et le vecteur de filtre adapté. Ceci est équivalent à projeter le vecteur contenant les signaux reçus par chaque antenne à la station de base sur l'espace vectoriel des colonnes de la matrice de canal. Comme il sera montré dans la prochaine section, chaque colonne de la matrice de canal correspond aux coefficients complexes du CSI en liaison montante entre un usager et chaque antenne à la station de base. Ce prétraitement permet de rendre le système d'équations carré et d'obtenir des résultats robustes face aux perturbations externes puisqu'il est dérivé d'un estimateur LS.

Une fois cette première phase de calculs terminée, la première approche consiste à utiliser un algorithme d'inversion de matrice implicite afin de résoudre de façon itérative le système d'équations pour ensuite retrouver les symboles envoyés par les usagers. Un inconvénient qui peut parfois arriver avec cette approche est que lorsque les utilisateurs envoient de nouveaux symboles, l'algorithme itératif doit être réemployé partiellement ou totalement, et ce, même si le CSI des canaux n'a pas changé. Il s'agit d'un contraste par rapport à l'autre approche qui ne fait qu'inverser explicitement la matrice de Gram pour ensuite multiplier le résultat obtenu avec le vecteur de filtre adapté. Bien que généralement, l'inversion directe d'une matrice peut demander une grande complexité de calculs, une fois que le résultat est

en main, seule l'étape de multiplication est nécessaire tant et aussi longtemps que le CSI des canaux ne change pas significativement (temps de cohérence des canaux de communication).

Dans la section 3.3, des simulations permettront de comparer les performances en termes du taux d'erreur des symboles, « symbol error rate » (SER) en anglais, entre quatre algorithmes de détections populaires dans la littérature, à savoir les méthodes implicites de résolution GS et OCD, la méthode d'inversion approximative NSE d'ordre trois et la méthode explicite d'inversion RGMIU. L'objectif est de présenter les algorithmes les plus utilisés dans le domaine et de montrer comment le RGMIU se comporte par rapport à ces autres méthodes.

3.2 Modèle des signaux

En considérant un système MIMO massif avec K usagers et M antennes à la station de base, les signaux reçus à cette dernière peuvent être modélisés par :

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{3.1}$$

où $\mathbf{y} \in \mathbb{C}^{M\times 1}$ est le vecteur contenant les signaux reçus par chaque antenne, $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_K \end{bmatrix}$ est la matrice de canal pour une sous-porteuse (bande de fréquence) donnée avec chaque colonne qui représente le CSI du canal en liaison montante entre chacune des antennes et un usager en particulier, ce qui signifie que $\mathbf{H} \in \mathbb{C}^{M\times K}$, $\mathbf{s} = \begin{bmatrix} s_1 & s_2 & \cdots & s_K \end{bmatrix}^T$ est le vecteur contenant les K symboles envoyés par les K usagers et $\mathbf{n} \in \mathbb{C}^{M\times 1}$ est un bruit Gaussien complexe circulairement symétrique de moyenne nulle et de variance σ^2. La matrice de Gram et le vecteur de filtre adapté se calculent comme suit :

$$\mathbf{G} = \mathbf{H}^H\mathbf{H} \tag{3.2}$$

$$\mathbf{y}_{MF} = \mathbf{H}^H \mathbf{y} \tag{3.3}$$

Où $\mathbf{G}$ est la matrice de Gram et $\mathbf{y}_{MF}$ est le vecteur de filtre adapté, « Matched Filter Vector » en anglais. Il est ensuite possible d'écrire le nouveau système d'équations à résoudre :

$$\mathbf{y}_{MF} = \mathbf{G}\mathbf{s} + \mathbf{H}^H \mathbf{n} \tag{3.4}$$

L'estimation des symboles envoyés par les usagers peut se faire à l'aide d'une inversion explicite de la matrice de Gram :

$$\hat{\mathbf{s}} = (\mathbf{G})^{-1} \mathbf{y}_{MF} \tag{3.5}$$

où $\hat{\mathbf{s}}$ est le vecteur contenant l'estimation des symboles envoyés par les usagers. De plus, tel que mentionné dans la section précédente, il est également possible de résoudre l'équation (3.4) à l'aide d'une méthode implicite itérative.

Cela dit, le processus consistant à résoudre l'équation (3.4) est celui du ZF. En additionnant la variance σ^2 à chaque élément de la diagonale de la matrice de Gram, et en résolvant le même système d'équations, c'est la méthode du MMSE qui aurait été employée. En approximant la matrice de Gram seulement par sa diagonale, c'est la méthode du MRT qui aurait été utilisée. Néanmoins, puisqu'il est difficile d'obtenir en pratique la valeur de σ^2 et que la méthode du MRT est moins robuste que les deux autres, c'est le processus du ZF qui est retenu pour les simulations.

Par la suite, les valeurs contenues dans le vecteur $\hat{\mathbf{s}}$ doivent être modifiées afin qu'elles puissent faire partie du même ensemble (constellation des symboles) que celui de $_{\mathbf{s}}$. Cette procédure est connue sous le nom de décision de sortie et elle peut être qualifiée de « soft » lorsqu'un processus de correction des erreurs par anticipation est utilisé et de « hard » dans

le cas contraire. Dans un souci de simplicité, l'analyse des simulations du SER pour chaque algorithme se fera à l'aide de la détection « hard » qui consiste à prendre une décision de sortie basée sur la plus petite distance euclidienne entre les valeurs de $\hat{\mathbf{s}}$ et les symboles faisant partie de la constellation discrète de $\mathbf{s}$. De façon mathématique, cela se traduit par :

$$\hat{s}_i^{hard} = \arg\min_{z \in \mathrm{O}} \left| \hat{s}_i - z \right| \quad i = 1, \ldots, K \tag{3.6}$$

où O correspond à l'ensemble des symboles de la constellation de $\mathbf{s}$ et i représente un usager en particulier. Avec cette métrique en main, il est possible de déterminer le SER en calculant le ratio des mauvaises décisions de sortie sur le nombre total de symboles envoyés par les usagers.

Puisque c'est l'algorithme RGMIU [29] qui est implémenté sur FPGA dans le cadre de ce travail, les étapes de calculs pour la $m^{ième}$ itération de cette méthode sont :

$$\mathbf{z} = \mathbf{G}_{m+1,m+1} \tag{3.7}$$

$$\mathbf{y}_1 = \mathbf{G}_{1:m,m+1} \tag{3.8}$$

$$\mathbf{y}_2 = \mathbf{B}_m \mathbf{y}_1 \tag{3.9}$$

$$c = 1 / \left(\mathbf{z} - \mathbf{y}_1^H \mathbf{y}_2 \right) \tag{3.10}$$

$$\mathbf{y}_3 = c\, \mathbf{y}_2 \tag{3.11}$$

$$\mathbf{\Gamma} = \mathbf{B}_m + c\, \mathbf{y}_2 \mathbf{y}_2^H \tag{3.12}$$

$$\mathbf{B}_{m+1} = \begin{bmatrix} \mathbf{\Gamma} & -\mathbf{y}_3 \\ -\mathbf{y}_3^H & c \end{bmatrix} \tag{3.13}$$

où $m \in [1:K-1]$, $\mathbf{G}_{m+1,m+1}$ représente l'élément de la matrice de Gram à la ligne et à la colonne $m+1$, $\mathbf{G}_{1:m,m+1}$ est le vecteur qui correspond aux lignes 1 à m et à la colonne $m+1$ de la matrice de Gram. Les conditions initiales de cet algorithme sont définies comme suit :

$$\mathbf{G} = \mathbf{H}^H \mathbf{H} \tag{3.14}$$

$$\mathbf{B}_1 = \left(\mathbf{G}_{1,1}\right)^{-1} \tag{3.15}$$

Une analyse rapide permet de constater que les dimensions de $\mathbf{B}$ augmentent à chaque itération jusqu'à obtenir $\mathbf{B}_K$ qui correspond à l'inverse de la matrice de Gram.

3.3 Résultats de simulations

Les paramètres de simulations ont été inspirés de ceux retrouvés dans [25] avec 128 antennes à la station de base et 8 ou 25 usagers. La matrice de canaux est modélisée avec une variable aléatoire Gaussienne complexe circulairement symétrique de moyenne nulle et de variance unitaire. Une constellation 64 QAM a été utilisée pour les symboles qui ont une puissance moyenne émise unitaire. Les performances des algorithmes GS, NSE, RGMIU et OCD sont simulées.

Avant même de regarder les résultats de simulations, il est possible de prédire certains comportements de ces algorithmes. Premièrement, puisque les méthodes GS et RGMIU résolvent toutes les deux le problème du ZF, il est réaliste de supposer qu'avec suffisamment d'itérations, l'algorithme GS atteindra les mêmes performances que celui du RGMIU. De plus, tel que mentionné dans [25], l'algorithme OCD a le potentiel d'obtenir des performances proches du MMSE sans avoir explicitement accès à la valeur de σ^2.

Les figures 5 et 6 aux pages suivantes montrent le SER obtenu pour les quatre algorithmes avec 8 et 25 usagers. Les méthodes GS et OCD sont présentées pour deux itérations à la figure 5 et pour trois itérations à la figure 6. Comme prévu, avec 8 usagers, il faut au moins deux itérations pour que l'algorithme GS atteigne les mêmes performances que l'algorithme RGMIU. Cela dit, pour ce même nombre d'usagers, il faut trois itérations pour que la méthode OCD obtienne les mêmes résultats que les algorithmes RGMIU et GS. La raison pour laquelle la méthode OCD ne surpasse pas les autres est que les CSI des canaux dans la matrice de canaux ne sont pas corrélés. De cette façon, la diversité à la station de base est maximisée et le système d'équations est « facile » à résoudre avec une bonne précision. En pratique, puisque les antennes à la station de base sont relativement proches les unes des autres, un certain degré de corrélation peut être observé entre certains canaux. Tel que démontré dans [25], la méthode OCD est moins sensible que les autres algorithmes dans ce genre de situation. Néanmoins, l'objectif est simplement de comparer simplement les performances relatives de ces méthodes les unes par rapport aux autres. D'autre part, l'algorithme NSE d'ordre 3 n'atteint jamais les performances d'une inversion matricielle exacte et a un SER catastrophique avec 25 usagers.

Il est particulièrement intéressant de noter que pour 25 usagers, les méthodes GS et OCD ont besoin de plus d'itérations pour atteindre les performances du RGMIU. Dans la figure 6, ces deux algorithmes n'atteignent pas le SER du RGMIU avec trois itérations. En général, il a été observé que le nombre d'itérations optimal pour les méthodes implicites de résolution dépendra du rapport entre le nombre d'antennes à la station de base et le nombre d'usagers, ce qui représente un inconvénient, car la complexité de l'algorithme devient difficile à prévoir. En revanche, celle de la méthode RGMIU peut être connue à l'avance, car il faut

autant d'itérations que le nombre d'usagers moins un pour inverser la matrice de Gram. En outre, les résultats de cet algorithme seront toujours précis, car sa sortie est l'inverse exact de la matrice de Gram. De plus, la méthode NSE d'ordre trois n'est précise que lorsque le rapport du nombre d'antennes à la station de base sur le nombre d'usagers est élevé. Au-delà de l'ordre trois, la complexité de l'algorithme NSE devient trop importante pour rivaliser avec les autres méthodes.

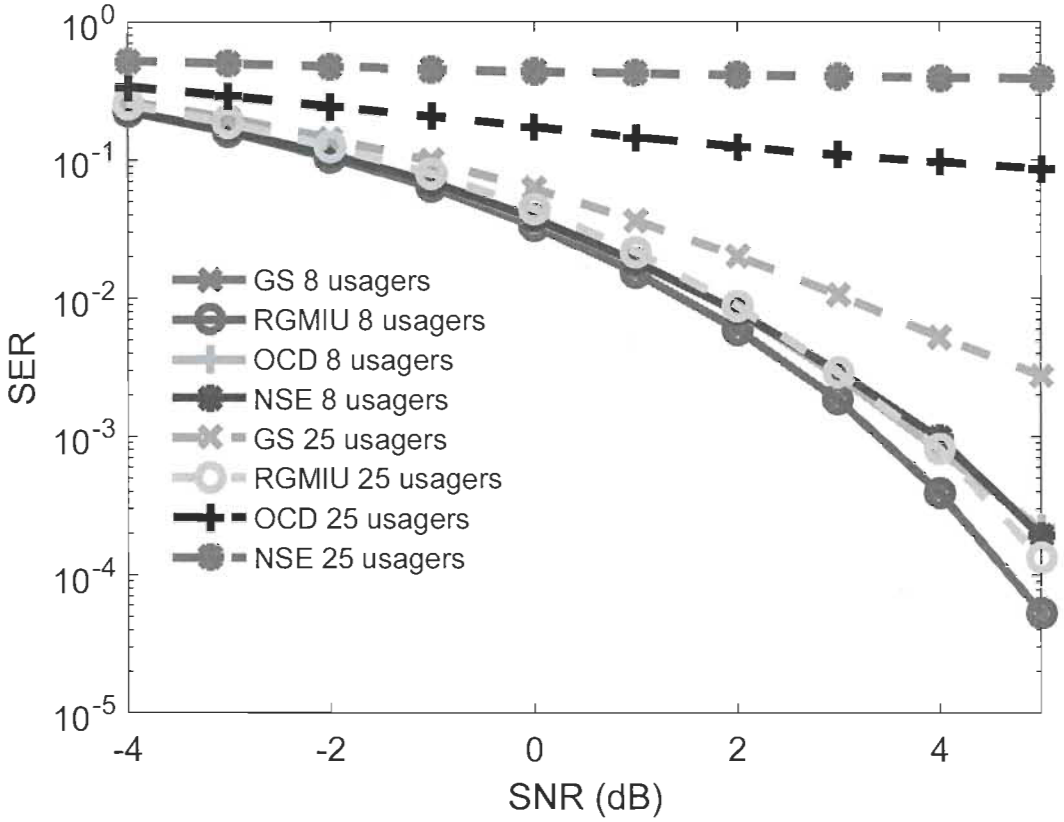


Figure 5. Détection : SER obtenus (GS et OCD avec deux itérations)

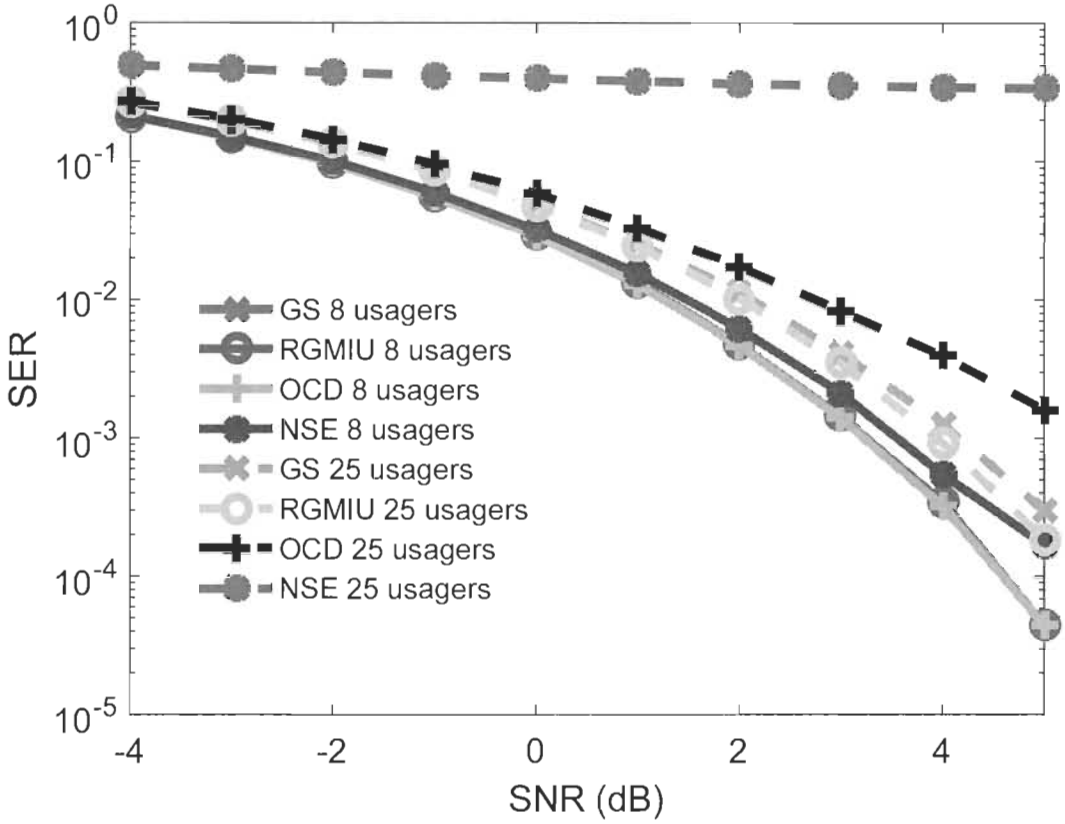


Figure 6. Détection : SER obtenus (GS et OCD avec trois itérations)

3.4 Implémentation

Vivado HLS 2019.1 a été utilisé pour mettre en œuvre l'implémentation de l'algorithme RGMIU [46]. Cet outil permet de générer une description VHDL (*Very High Speed Integrated Circuit Hardware Description Langage*) à partir d'un code en langage C ou C++. Toutes les solutions ont été validées avec une cosimulation RTL (*Register Transfer Level*) et le FPGA xc7vx690t de la famille Virtex 7 de Xilinx a été choisi en raison du grand nombre de ressources matérielles disponibles pour l'implémentation.

3.4.1 Stratégies d'implémentation FPGA

Avant de de décrire directement les stratégies de conception, il est important de mentionner que seul le cœur de l'algorithme RGMIU, c'est-à-dire, les équations (3.7) à (3.13), a été implémenté. En d'autres termes, la phase de prétraitement de la matrice de Gram et du vecteur de filtre adapté n'est pas mise en œuvre ainsi que la phase de post-traitement qui consiste à multiplier l'inverse de la matrice de Gram avec le vecteur de filtre adapté.

Le but est seulement d'optimiser l'algorithme d'inversion parce qu'il a la capacité d'atteindre un débit de détection de données énorme, mais les unités de prétraitement et de post-traitement ne peuvent pas suivre la vitesse de calculs à laquelle le noyau de la méthode RGMIU peut fonctionner sans une forte augmentation de l'utilisation des ressources sur le FPGA. L'objectif est seulement de montrer le potentiel du cœur de l'algorithme. D'autre part, la mise en œuvre complète de la méthode RGMIU sur un circuit intégré ASIC, qui peut atteindre des fréquences d'horloge plus élevées que le FPGA, pourrait être réaliste dans le sens où les phases de prétraitement et de post-traitement fonctionneraient à une fréquence d'horloge plus élevée que le cœur de l'algorithme afin de l'utiliser à sa pleine capacité.

Trois stratégies d'implémentation ont été utilisées [47]. La première, dénommée S1, utilise des ressources matérielles différentes pour chaque itération. Il est important de noter que l'algorithme RGMIU implique des vecteurs et des matrices de tailles croissantes à chaque mise à jour, de sorte que les limites des boucles dans le code sont variables en fonction de l'itération en cours. Ainsi, il a été déduit que la meilleure façon de traduire cela dans Vivado HLS (HLS – *High Level Synthesis*) est de créer des fonctions modèles en C++ avec le paramètre modèle (« *template parameter* » en anglais) correspondant aux limites des boucles de sorte que chaque fois qu'une fonction modèle est appelée avec un nouveau

paramètre modèle, Vivado HLS comprend qu'il doit utiliser de nouvelles ressources matérielles pour cet appel de fonction. Chaque fonction a été écrite de telle sorte qu'elle puisse être pipelinée avec un intervalle d'initiation unitaire. Sans les fonctions modèles, il aurait fallu déclarer autant de fonctions qu'il y a d'itérations multipliées par le nombre d'étapes dans une itération, la seule différence étant les valeurs des paramètres indiquant le nombre de fois que les boucles doivent répéter le code qu'elles exécutent. Vivado HLS permet également l'utilisation de directives préprocesseur de type #pragma qui indiquent au logiciel la façon d'utiliser les ressources matérielles disponibles sur le FPGA. En ce sens, il aurait pu être possible d'utiliser la consigne #pragma FUNCTION_INSTANTIATE à la place d'utiliser les fonctions modèles, mais pour une raison inconnue, cette directive interférait avec la consigne #pragma PIPELINE qui elle, permet de pipeliner l'architecture de l'implémentation. En résumé, une fonction générale qui appelle toutes les fonctions modèle les unes après les autres a été créée [47]. Cette fonction générale a été pipelinée, et les fonctions modèles ont été alignées à l'aide de la directive #pragma INLINE afin d'augmenter le flux de données et pour maximiser les performances. Au total, il y a cinq fonctions modèles correspondant aux équations (3.9) à (3.13) et la valeur du paramètre modèle passe de 1 à $K-1$ dépendamment de l'itération de l'algorithme RGMIU. En outre, comme les limites des boucles sont différentes pour chaque itération, les ressources matérielles utilisées le sont également, car la plupart des boucles sont automatiquement déroulées par la directive de pipelinage. La figure 7 à la page suivante montre l'architecture de la première stratégie d'implémentation.

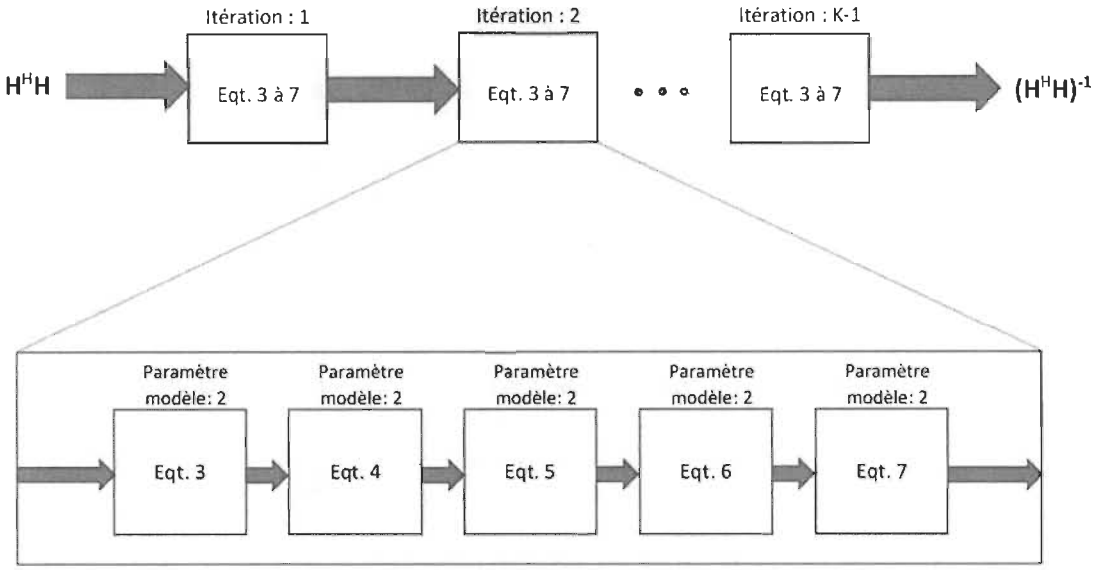


Figure 7. Architecture pour S1, S2 et S3

La deuxième stratégie (S2) est fondamentalement la même que la première, mis à part que la conception est fortement marquée par un plus grand nombre de registres intermédiaires, de sorte que le signal d'horloge peut atteindre des fréquences plus élevées, ce qui permet un meilleur débit de calcul des données. Les inconvénients de cette stratégie sont l'augmentation de la consommation des ressources matérielles et une latence de traitement plus élevée. Pour traduire cela dans Vivado HLS, la première solution a été réutilisée et la contrainte d'horloge a été fixée avec une période plus courte. Ainsi, l'architecture générale de S2 est la même que celle de S1 illustrée à la figure 7.

La troisième stratégie (S3) est conçue de telle manière que la consommation des ressources matérielles est similaire en termes de pourcentage pour les éléments de traitement de signal (DSP48E), les bascules et les tables de correspondance. De cette façon, l'utilisation des ressources est équilibrée de sorte qu'elle laisse plus de place pour d'autres implémentations sur le même FPGA. L'inconvénient de cette méthode est que le débit de

traitement des données est généralement réduit de moitié. Pour déployer cette stratégie, la directive #pragma ALLOCATION a été utilisée pour limiter le nombre de DSP48E dans la fonction générale. Cela implique que des bascules et des tables de correspondances supplémentaires sont nécessaires pour compenser cette limitation. Une fois de plus, l'architecture générale de S3 est la même que celle de S1 illustrée à la figure 7.

Cela étant dit, une quatrième stratégie qui consiste à réutiliser les mêmes ressources matérielles pour les deux dernières itérations de l'algorithme RGMIU a été envisagée. L'objectif était de réduire considérablement la consommation des ressources puisque l'utilisation des DSP48E est une fonction cubique par rapport au nombre d'itérations. Cette stratégie permettait effectivement d'avoir une grande amélioration, mais le débit de traitement des données devenait trop lent et donc, l'idée fut abandonnée.

Une stratégie qui n'a pas été essayée, mais qui aurait pu être intéressante, consiste en un pipeline entrelacé tel que présenté dans [25]. En effet, puisque la boucle de la méthode RGMIU décrite aux équations (3.7) à (3.13) contient des dépendances inter itérations, cela empêche l'implémentation d'un pipeline traditionnel à moins que l'algorithme soit complètement déroulé comme dans S1, S2 et S3. Pour surmonter ce problème sans un déroulement complet, il aurait été possible de réutiliser toujours les mêmes ressources matérielles qui représentent une itération et de créer plusieurs étages de pipelines à l'intérieur du design. Le noyau de l'algorithme aurait ainsi pu accepter une nouvelle entrée de matrice de Gram à chaque cycle d'horloge jusqu'à ce que le pipeline soit plein. Par la suite, l'algorithme n'aurait eu qu'à terminer ses itérations avant de pouvoir accepter un nouveau lot d'entrées. En ce sens, il aurait également été possible de développer une implémentation qui calcule plus d'une itération afin d'obtenir un plus grand nombre d'étages de pipeline dans la

conception et un débit global plus élevé au détriment d'une plus grande utilisation des ressources. Cette stratégie de pipeline entrelacé est réservée pour des travaux futurs.

3.4.2 Analyse des nombres à virgule fixe

L'analyse en virgule fixe a été effectuée avec des mots de 16 bits. La plupart des variables internes ont utilisé une partie entière d'un bit signé et une partie fractionnaire de 15 bits. Heureusement, aucun décalage n'a été nécessaire pour réduire la plage dynamique des nombres et les opérateurs courants de Vivado HLS ont pu aligner automatiquement les virgules binaires entre deux variables ayant un nombre de bits fractionnaires différent. Cela étant dit, comme les DSP48E des FPGA peuvent accepter une largeur maximale de 18 bits, la conception n'a pas été pénalisée en termes de consommation de ressources. Pour toutes les multiplications de valeurs complexes, quatre multiplicateurs réels et deux additionneurs réels sont utilisés.

Une bonne précision a pu être atteinte de sorte que la différence entre la courbe du SER en virgule flottante et de celle en virgule fixe pour 8 usagers ne dépasse jamais 0,5 dB. La figure 8 à la page suivante présente les performances du RGMIU en termes de SER avec des nombres à virgule fixe de 15 et 16 bits. Il est clair que 16 bits est la grandeur minimale requise pour obtenir des résultats qui se situent toujours à moins de 0,5 dB du SER obtenu avec des nombres à virgule flottante de 32 bits.

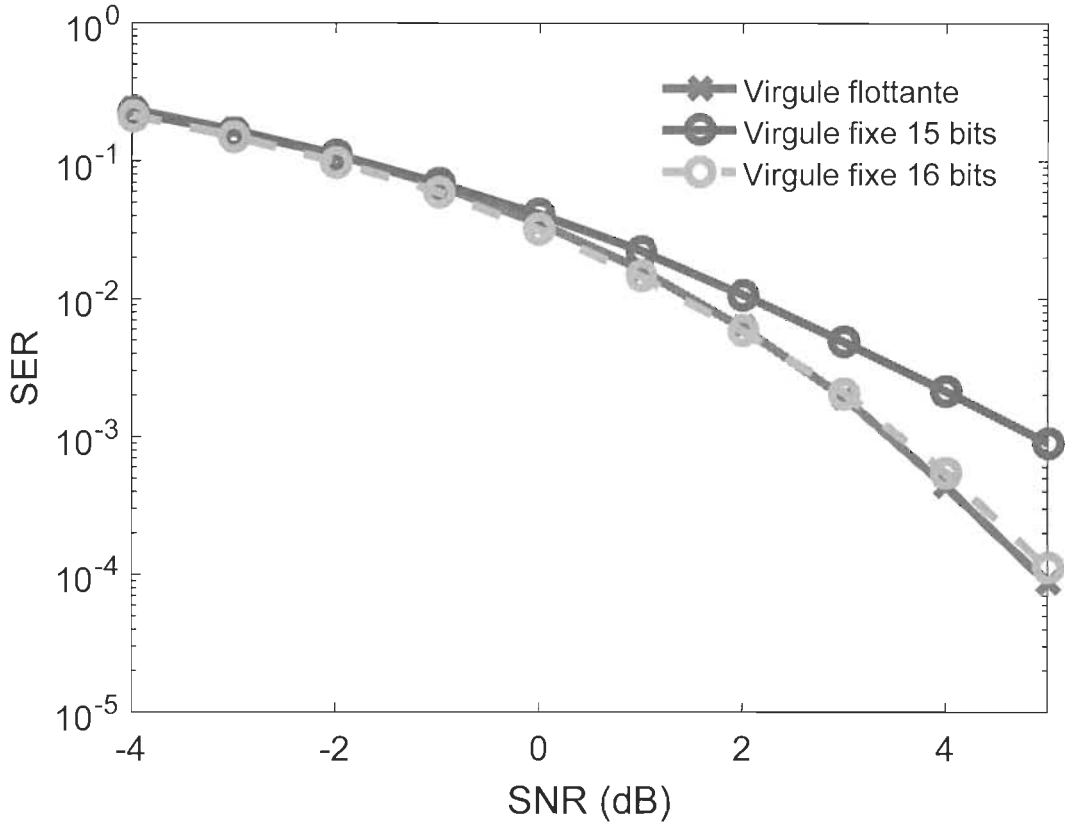


Figure 8. RGMIU : analyse des nombres à virgule fixe

3.4.3 Estimation de l'utilisation des ressources

L'utilisation des ressources pour toutes les stratégies en termes de DSP48E, de bascules et de tables de correspondance est décrite à la page suivante au tableau 7 pour différents nombres d'usagers. En analysant S1 et S2, le nombre de DSP48E utilisé est indépendant de la stratégie employée. Ceci est tout à fait normal, car ils sont utilisés pour des opérations de multiplication et de multiplication-accumulation (MAC) qui sont indépendantes du niveau de pipelinage. Les DSP48E ne dépendent que du nombre d'usagers. Il est possible de dériver une formule qui relie ces deux paramètres entre eux. En analysant l'implémentation avec l'outil de perspective d'analyse de Vivado HLS, il a été déterminé empiriquement que le nombre requis de DSP48E, à savoir $d(l)$, pour la $l^{ième}$ itération est défini comme suit :

$$d(l) = 8l^2 + 4l - 2 \tag{3.16}$$

Il est par la suite possible de déterminer le nombre total de DSP48E utilisés, à savoir D, qui correspond à :

$$D = \sum_{l=1}^{L} d(\mathrm{l}) = \frac{16L^3 + 36L^2 + 8L}{6} \tag{3.17}$$

où L est le nombre total d'itérations qui correspond au nombre d'usagers moins un. L'équation (3.8) montre que le nombre de DSP48E suit un ordre cubique qui peut être un certain facteur limitant pour les implémentations ayant plus de 8 usagers. C'est pour cette raison que la stratégie S3 a été mise en œuvre afin de répartir les ressources matérielles de telle sorte qu'il y ait suffisamment de DSP48E de disponibles pour 12 usagers. Puisque S3 utilise les ressources matérielles de manière équilibrée, cela réduit considérablement le nombre de DSP48E de 13% avec 8 usagers.

Tableau 7. Résumé des performances de chacune des stratégies

Stratégie	Horloge (ns)	usagers	Latence estimée (µs)	Débit estimé (Gb/s)	DSP48Es	Tables	Bascules
S1	2.64 ± 0.38	4	0.428	9.09	130 (3%)	12192 (2%)	18065 (2%)
		8	0.966	18.18	1218 (33%)	60216 (13%)	95177 (10%)
S2	1.934 ± 0.13	4	0.404	12.41	130 (3%)	14726 (3%)	32829 (3%)
		8	1.039	24.82	1218 (33%)	84282 (19%)	210569 (24%)
S3	2.577 ± 0.38	8	0.905	9.31	750 (20%)	84098 (19%)	161753 (18%)
		12	1.495	13.97	2400 (66%)	289930 (66%)	534689 (61%)

3.4.4 Estimation des latences et des débits

En se référant au tableau 7, les latences entre S1 et S2 sont similaires, mais les débits les plus élevés sont atteints par S2. La figure 9 à la page suivante compare les latences de S1, S2 et S3 en fonction du nombre d'usagers. Pour 8 utilisateurs, une latence potentielle de 0,966 µs et un débit potentiel de 18,18 Gb/s pourraient être atteints avec S1. Ces chiffres sont conditionnels au fait que les unités de prétraitement et de post-traitement puissent suivre le rythme. Sinon, pour 8 usagers avec S2, il est théoriquement possible d'atteindre une latence de 1,04 µs et un débit de 24,82 Gb/s. La restriction des ressources de S3 a eu pour effet de réduire de moitié environ le débit maximal de traitement des données si on le compare à celui de S1. Par contre, la latence de S3 semble suivre la même tendance linéaire que S1 et S2. De plus, pour les trois stratégies, le débit et la latence sont tous deux pratiquement linéaires par rapport au nombre d'utilisateurs. Ceci est tout à fait normal pour le débit, car le nombre d'usagers est le seul facteur multiplicatif qui peut varier dans l'implémentation. D'autre part, en examinant la perspective d'analyse de Vivado HLS avec S1, S2 et S3, ce fait a également été confirmé pour la latence. En effet, en prenant S1 comme exemple, il y a une division de 26 cycles pour l'initialisation de $\mathbf{B}_1$ et chaque nouvelle itération comporte une nouvelle division de 26 cycles (équation (3.10)) et des opérations parallèles qui prennent une latence d'environ 20 cycles pour les premières itérations et qui croît très lentement avec les mises à jour subséquentes. Cela donne une relation approximativement linéaire entre la latence et le nombre d'usagers.

L'algorithme RGMIU devient particulièrement intéressant lorsqu'un nouvel usager est ajouté dynamiquement au système d'équations ou lorsque le CSI entre la station de base et un usager en particulier change. En effet, si un nouvel utilisateur est inclus, le point de départ

de l'algorithme est l'inverse de la matrice Gram déjà calculée. En d'autres termes, chaque fois qu'un nouvel usager est ajouté, l'algorithme n'a besoin de faire qu'une seule itération. De plus, tel que démontré dans [29], lorsque le canal change pour un utilisateur, seul un calcul partiel équivalent, en termes de latence, à deux fois la dernière itération de l'algorithme doit être effectuée. Intuitivement, cela correspond à d'abord retirer l'usager ciblé, puis à l'ajouter de nouveau en considérant la nouvelle valeur du CSI entre ce dernier et la station de base. La majorité des autres algorithmes doivent refaire tous les calculs dès que ce genre de situation se produit. Le tableau 8 à la page suivante montre les latences approximatives pour différentes configurations de mise à jour de l'inverse de la matrice Gram lorsque le CSI entre un utilisateur et la station de base change. Cette approximation est calculée en multipliant par deux le résultat obtenu en faisant la différence entre les latences totales de deux configurations avec $n-1$ et n usagers, où n est arbitraire.

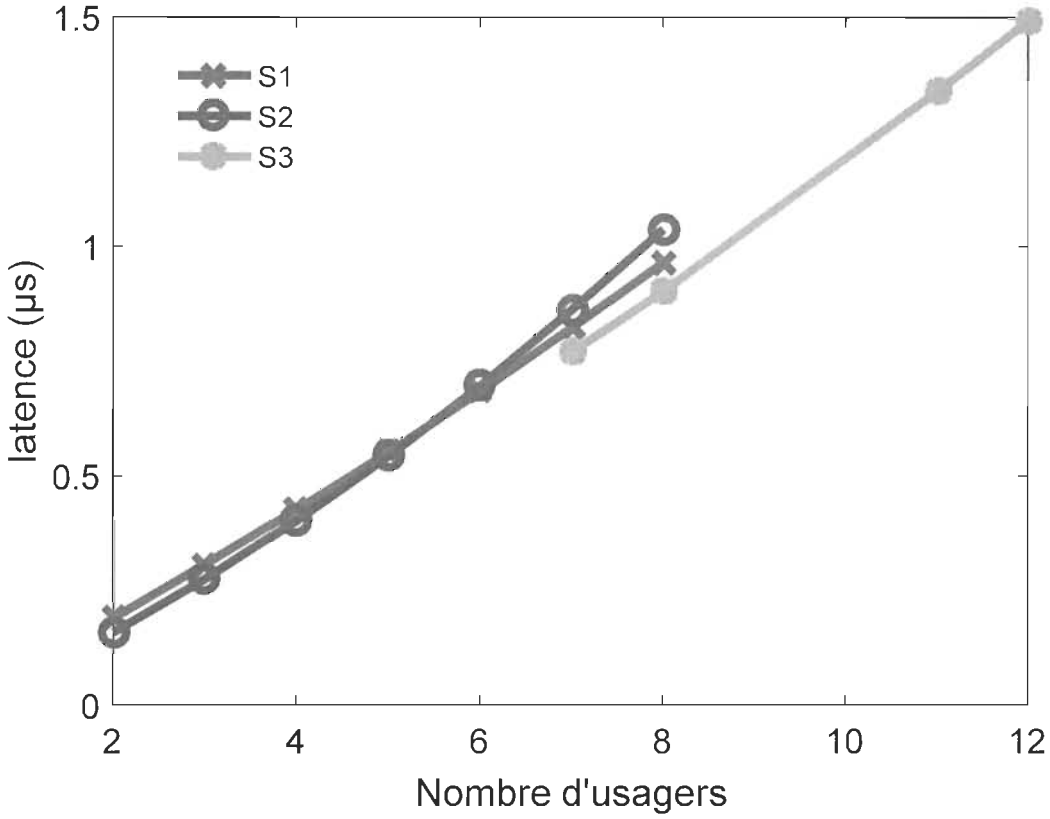


Figure 9. Latences des différentes stratégies

Tableau 8. Latences estimées pour le remplacement d'un usager

Stratégie	Nombre d'usagers	Latence estimée pour remplacer un usager (µs)
S1	4	0.243
	6	0.264
	8	0.285
S2	4	0.259
	6	0.306
	8	0.352
S3	8	0.201
	12	0.236

3.4.5 Efficacité énergétique

Il a été possible de produire une estimation de la consommation énergétique des différentes stratégies à l'aide du tableur Excel « Xilinx Power Estimator » de Xilinx. Le tableau 9 à la page suivante présente les résultats obtenus selon différents nombres d'usagers. Les métriques observées sont la consommation de puissance en Watts (W) et l'efficacité énergétique en gigabits par joule (Gb/J). La première chose à remarquer est que cette dernière diminue avec l'augmentation du nombre d'usagers, et ce, quelle que soit la stratégie utilisée. Puisque S2 est fortement pipelinée, la fréquence maximale d'horloge plus élevée augmente fortement la consommation de puissance de l'implémentation. De S1 à S2, pour 8 usagers avec une modulation 64 QAM, l'efficacité énergétique passe d'environ 3,615 Gb/J à 2,189 Gb/J, de sorte que le compromis global pour S2 correspond à un débit plus élevé au détriment d'une efficacité énergétique moindre et d'une consommation de ressources matérielles plus élevée. D'autre part, comme S3 doit compenser la limite maximale de DSP48E avec beaucoup de tables de correspondance et de bascules, la consommation de puissance est plus élevée par rapport à S1. En effet, un rendement de 1,480 Gb/J pour 8 utilisateurs est obtenu avec une modulation 64 QAM. Le compromis global de S3 devient alors un sacrifice du débit et de la consommation énergétique pour une utilisation plus équilibrée des ressources. Il est important de noter ici que l'efficacité énergétique est calculée uniquement pour le cœur de l'algorithme RGMIU. En ajoutant les unités de prétraitement et de post-traitement, l'efficacité énergétique serait moins élevée que ce qui est estimé au tableau 8.

Tableau 9. Estimation de la consommation énergétique des stratégies

Stratégie	Nombre d'usagers	Estimation de la consommation de puissance (W)	Estimation de l'efficacité énergétique (Gb/J)
S1	4	1.16	7.837
	6	2.541	5.367
	8	5.03	3.615
S2	4	2.068	6.001
	6	5.306	3.508
	8	11.34	2.189
S3	8	6.291	1.480
	12	16.809	0.831

3.5 Récapitulatif du chapitre 3

Pour résumer, des simulations ont permis de comparer les performances en termes de SER entre quatre différents types d'algorithmes de détection. Les résultats ont démontré que les algorithmes itératifs d'inversion implicite sont sensibles au nombre d'itérations choisies. Cela dit, lorsqu'il est correctement ajusté, l'algorithme GS a les mêmes performances que la méthode d'inversion explicite RGMIU. D'un autre côté, l'algorithme NSE présente de très mauvais résultats lorsque le nombre d'usagers augmente par rapport au nombre d'antennes à la station de base. Le but de ces simulations est de comparer les performances relatives des méthodes actuellement les plus utilisées pour résoudre le problème de détection du MIMO massif.

De plus, l'algorithme RGMIU a été implémenté sur Vivado HLS afin de démontrer son grand potentiel pour atteindre un débit de détection de données très élevé à condition que les unités de prétraitement et de post-traitement soient capables de suivre le rythme du cœur de

cette méthode. La conception a été faite sur des mots de 16 bits et les performances de trois stratégies d'implémentation différentes en termes de consommation de ressources, de latence/débit et d'efficacité énergétique ont été comparées afin d'en faire ressortir les compromis possibles.

Chapitre 4 - Conclusion

Ce travail a permis de mettre en lumière deux aspects importants du MIMO massif de la 5G, soit le problème de calibration de réciprocité et le problème de détection. La littérature récente tend à démontrer que l'introduction de l'AP dans le domaine des télécommunications permettra d'améliorer les performances des algorithmes classiques afin de repousser les barrières qui semblaient infranchissables par le passé.

Dans cet ordre d'idées, le déroulement profond du GC a d'abord été présenté au chapitre 2 afin de démontrer le potentiel de cette approche novatrice au niveau du problème de calibration de réciprocité. Malgré le fait que les résultats étaient concluants pour des coefficients de calibrations fixes, beaucoup de travail reste encore à accomplir afin de démontrer que cette méthode serait viable dans le cadre d'une implémentation dans le monde réel. En effet, les coefficients de calibration peuvent légèrement changer avec le temps en raison de facteurs externes comme la température. Il serait donc intéressant de valider la robustesse de ce réseau neuronal face à ces changements. De plus, il faudrait développer une stratégie d'apprentissage plus poussée pour déterminer le nombre optimal de couches à sélectionner lorsqu'un réentraînement partiel serait nécessaire. Aussi, bien que cela n'ait pas été mentionné, il serait impératif de trouver une façon d'accélérer le processus d'apprentissage initial du réseau neuronal, car cela peut rapidement devenir un facteur limitant. Ainsi, des méthodes comme celle présentée dans [48] qui utilise un filtre de Kalman pour augmenter la vitesse de convergence du processus d'apprentissage ou bien comme celle

développée dans [12] qui repose sur l'envoie de perturbations dans le réseau neuronal afin de déterminer l'ajustement optimal des poids internes, devront être appliquées au GCD.

Malgré tout cela, le plein potentiel du GCD ne serait pas encore atteint dans le sens où aucune fonction d'activation non linéaire n'a été utilisée dans ce réseau de neurones. Il aurait pu être possible d'en ajouter une au vecteur $\mathbf{r}$ de l'équation (2.20) par exemple. Beaucoup de simulations devront être réalisées afin de déterminer le bon de type de fonction non linéaire qui s'appliquerait le mieux en fonction du problème ciblé.

Une fois tous ces correctifs effectués, il faudrait évaluer les performances en termes de ressources matérielles, d'efficacité énergétique, de complexité de calculs, de débits et de latences qu'une implémentation réelle de cet algorithme impliquerait. En ce sens, dans le cadre de ce travail, une évaluation de l'implémentation réelle de la méthode RGMIU a été appliquée au problème de détection du MIMO massif de la 5G. Bien que les résultats obtenus soient forts prometteurs, il reste une difficulté à résoudre de façon concrète au niveau des étapes de prétraitement et de post-traitement des données. De plus, afin de réduire les ressources matérielles utilisées lors d'une implémentation physique de l'algorithme RGMIU, la stratégie de pipeline entrelacé devra être minutieusement étudiée afin de faire ressortir les meilleurs compromis possibles.

Avec le réseau 5G qui est déjà en processus de déploiement et le réseau 6G qui fera probablement son apparition dans plusieurs années, les performances des algorithmes de télécommunications actuellement utilisés devront être repoussées à leurs limites afin de satisfaire les critères de plus en plus exigeants des consommateurs et des nombreux objets connectés. Ce travail ne fut qu'une infime introduction au vaste domaine des

télécommunications qui ne cesse de croitre et qui est devenu un élément crucial au bon fonctionnement de la société moderne.

Références

[1] L. J. Vora "Evolution of mobile generation technology: 1G to 5G and review of upcoming wireless technology 5G," *International journal of modern trends in engineering and research*, 2015, vol. 2, no. 10, pp. 281-290.

[2] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE communications magazine*, 2014, vol. 52, no. 2, pp. 186-195.

[3] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE signal processing magazine*, 2012, vol. 30, no. 1, pp. 40-60.

[4] E. Björnson, E. G. Larsson and T. L. Marzetta, "Massive MIMO: 10 myths and one grand question," *arXiv preprint arXiv:1503.06854*, 2015, vol. 1, no. 2.2.

[5] Y. S. Cho, J. Kim, W. Y. Yang and C. G. Kang *MIMO-OFDM wireless communications with MATLAB*. John Wiley & Sons, 2010.

[6] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Dec. 2017.

[7] N. Samuel, T. Diskin and A. Wiesel, "Deep MIMO detection," *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, 2017, pp. 1-5.

[8] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in *Proc. IEEE Int. Workshop Sensing, Processing and Learning for Intelligent Machines (SPLINE)*, 2016, pp. 1–5.

[9] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning based channel decoding," in *Proc. IEEE 51st Annu. Conf. Inf. Sciences Syst. (CISS)*, 2017, pp. 1–6.

[10] T. Schenk, *RF imperfections in high-rate wireless systems: Impact and digital compensation*. Springer Science & Business Media, 2008.

[11] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[12] V. Raj and S. Kalyani, "Backpropagating through the air: Deep learning at physical layer without channel models," in *IEEE Communications Letters*, vol. 22, no. 11, pp. 2278-2281, Nov. 2018.

[13] H. Ye, G. Y. Li and B. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," in *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114-117, Feb. 2018.

[14] M. Borgerding, P. Schniter. "Onsager-corrected deep learning for sparse linear inverse problems," 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2016.

[15] S. Neev, T. Diskin, A. Wiesel. "Learning to detect," *IEEE Transactions on Signal Processing*, 2019, vol. 67, no 10, p. 2554-2564.

[16] S. Ali, W. Saad, N. Rajatheva, K. Chang, D. Steinbach, B. Sliwa and H. Malik, "6G white paper on machine learning in wireless communication networks," *arXiv preprint arXiv:2004.13875*, 2020.

[17] N. Rajatheva, I. Atzeni, E. Bjornson, A. Bourdoux, S. Buzzi, J. B. Dore and W. Xu, "White paper on broadband connectivity in 6G, " *arXiv preprint arXiv:2004.14247*, 2020.

[18] H.Q. Ngo, "*Massive MIMO: fundamentals and system designs*," Ph.D. Thesis. Linköping University Electronic Press, 2015.

[19] M. Wu, B. Yin, G. Wang, C. Dick, J.R. Cavallaro, and C. Studer, "Large-scale MIMO detection for 3GPP LTE: algorithms and FPGA implementations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, 2014, pp. 916–929.

[20] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X. You, "Efficient architecture for soft-output massive MIMO detection with Gauss-Seidel method," IEEE Int. Symp. on Circuits and Systems, May 2016, pp. 1886–1889.

[21] B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "VLSI design of largescale soft-output MIMO detection using conjugate gradients," IEEE International Symposium on Circuits and Systems, May 2015, pp. 1498–1501.

[22] Z. Wu, Y. Xue, X. You, and C. Zhang, "Hardware efficient detection for massive MIMO uplink with parallel Gauss-Seidel method," International Conference on Digital Signal Processing, August 2017, pp. 1–5.

[23] Z. Zhang, J. Wu, X. Ma, Y. Dong, Y. Wang, S. Chen, and X. Dai, "Reviews of recent progress on low-complexity linear detection via iterative algorithms for massive MIMO systems," IEEE International Conference on Communication, July 2016, pp. 1–6.

[24] X. Qin, Z. Yan, and G. He, "A near-optimal detection scheme based on joint steepest descent and Jacobi method for uplink massive MIMO systems," IEEE Communication Letter, vol. 20, no. 2, Feb. 2016, pp. 276–279.

[25] M. Wu, C. Dick, J.R. Cavallaro, and C. Studer, "High-throughput data detection for massive MU-MIMO-OFDM using coordinate descent," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, 2016, pp. 2357–2367.

[26] B. Yin, M. Wu, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "A 3.8Gb/s large-scale MIMO detector for 3GPP LTE-advanced," Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc., May 2014, pp. 3879–3883.

[27] J. Zhou, Y. Ye, and J. Hu, "Biased MMSE soft-output detection based on Jacobi method in massive MIMO," Proc. IEEE Int. Conference on communication problem-solving, Dec 2014, pp. 442–445.

[28] W. Song, X. Chen, L. Wang, and X. Lu, "Joint conjugate gradient and Jacobi iteration based low complexity precoding for massive MIMO systems," IEEE International Conference on Communication, July 2016, pp. 1–5.

[29] M. Ahmed Ouameur, D. Massicotte, A. M. Akhtar and R. Girald, "Performance Evaluation and Implementation Complexity Analysis Framework for ZF Based Linear Massive MIMO Detection," Wireless Networks Journals, Springer, pp. 1–15, April 2020.

[30] J. E. Gentle, Matrix Algebra: Theory, computations and applications in statistics, Springer, 2007, ISBN: 978-0-387-70872-0.

[31] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta and P. Popovski, "Five disruptive technology directions for 5G," IEEE Commun. magazine, vol. 52, no. 2, Feb. 2014.

[32] T. L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," IEEE Trans. Wireless Commun., vol. 9, no. 11, Nov. 2010.

[33] J. Hoydis, K. Hosseini, S. T. Brink, and M. Debbah, "Making Smart Use of Excess Antennas: Massive MIMO, Small Cells, and TDD," Bell Labs Technical Journal, vol. 18, no. 2, Sep. 2013.

[34] H. Q. Ngo, Massive MIMO: Fundamentals and System Designs, Ph.D. Thesis, Linköping University Electronic Press, 2015.

[35] N. Shariati, E. Björnson, M. Bengtsson and M. Debbah, "Low-Complexity Polynomial Channel Estimation in Large-Scale MIMO With Arbitrary Statistics," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 815-830, Oct. 2014.

[36] H. Yin, D. Gesbert, M. Filippou, and Y. Liu, "A coordinated approach to channel estimation in large-scale multiple-antenna systems," IEEE J. Sel. Areas Commun., vol. 31, no. 2, pp. 264–273, 2013.

[37] A. Bourdoux and L. Van der Perre, "Analysis of non-reciprocity impact and possible solutions," Technical report, MAMMOET project, ref. no. ICT-619086/D2.4, September 2015.

[38] J. Vieira, F. Rusek, and F. Tufvesson, "Reciprocity calibration methods for massive MIMO based on antenna coupling," 2014 IEEE Global Communications Conference, Austin, TX, 2014, pp. 3708-3712.

[39] F. Kaltenberger, H. Jiang, M. Guillaud, and R. Knopp, "Relative channel reciprocity calibration in MIMO/TDD systems," in Future Network and Mobile Summit, 2010, June 2010, pp. 1–10

[40] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical many-antenna base stations," in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 53–64.

[41] M. Ahmed Ouameur and D. Massicotte, "Successive Column-wise Matrix Inversion Update for Large Scale Massive MIMO Reciprocity Calibration," accepted in IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, April 2019.

[42] J. Vieira, F. Rusek, O. Edfors, S. Malkowsky, L. Liu and F. Tufvesson, "Reciprocity Calibration for Massive MIMO: Proposal, Modeling, and Validation," in IEEE Transactions on Wireless Communications, vol. 16, no. 5, pp. 3042-3056, May 2017.

[43] R. Rogalin *et al.*, "Scalable Synchronization and Reciprocity Calibration for Distributed Multiuser MIMO," in *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 1815-1831, April 2014.

[44] K. S. Arun, 'A unitary constrained total least squares problem in signal processing,' SIAM J. Matrix Anal. Appl. Vol. 13, no. 3, pp. 728-745.

[45] A. Cichocki and R. Unbehauen, "Simplified neural networks for solving linear least squares and total least squares problems in real time," in IEEE transactions on neural networks, 1994, vol. 5, no 6, pp. 910-923.

[46] Xilinx, Vivado Design Suite User Guide, High-Level Synthesis UG902 (v2019.1) July 12, 2019.

[47] Xilinx, SDx Pragma Reference Guide UG1253 (v2019.1) June 5, 2019

[48] D. Massicotte, S. Legendre and A. Barwicz, "Neural-network-based method of calibration and measurand reconstruction for a high-pressure measuring system, " *IEEE Transactions on Instrumentation and Measurement*, 1998, vol. 47, no. 2, pp. 362-370.

Annexe A – Articles de référence

Deep Unfolded Extended Conjugate Gradient Method for Massive MIMO Processing with Application to Reciprocity Calibration

Samuel Sirois, Messaoud Ahmed Ouameur and Daniel Massicotte

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering,
3351, Boul. des Forges, Trois-Rivières, Québec, Canada
Laboratoire des Signaux et Systèmes Intégrés,
{samuel.sirois, messaoud.ahmed.ouameur, daniel.massicotte}@uqtr.ca

***Abstract*—In this paper, we consider deep unfolding the standard iterative conjugate gradient (CG) algorithm to solve a linear system of equations. Instead of being adjusted with known rules, the parameters are learned via backpropagation to yield the optimal results. However, the proposed unfolded CG (UCG) is extended wherein a scalar parameter is substituted by a matrix-parameter to augment the degrees of freedom per layer. Once the training is completed, the UCG has revealed to require far a smaller number of layers than the number of iterations needed using the standard iterative CG. It is also shown to be very robust to noise and outperforms the standard CG in low signal to noise ratio (SNR) region. A key merit of the proposed approach is the fact that no explicit training data is dedicated to the learning phase as the optimization process relies on the residual error which is not explicitly expressed as a function of the desired data. As an example, the proposed UCG is applied to solve the reciprocity calibration problem encountered in massive MIMO (Multiple-Input Multiple-Output) systems.**

***Index Terms*— Deep unfolding, conjugate gradient, Massive MIMO, reciprocity calibration, least squares.**

I. Introduction

Conjugate gradient (CG) algorithm is widely used to iteratively solve a large linear system of equations. It is well known to converge rapidly in sparse systems. Otherwise, it is assured to converge in the number of iterations less than or equal the number of unknowns [1-page 214]. It is worth noting that the residual error computed at every iteration doesn't depend on the explicit real solution of the problem. This fact is exploited in the proposed deep unfolded algorithm architecture to train a network without explicit knowledge of the training data.

Recent contributions advocate for the potential of using deep learning (DL) for communication system designs [2]-[6]. As such, some initial insights and findings, using state-of-the-art DL tools, on signal compression [4] and channel decoding [5] are revealed. On the other hand, massively parallel processing architectures, such as graphic processing units (GPUs), have shown to be very energy efficient with remarkable computational capabilities when fully exploited by concurrent algorithms [7]. So far, the goal of introducing DL is to either improve parts of existing algorithms or to completely replace them with an end-to-end approach [8] [9]. As an example, the authors in [2] have discussed several promising new applications of DL to the physical layer. On the other hand, two different deep architectures for point-to-point MIMO detection are introduced in [8] wherein the promising architecture relies on unfolding the iterations of a projected gradient descent algorithm into a network. Deep unfolding to solve a general system of equations consists of using the structure of a known iterative algorithm and consider every iteration as a layer of a neural network. Every parameter in the iterative method that is normally updated with a deterministic rule is instead trained with backpropagation process to yield optimum results in solving the system of equations. Certain parameters can be added or modified in the deep unfolded network to give more degrees of freedom in the training process to allow the algorithm to capture features that would not be considered with the original iterative method. This can augment the complexity of one layer compared to one iteration of the original algorithm but in the end, the number of layers required to solve the problem can be drastically reduced to yield an overall computation complexity and latency smaller than the one obtained with the standard iterative method. Also, another advantage to unfold an algorithm is that once the training is properly done, which can be very tedious sometimes, the network will end up capturing all the inherent details of the problem and can always find a solution with its parameters fixed, whereas the iterative algorithm usually needs to recompute its coefficients every time the system to solve has changed. In other words, once the training is done for a big variety of inputs, the network can generalize for all other inputs. The secret resides in the strategy to train the network to be sure that the parameters are not overtrained or undertrained. In this paper, we will show a training method we used to solve the least-squares (LS) problem inherent to a massive MIMO (Multiple-Input Multiple-Output) reciprocity calibration problem. Another advantage of deep unfolding is that activation functions can be added to the layers to consider nonlinearity of the given problem.

Deep unfolding methods were successfully developed in [10] and [11] to solve a linear system of equations. In [10], the authors deep unfolded the approximate message passing (AMP) iterative algorithm to solve the sparse linear inverse problem. Training data was necessary to learn the parameters of each layer. In [11], the authors unfolded the iterations of a projected gradient descent algorithm and applied their network to the massive MIMO detection problem.

To the best of our knowledge, all deep unfolded algorithm used to solve a linear system of equations need explicit and dedicated knowledge of the training data to be efficient. As

pinpointed above, the residual error parameter in the CG algorithm can be exploited to create a deep network that can train without having access to explicit training data. This can be useful in many applications such as in reciprocity calibration and detection problems that are encountered in massive MIMO systems. To gain insights on the potential of this property, we proposed an extended deep unfolded conjugate gradient (UCG) architecture, wherein the parameters are learned via backpropagation, and we applied it to solve the reciprocity calibration problem.

As a matter of fact, massive MIMO is the key technology for the new mobile network generation also known as 5G [12]-[13]. Massive MIMO in the context of the 5G is defined as classical multiuser MIMO with every base station (BS) having a number of antennas much larger than the number of users (UTs) served at the same time [14]-[15]. Massive MIMO technology works in time division duplex (TDD) mode using channel state information (CSI) of the uplink channel for both uplink detection and downlink beamforming [16]-[17]. Unfortunately, the uplink and the downlink channels are not reciprocal. This implies that if the uplink CSI is used for beamforming, the sum-rate capacity performances would degrade [18]-[19]. This non-reciprocity is due to the impairments in the radio frequency (RF) chains in both the transmitters and the receivers of the BS and the UTs. In other words, reciprocity calibration is required to implicitly infer the CSI of the downlink channel [18]. Details of the impacts of non-reciprocal channels and possible solutions are addressed in [18]. Also, a more detailed introduction to the reciprocity calibration problem is addressed in [20] whereas the calibration solution is implemented in a real testbed in [21]. The main idea is that a reference antenna at the BS is used to do the calibration by sending and then receiving signals to and from the other antennas at the BS. Based on the collected data signals, calibration can be done by solving an optimization problem. Unfortunately, the performance of this method varies with the position of the reference antenna and is very noise sensitive. For distributed MIMO systems, access point calibration is proposed in [22] and [23] to generalize the method used in [21]. To eliminate the position-sensitive reference antenna issue, the authors in [19] proposed several LS estimators based on the mutual-coupling between each antenna pair at the BS. In other words, each antenna can be seen as a reference antenna that sends and then receives signals to and from the other antennas at the BS. The authors in [24] proposed a new efficient way to inverse a large matrix based on a successive column-wise update algorithm to solve the LS problem formulation related to the reciprocity calibration. From another point of view, the authors in [25] formulated an expectation-maximization algorithm (EM) that iteratively improved results obtained from other methods such as LS. Both LS and EM methods are well suited for collocated and distributed MIMO BS [23].

The LS problem formulation implies the inversion of a $(M-1)\times(M-1)$ matrix to solve a large linear system of equations, where M is the number of antennas at the BS. Even if the calibration needs to be done approximately on an hourly basis for collocated MIMO systems, fast algorithms are necessary in the case of distributed MIMO systems where the reference clocks are not the same so that the calibration needs to be done frequently. With that in mind, we propose the UCG algorithm to solve the linear system of equations inherent to the reciprocity calibration problem.

There exist several efficient iterative techniques to solve a linear system of equations. These methods are well suited for problems where noise is only present on one side of the system of equations. In the LS problem formulation of reciprocity calibration, noise is present on both sides of the equality sign of the system of linear equations so that classical algorithms such as Gauss-Seidel (GS) and Neumann series expansion (NSE) need a lot of iterations to find a good solution [24].

The reason why we choose to unroll the CG algorithm, instead of other known iterative methods, is that it has a residual error parameter upon which the loss function is computed. A trigger can be set using this parameter to know when to stop learning without having access to explicit training data, which is the case in the reciprocity calibration problem. Also, the CG algorithm is one of the most popular methods to solve a large system of linear equations due to its quick convergence. Furthermore, the reason we decided to use deep unfolding technique in this paper instead of a normal deep neural network (DNN) is because the inner architecture is specially designed to solve linear system of equations whereas DNN is a more general architecture that suits for all sorts of problems so that it would probably need more layers to obtain similar results. This results in a higher algorithm complexity that needs to be avoided in most problems. Some may argue that the nonlinear activation functions of a standard DNN can efficiently solve the reciprocity calibration problem, but nothing prevents us to add a nonlinear function on the output of every layer of the UCG method. Nevertheless, we have chosen to keep the activation function linear.

Therefore, our approach starts by unfolding the iterative CG method to infer the data tree dependence graph over which the gradients are computed. As such, the contributions are: (i) We propose a data dependence tree graph based on unfolding the iterative CG algorithm where the parameters are now trained rather than being explicitly computed. (ii) We derive closed-form expressions of the gradients of the loss function with respect to the parameters. The loss function is set to be the squared norm of the residual error term, which as mentioned above is not an explicit function of the training data. (iii) We discuss the proposed algorithm performance in comparison to the standard iterative CG in the reciprocity calibration use case.

The paper is organized as follows: Section II presents the deep UCG method. Section III summarizes the reciprocity calibration problem. Section IV discusses some simulation results obtained with the UCG. Finally, a conclusion is drawn.

II. Deep Unfolded Extended Conjugate Gradient

Consider solving a real valued linear system of equations of the form $\mathbf{Ax} = \mathbf{y}$ with $\mathbf{A}$ being a squared positive-definite matrix of dimensions $(2M-2)\times(2M-2)$, where M is the number of antennas at the BS in the reciprocity calibration problem context. The standard iterative CG algorithm initialization steps [1-page 214] are depicted as follow

$$\hat{\mathbf{x}}_1 = \textit{Initial guess} \tag{1}$$

$$\mathbf{r}_1 = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}_1 \tag{2}$$

$$\mathbf{s}_1 = \mathbf{A}\mathbf{r}_1 \tag{3}$$

$$\mathbf{p}_1 = \mathbf{s}_1 \tag{4}$$

$$\gamma_1 = \|\mathbf{s}_1\|^2 \tag{5}$$

where $\hat{\mathbf{x}}_1$ is the first approximate solution of $\mathbf{x}$. Once these initializations are done, the main CG algorithm for the k^{th} iteration goes as follow

$$\mathbf{q}_k = \mathbf{A}\mathbf{p}_k \tag{6}$$

$$\alpha_k = \frac{\gamma_k}{\|\mathbf{q}_k\|^2} \tag{7}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k \tag{8}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k \tag{9}$$

$$\mathbf{s}_{k+1} = \mathbf{A}\mathbf{r}_{k+1} \tag{10}$$

$$\gamma_{k+1} = \|\mathbf{s}_{k+1}\|^2 \tag{11}$$

$$\mathbf{p}_{k+1} = \mathbf{s}_{k+1} + \frac{\gamma_{k+1}}{\gamma_k}\mathbf{p}_k \tag{12}$$

Hence, we propose deep unfolding the conjugate gradient iterative algorithm into few layers and learn the parameters α_k and $\lambda_k \triangleq \gamma_{k+1}/\gamma_k$ via backpropagation. In fact, the original problem is a complex valued linear system of equations $\overline{\mathbf{A}}\,\overline{\mathbf{x}} = \overline{\mathbf{y}}$, with $\overline{\mathbf{A}}$ being a square matrix of dimensions $(M-1)\times(M-1)$. We will use the following transformation to convert the problem into a real valued one.

$$\breve{\mathbf{y}} = \begin{pmatrix} \mathrm{Re}(\overline{\mathbf{y}}) \\ \mathrm{Im}(\overline{\mathbf{y}}) \end{pmatrix} \in \mathbb{R}^{(2M-2)\times 1} \tag{13}$$

$$\breve{\mathbf{A}} = \begin{pmatrix} \mathrm{Re}(\overline{\mathbf{A}}) & -\mathrm{Im}(\overline{\mathbf{A}}) \\ \mathrm{Im}(\overline{\mathbf{A}}) & \mathrm{Re}(\overline{\mathbf{A}}) \end{pmatrix} \in \mathbb{R}^{(2M-2)\times(2M-2)} \tag{14}$$

$$\mathbf{x} = \begin{pmatrix} \mathrm{Re}(\overline{\mathbf{x}}) \\ \mathrm{Im}(\overline{\mathbf{x}}) \end{pmatrix} \in \mathbb{R}^{(2M-2)\times 1} \tag{15}$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ denote the real and imaginary part of the terms in parenthesis respectively. Since the CG algorithm can only solve for positive definite symmetric matrix, we define

$$\mathbf{A} = \breve{\mathbf{A}}^T\breve{\mathbf{A}} \in \mathbb{R}^{(2M-2)\times(2M-2)} \tag{16}$$

$$\mathbf{y} = \breve{\mathbf{A}}^T\breve{\mathbf{y}} \in \mathbb{R}^{(2M-2)\times 1} \tag{17}$$

with these conventions established, we can write

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{18}$$

and solve for $\mathbf{x}$ and then after retrieve the complex form $\overline{\mathbf{x}}$ using (15).

The conjugate gradient method iteratively computes an estimate of $\mathbf{x}$ denoted as $\hat{\mathbf{x}}$. To unfold the algorithm, equations (6) and (9) are combined, equations (8) and (10) are used as-is and the scalar term $\frac{\gamma_{k+1}}{\gamma_k}$ in equation (12) is replaced by the matrix $\boldsymbol{\lambda}$ to yield the following formulas describing the k^{th} layer (unfolded iteration) of the UCG algorithm

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k \tag{19}$$

$$\mathbf{s}_k = \mathbf{A}\mathbf{r}_{k+1} \tag{20}$$

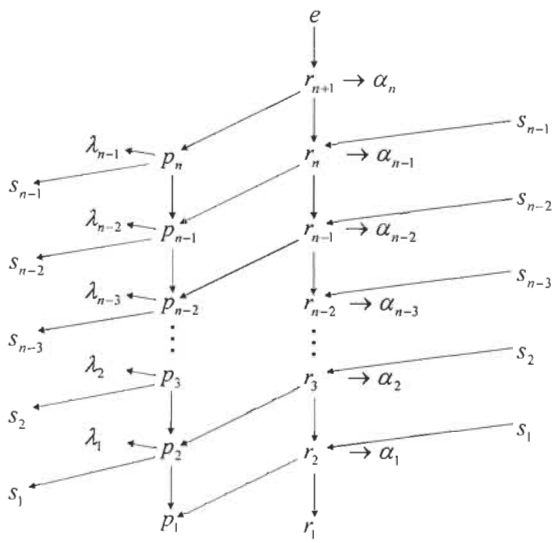


Figure 1. Dependence tree of the unfolded algorithm

$$\mathbf{p}_{k+1} = \mathbf{s}_k + \boldsymbol{\lambda}_k \mathbf{p}_k \tag{21}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k \tag{22}$$

Where $\mathbf{r}$ is the residual error on the estimation of the unknown parameter $\mathbf{X}$ with dimensions $(2M-2)\times 1$, $\mathbf{p}$ and $\mathbf{s}$ are intermediate variables with dimensions $(2M-2)\times 1$, α_k is a scalar parameter and $\boldsymbol{\lambda}$ is a $(2M-2)\times(2M-2)$ matrix parameter to learn. Normally, the parameter $\boldsymbol{\lambda}$ is represented by a scalar in the classical conjugate gradient method. Substituting such a scalar parameter with a matrix is intentionally introduced to provide more degree of freedom to every layer. Hence the name *extended* unfolded conjugate gradient which is we refer to as UCG for simplicity. The use of $\boldsymbol{\lambda}$ as a matrix yields better results against noise effects. The initial conditions for the UCG are defined by

$$\mathbf{r}_1 = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}_1 \tag{23}$$

$$\mathbf{p}_1 = \mathbf{A}\mathbf{r}_1 \tag{24}$$

with the initial guess $\hat{\mathbf{x}}_1$ that can be set to the zero vector or to the best initial guess. Figure 1 shows the dependence tree of the deep unfolded algorithm. The loss function e is based on the L_2 norm applied on $\mathbf{r}_{n+1}$ as

$$e = \mathrm{Loss}(\Theta;\mathbf{A},\mathbf{x}) = \|\mathbf{r}_{n+1}\|_2^2 \tag{25}$$

$$\Theta = \{\alpha_k, \boldsymbol{\lambda}_k\}_{k=1}^{n \text{ for } \alpha \text{ and } (n-1) \text{ for } \lambda} \tag{26}$$

The loss function e is minimized over $\Theta = \{\alpha_k, \boldsymbol{\lambda}_k\}_{k=1}^{n \text{ for } \alpha \text{ and } (n-1) \text{ for } \lambda}$. When the loss function based on the residual error tends to the zero vector, the error on the unknown parameter $\mathbf{x}$ will also tend towards zero. This fact is the key concept of the algorithm. Indeed, as we will see in the next section, it enables the algorithm to decide whether it needs to *retrain* or not based on a threshold derived from the loss function. In addition, since the loss function is not an explicit function of the desired data, it makes the proposed approach blind.

The derivation of the gradient of the loss function with

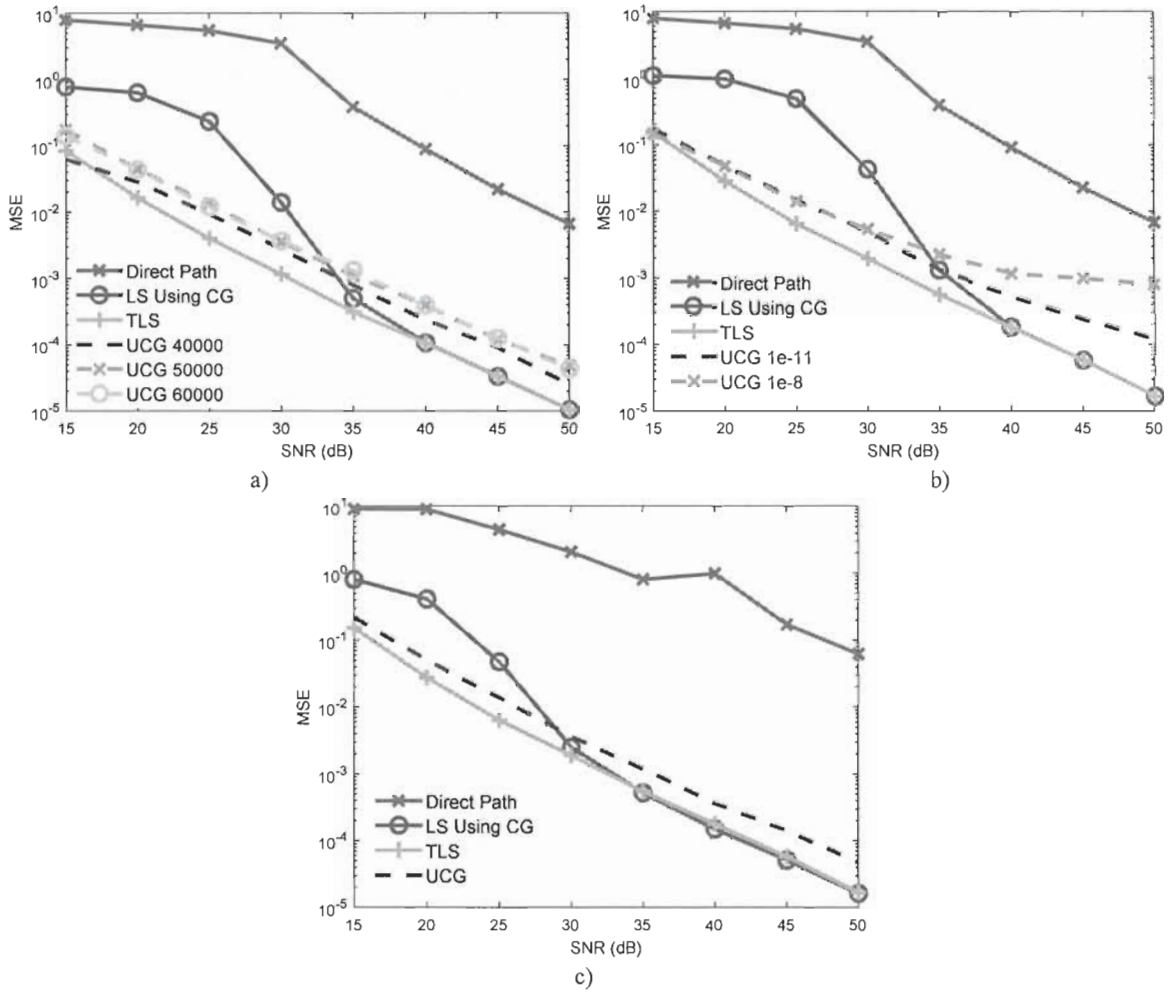


Figure 2. Reciprocity calibration results for UCG algorithm. a) Results obtained with a training SNR of 60 dB and a trigger value set to 10^{-13} for three different sizes of training, namely 40000, 50000 and 60000 iterations per realization of system of equations. b) Results obtained with a training SNR of 45 dB for two different triggers values, namely 10^{-8} and 10^{-11} with a size of training of 40000 iterations per realization of a system of equations. c) Results obtained with 50 antennas with a training SNR of 60 dB and a 10^{-13} trigger value.

respect to α and λ is presented in Appendix A as well as the training method used during the backpropagation process.

III. USE CASE: RECIPROCITY CALIBRATION IN MASSIVE MIMO

A. Least square problem formulation for reciprocity calibration

This sub-section summarizes the LS problem formulation for reciprocity calibration. In fact, solving the LS linear system of equations permits us to find estimations of calibration coefficients. Once these coefficients are computed, they can be multiplied by the uplink channel coefficients to get the implicit downlink channel estimation. For more details on the LS formulation, readers can refer to [19], [21] and [24]. The calibration coefficients $b^{(m)}$, where $m=1,\cdots,M$, are estimated with every antenna at the BS sending and receiving known pilots signals $s_p = +1$ from and to the other antennas. We denote $y^{(m,l)}$ the signal received by the antenna m and transmitted by the antenna l. Grouped in pair, the signals exchanged between two antennas are written as [9]

$$\begin{pmatrix} y^{(l,m)} \\ y^{(m,l)} \end{pmatrix} = \alpha^{(l,m)} \begin{pmatrix} b^{(l)} \\ b^{(m)} \end{pmatrix} + \begin{pmatrix} n^{(l,m)} \\ n^{(m,l)} \end{pmatrix} \tag{27}$$

where $\alpha^{(l,m)} = t_B^{(l)} t_B^{(m)} \tilde{h}^{(l,m)} = t_B^{(m)} t_B^{(l)} \tilde{h}^{(m,l)}$, $t_B^{(m)}$ and $t_B^{(l)}$ are the complex-valued coefficients (gain and phase) that model the RF transmitter of the BS, and $\tilde{h}^{(l,m)} = \tilde{h}^{(m,l)}$ is the reciprocal propagation channel. $\left(n^{(l,m)} \quad n^{(m,l)}\right)^T$ is an independent zero-mean circularly symmetric complex Gaussian distributed random noise vector whose components have variance N_0.

To model the reciprocal channel $\tilde{h}^{(l,m)}$, the authors in [19] empirically determine an equation based on measurements done in an anechoic chamber. The channel CSI between antenna l and antenna m can be approximated as

$$\tilde{h}^{(l,m)} = \beta^{(l,m)} \exp\left(j\phi^{(l,m)}\right) + w^{(l,m)} \tag{28}$$

where $\beta^{(l,m)} = 0.03\left(d^{(l,m)}\right)^{-3.7}$, the distance $d^{(l,m)}$ between the antennas is in multiple of half the wavelength, the phase $\phi^{(l,m)}$ is uniformly distributed in the range $[0, 2\pi]$ and the component $w^{(l,m)}$ is a circularly symmetric complex Gaussian distributed

random variable whose components have a variance N_w to model the weak effects of the scattering due to far elements in the environment.

The direct path method that only use one reference antenna to estimate the calibration coefficients $\tilde{\mathbf{b}} = \left(\tilde{b}^{(0)}, \ \tilde{b}^{(1)}, \ldots, \ \tilde{b}^{(M-1)}\right)^T$ with the pairs of signals $y^{(ref,m)}$ and $y^{(m,ref)}$ can be written as

$$\tilde{b}^{(m)} = y^{(ref,m)} / y^{(m,ref)} \tag{29}$$

where *ref* designates the reference antenna.

The direct path method can be generalized with every antenna at the BS being used as a reference antenna to augment diversity and to get a better LS-based estimator. In other words, all pairs of signals in equation (27) are used in the estimator to yield the following cost function to minimize

$$J\left(\tilde{\mathbf{b}}\right) = \sum_{m,l \neq m} \left\| \tilde{b}^{(m)} y^{(m,l)} - \tilde{b}^{(l)} y^{(l,m)} \right\|^2 \tag{30}$$

which leads to the closed-form solution

$$\tilde{\mathbf{b}} = \left(\tilde{b}^{(1)}, \ \tilde{b}^{(2)}, \ldots, \ \tilde{b}^{(M-1)}\right)^T = -\left(\tilde{\mathbf{A}}_1^H \tilde{\mathbf{A}}_1\right)^{-1} \tilde{\mathbf{A}}_1^H \tilde{\mathbf{a}} \tag{31}$$

where $\tilde{\mathbf{a}}$ is the first column of $\tilde{\mathbf{A}} \triangleq \left[\tilde{\mathbf{a}} \quad \tilde{\mathbf{A}}_1\right]$ and $\tilde{b}^{(1)}$ is set to 1 to avoid the all zeros solution $\tilde{\mathbf{b}} = \mathbf{0}$. This constraint does not impact the quality of the calibration coefficients because the precision of the estimation can be done up to a multiple of a constant without degrading the performances of downlink beamforming. The matrix $\tilde{\mathbf{A}}$ is defined as

$$\tilde{\mathbf{A}}^{(m,l)} = \begin{cases} \sum_{l=0}^{M-1} \left| y^{(m,l)} \right|^2, & m = l \\ -y^{*(m,l)} y^{(l,m)}, & m \neq l \end{cases} \tag{32}$$

Other types of LS estimators such as weighted LS are also presented in [9].

B. *Total least squares estimator*

Rewriting equation (31) leads to $\tilde{\mathbf{A}}_1 \tilde{\mathbf{b}} = -\tilde{\mathbf{a}}$ which has the same form as $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{y}}$ (refer to Section II). When Gaussian noise is only present in $\tilde{\mathbf{a}}$, the LS estimator is an optimal solution to solve for $\tilde{\mathbf{b}}$ [26]. Unfortunately, in the context of reciprocity calibration, noise is present in both $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{a}}$. A more reliable technique than directly inverting the matrix $\tilde{\mathbf{A}}_1$ is to solve the system of equations with the total least squares (TLS) method which is more robust when Gaussian noise is on both sides of the system of equations. The TLS estimator finds a value of $\tilde{\mathbf{b}}$ which minimize

$$\left\| \Delta\tilde{\mathbf{A}}_1 \right\|^2 + \left\| \Delta\tilde{\mathbf{a}} \right\|^2 \tag{33}$$

subject to the equality constraint

$$\left(\tilde{\mathbf{A}}_1 - \Delta\tilde{\mathbf{A}}_1\right)\tilde{\mathbf{b}} = \left(-\tilde{\mathbf{a}} - \Delta\tilde{\mathbf{a}}\right) \tag{34}$$

where $\Delta\tilde{\mathbf{A}}_1$ and $\Delta\tilde{\mathbf{a}}$ define the noise on $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{a}}$ respectively [27]. To obtain the TLS solution, a singular value decomposition (SVD) of the extended matrix $\left(\tilde{\mathbf{A}}_1 \quad \tilde{\mathbf{a}}\right)$ needs to be done in order to retrieve the right singular vector $\mathbf{v} = \left(v^{(1,M)}, \ v^{(2,M)}, \ldots, \ v^{(M,M)}\right)^T$ corresponding to the smallest singular value of the decomposition. One can normalize the estimation as

$$\tilde{\mathbf{b}}_{TLS} = \frac{1}{v^{(1,M)}} \mathbf{v} \tag{35}$$

to set $\hat{b}_{TLS}^{(1)} = 1$. As it will be shown in the results, TLS and LS estimators perform equally well at high SNR because the noise is less important in the system of equations. Unfortunately, this method has a high computational cost for real-time implementation.

IV. Performance Analysis

A. *Simulation set up and parameters*

The simulation set up is inspired from [19] and [24] with a 10×10 planar patch array and the variance N_w of the channel Rayleigh component is set to -50 dB. The center antenna (antenna element 49) is set as the reference element for the direct path method. The RF chain is modeled with the parameters in [23] where the coefficients of the transmitters and the receivers have a uniformly distributed phase in the range of $[-\pi, \pi]$ and their magnitude is uniformly distributed within $[1-\delta, 1+\delta]$, where δ is selected to respect the following condition

$$\sqrt{E\left(\left(\left|t_m^B\right| - 1\right)^2\right)} = \sqrt{E\left(\left(\left|r_m^B\right| - 1\right)^2\right)} = 0.1 \tag{35}$$

where $E(\cdot)$ is the expectation operator. The UCG algorithm has 7 layers (n=7) where α and $\boldsymbol{\lambda}$ are initialized with non-zero small values so that the algorithm doesn't diverge. The ratio μ_1 / μ_2 is approximately 10^5 and the momentum coefficient β is set as close as to 1 to prevent the algorithm from diverging. The reader can refer to appendix A for more details.

The calibration SNR, in the simulation results (c.f. figure 2), is normalized with respect to the SNR of two adjacent antennas. Since solving equation (18) implies a $(2M-2)\times(2M-2)$ matrix inversion, iterative methods such as GS and NSE can be used to solve for $\tilde{\mathbf{b}}$ [24]. Indeed, these methods have been successfully used in the zero-forcing (ZF) and the minimum mean square error (MMSE) massive MIMO detection problem where the direct matrix inversion is computationally expensive. Unfortunately, as it is shown in [24], such methods will require a large number of iterations to converge in the context of solving the LS problem encountered in reciprocity calibration.

B. *Simulation results and discussion*

Figure 2 depicts the MSE as a function of the normalized SNR. The TLS estimator has good results at low SNR compared to the LS because of the presence of the noise in both $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{a}}$. For the simulation purposes, the LS problem is solved with the original conjugate gradient method which would have yielded the same results as if it would have been done with the direct matrix inversion. Also, as expected, when the SNR increases, both LS and TLS estimators tend to reach the same results.

TABLE 2. COMPLEXITY IN TERMS OF REAL ADDITIONS AND MULTIPLICATIONS OF THE INITIALIZATION OF THE CG ALGORITHM

EQ.	ADD.	MULT.
(2.2)	$2(M-1)^2+2(M-1)(M-2)+2(M-1)$	$4(M-1)^2$
(2.3)	$2(M-1)^2+2(M-1)(M-2)$	$4(M-1)^2$
(2.5)	$(M-1)+(M-2)$	$2(M-1)$
TOT	$8M^2-16M+8$	$8M^2-14M+6$

TABLE 3. COMPLEXITY IN TERMS OF REAL ADDITIONS, MULTIPLICATIONS, AND DIVISIONS OF ONE ITERATION OF THE CG ALGORITHM

EQ.	ADD.	MULT.	DIV.
(3.1)	$2(M-1)^2+2(M-1)(M-2)$	$4(M-1)^2$	0
(3.2)	$(M-1)+(M-2)$	$2(M-1)$	1
(3.3)	$2(M-1)$	$2(M-1)$	0
(3.4)	$2(M-1)$	$2(M-1)$	0
(3.5)	$2(M-1)^2+2(M-1)(M-2)$	$4(M-1)^2$	0
(3.6)	$(M-1)+(M-2)$	$2(M-1)$	0
(3.7)	$2(M-1)$	$2(M-1)$	1
TOT	$8M^2-10M$	$8M^2-6M-2$	2

TABLE 4. COMPLEXITY IN TERMS OF REAL ADDITIONS AND MULTIPLICATIONS OF THE INITIALIZATION OF THE UCG ALGORITHM

EQ.	ADD.	MULT.
(17)	$(2M-2)(2M-3)+(2M-2)$	$(2M-2)^2$
(18)	$(2M-2)(2M-3)$	$(2M-2)^2$
TOT.	$8M^2-18M+10$	$8M^2-16M+8$

TABLE 5. COMPLEXITY IN TERMS OF REAL ADDITIONS AND MULTIPLICATIONS OF ONE LAYER OF THE UCG ALGORITHM

EQ.	ADD	MULT
(13)	$(2M-2)(2M-3)+(2M-2)$	$(2M-2)^2+(2M-2)$
(14)	$(2M-2)(2M-3)$	$(2M-2)^2$
(15)	$(2M-2)(2M-3)+(2M-2)$	$(2M-2)^2$
(16)	$(2M-2)$	$(2M-2)$
TOT	$12M^2-24M+12$	$12M^2-20M+8$

TABLE 6. COMPLEXITY IN TERMS OF REAL ADDITIONS AND MULTIPLICATIONS TO MAKE THE SYSTEM OF EQUATION POSITIVE DEFINITE SYMMETRIC FOR THE CG AND THE UCG

ALG.	ADD.	MULT.
CG	$2M^3+M^2-5M+2$	$2M^3+2M^2-4M$
UCG	$4M^3-4M^2-M+1$	$4M^3-2M^2-2M$

TABLE 7. TOTAL COMPLEXITY IN TERMS OF REAL ADDITIONS AND MULTIPLICATIONS FOR THE CG AND THE UCG

ALG.	ADD.	MULT.
CG	$2M^3+2021M^2-2543M+19$	$2M^3+2006M^2-1516M-492$
UCG	$4M^3+80M^2-169M+85$	$4M^3+82M^2-142M+56$

Of particular interest is the proposed UCG method which only requires 7 layers and has better results than LS at low SNR. The original CG algorithm requires approximately 250 iterations to converge and it gets the same results as the LS method. These improvements are mainly due to the fact that λ is a matrix that provides extra degrees of freedom in the learning process. Based on our early simulation results, unfolding the CG algorithm without extending λ as a matrix parameter shows similar performances as the normal iterative algorithm, so that λ has to be a matrix to improve the performance.

The complexity per layer/iteration in UCG becomes more important because it requires the multiplication of a matrix with a vector instead of a scalar with a vector. Tables 2, 3, 4 and 5 show the complexity of both CG and UCG in terms of real additions (ADD) and multiplications (MULT) with respect to the number of antennas at the BS. It is important to mention that the complexity of the CG has been calculated with the assumption that equations (13), (14) and (15) are not used. In other words, the CG algorithm solves directly $\tilde{\mathbf{A}}_1^H\tilde{\mathbf{A}}_1\tilde{\mathbf{b}}=\tilde{\mathbf{A}}_1^H(-\tilde{\mathbf{a}})$. This explains why the complexity to make the system of equations positive definite symmetric depicted in Table 6 is not the same for both algorithms. The reason we introduced the convention of equations (13), (14) and (15) for UCG was for sake of simplicity during the development process of the algorithm. The UCG could have been derived using the Wirtinger derivatives to yield even better complexity results. Therefore, for large M, it is easy to see that one layer of the UCG requires approximately $4M^2$ more additions and multiplications than one iteration of the CG. However, matrix-vector multiplication can easily be parallelized on GPU or on field-programmable gate array (FPGA) to compensate this disadvantage. Also, since UCG requires far fewer layers than the CG, the overall complexity of the UCG becomes much smaller than the complexity of the CG. In fact, the total complexity of both CG and UCG are shown in Table 7. As expected, the UCG is far computationally less expensive than the CG.

To train the algorithm for one particular value of the calibration coefficients vector, 5 to 15 realizations, depending on the value of the threshold of the system of equation $\tilde{\mathbf{A}}_1\tilde{\mathbf{b}}=-\tilde{\mathbf{a}}$ with different noise values, are necessary. This represents a big advantage for heterogeneous systems because a low interconnect bandwidth is used to transmit data. For example, the training phase of the UCG could be done on a GPU and the application of the algorithm could be implemented in an FPGA. To continue, on Figure 2.a, the backpropagation process is performed with 40000, 50000 and 60000 iterations per realization of a system of equations respectively. The optimal number of iterations is obviously 40000. On the other hand, simulations with a number of iterations below 35000 do not converge. Figure 2.b shows the performances of the UCG with training at an SNR of 45 dB with the optimal number of iterations and two different trigger values of 10^{-8} and 10^{-11} respectively. One can easily see the effect of the trigger on the performances of the UCG method. That said, when the trigger is set correctly, the MSE is excellent only for the values of SNR below the value of the training SNR. This is not a problem since with the performances of the generalized LS method, the sum-rate loss is negligible at a calibration SNR above 20 dB for maximum ratio transmission (MRT) precoding and above 35

dB for ZF precoding [18], [19]. Hence, as long as the training is done in a region of the SNR where the performances of the UCG are better than those of the generalized LS at 20 dB or 35 dB, no problem should occur. For sake of comparison, Figure 2.c shows the UCG performances with 50 antennas instead of 100. The results for UCG seem to be independent of the number of antennas whereas the LS method has a better MSE values in high SNR region as the number of antennas decreases. It is also worth mentioning that the training SNR can vary to ±5 dB from a system of equations to another without perturbing the performances. Once the training is done for one particular value of the calibration coefficients vector, one or more layers of the UCG can be updated every time the calibration needs to be redone. Eventually, the unfolded algorithm would consider the effects of the slight variation with time and external factors (temperature, humidity, clock drift, etc.) of the parameters of the system of equations $\tilde{\mathbf{A}}_i\tilde{\mathbf{b}}=-\tilde{\mathbf{a}}$ until it doesn't need to retrain very often. This implies a sacrifice in the complexity of the UCG in the learning phase but once it is done, the complexity becomes as depicted in Table 7. Figure 2 shows the results with a fixed value of the calibration coefficients vector. Evaluation of the performances of the UCG algorithm with coefficients of calibration varying with time and external factors is out of the scope of this paper and is reserved for future work.

Also, as mentioned in the previous section, the UCG algorithm is blind and knows when to stop learning with the value of the threshold set on the loss function. In other words, the algorithm keeps doing the backpropagation process as long as the value of the loss function is above the threshold value. This is a big advantage compared to other unfolded algorithms such as the learned iterative hard-thresholding (LISTA) method [10] and the Detection Network (DetNet) [11] which need explicit training data in order to learn their parameters. The goal is then to try to find the smallest trigger depending on the training SNR that the algorithm can reach during the learning process. This enables the UGC to first evaluate the solution of a system of equations and then decides if it needs to *retrain* or not even if it doesn't have access to the solution.

Finally, it is worth mentioning that the UCG can be seen as a linear neural network. That being said, nonlinear activation functions could be added to enhance the algorithm in a real-world implementation where nonlinearities often occur. This is also part of future work.

V. Conclusion

Deep unfolding for solving a large linear system of equations consists of taking separately each iteration of an iterative algorithm and consider them as a layer of a deep neural network. Instead of updating the parameters within each layer with a known formula, backpropagation is used to optimally adjust them to fit the specific problem. Therefore, the conjugate gradient algorithm is successfully unfolded to solve a large linear system of equations. Also, a scalar parameter within the algorithm is transformed into a matrix parameter to augment the versatility of the deep unfolded network. We have chosen to unfold the conjugate gradient method because there is an intrinsic parameter (residual error) calculated at every iteration to deduce if the algorithm is close or not to an acceptable solution. In other words, without learning data, we can set a threshold on this parameter to know when to stop training the deep unfolded network. This makes indeed the proposed network blind. In many real-life problems, access to training data is very difficult to obtain so that our proposed UCG method can overcome this issue. For example, in the massive MIMO context, the reciprocity calibration problem requires to solve a linear system of equations based on a LS estimator without having access to learning pilots. Toward that end, we have shown that our network can solve the problem with much fewer layers than the number of iterations required with the original conjugate gradient algorithm. Also, at low SNR, our method has performances close to the TLS algorithm which is considered to be optimal.

Acknowledgments

This work has been funded by the Natural Sciences and Engineering Research Council of Canada, Prompt and NUTAQ Innovation, and the CMC Microsystems for equipments. This work was supported by the Chaire de recherche sur les signaux et l'intelligence de systems haute performance for technical support.

References

[1] J. E. Gentle, Matrix Algebra: Theory, computations and applications in statistics, Springer, 2007, ISBN: 978-0-387-70872-0.

[2] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Dec. 2017.

[3] N. Samuel, T. Diskin and A. Wiesel, "Deep MIMO detection," *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, 2017, pp. 1-5.

[4] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in Proc. IEEE Int. Workshop Sensing, Processing and Learning for Intelligent Machines (SPLINE), 2016, pp. 1–5.

[5] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning based channel decoding," in Proc. IEEE 51st Annu. Conf. Inf. Sciences Syst. (CISS), 2017, pp. 1–6.

[6] T. Schenk, RF imperfections in high-rate wireless systems: Impact and digital compensation. Springer Science & Business Media, 2008.

[7] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 127–138, 2017.

[8] V. Raj and S. Kalyani, "Backpropagating through the air: Deep learning at physical layer without channel models," in IEEE Communications Letters, vol. 22, no. 11, pp. 2278-2281, Nov. 2018.

[9] H. Ye, G. Y. Li and B. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," in IEEE Wireless Communications Letters, vol. 7, no. 1, pp. 114-117, Feb. 2018.

[10] M. Borgerding, P. Schniter. "Onsager-corrected deep learning for sparse linear inverse problems," 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2016.

[11] S. Neev, T. Diskin, A. Wiesel. "Learning to detect," IEEE Transactions on Signal Processing, 2019, vol. 67, no 10, p. 2554-2564.

[12] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta and P. Popovski, "Five disruptive technology directions for 5G," IEEE Commun. magazine, vol. 52, no. 2, Feb. 2014.

[13] T. L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," IEEE Trans. Wireless Commun., vol. 9, no. 11, Nov. 2010.

[14] J. Hoydis, K. Hosseini, S. T. Brink, and M. Debbah, "Making Smart Use of Excess Antennas: Massive MIMO, Small Cells, and TDD," Bell Labs Technical Journal, vol. 18, no. 2, Sep. 2013.

[15] H. Q. Ngo, Massive MIMO: Fundamentals and System Designs, Ph.D. Thesis, Linköping University Electronic Press, 2015.

[16] N. Shariati, E. Björnson, M. Bengtsson and M. Debbah, "Low-Complexity Polynomial Channel Estimation in Large-Scale MIMO With

Arbitrary Statistics," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 815-830, Oct. 2014.
[17] H. Yin, D. Gesbert, M. Filippou, and Y. Liu, "A coordinated approach to channel estimation in large-scale multiple-antenna systems," IEEE J. Sel. Areas Commun., vol. 31, no. 2, pp. 264–273, 2013.
[18] A. Bourdoux and L. Van der Perre, "Analysis of non-reciprocity impact and possible solutions," Technical report, MAMMOET project, ref. no. ICT-619086/D2.4, September 2015.
[19] J. Vieira, F. Rusek, and F. Tufvesson, "Reciprocity calibration methods for massive MIMO based on antenna coupling," 2014 IEEE Global Communications Conference, Austin, TX, 2014, pp. 3708-3712.
[20] F. Kaltenberger, H. Jiang, M. Guillaud, and R. Knopp, "Relative channel reciprocity calibration in MIMO/TDD systems," in Future Network and Mobile Summit, 2010, June 2010, pp. 1–10
[21] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical many-antenna base stations," in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 53–64.
[22] E. Björnson, E. G. Larsson and T. L. Marzetta, "Massive MIMO: ten myths and one critical question," in *IEEE Communications Magazine*, vol. 54, no. 2, pp. 114-123, February 2016.
[23] R. Rogalin *et al.*, "Scalable Synchronization and Reciprocity Calibration for Distributed Multiuser MIMO," in *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 1815-1831, April 2014.
[24] M. Ahmed Ouameur and D. Massicotte, "Successive Column-wise Matrix Inversion Update for Large Scale Massive MIMO Reciprocity Calibration," accepted in IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, April 2019.
[25] J. Vieira, F. Rusek, O. Edfors, S. Malkowsky, L. Liu and F. Tufvesson, "Reciprocity Calibration for Massive MIMO: Proposal, Modeling, and Validation," in IEEE Transactions on Wireless Communications, vol. 16, no. 5, pp. 3042-3056, May 2017.
[26] K. S. Arun, 'A unitary constrained total least squares problem in signal processing,' SIAM J. Matrix Anal. Appl. Vol. 13, no. 3, pp. 728-745.
[27] A. Cichocki and R. Unbehauen, "Simplified neural networks for solving linear least squares and total least squares problems in real time," in IEEE transactions on neural networks, 1994, vol. 5, no 6, pp. 910-923.

APPENDIX A

The reader must refer to the dependence tree shown in Figure 1 and to equations (19), (20) and (21) to understand the following derivations. The explicit forms of the gradient of the loss function e with respect to α_k and $\boldsymbol{\lambda}_k$ are

$$\overline{\frac{\partial e}{\partial \alpha_k}} = 2\mathbf{r}_{n+1}^T \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_k}} \tag{A.1}$$

$$\overline{\frac{\partial e}{\partial \boldsymbol{\lambda}_k}} = 2\sum_{z=1}^{2M} \mathbf{r}_{n+1}^{(z)} \overline{\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}} \tag{A.2}$$

where $\overline{\frac{\partial(\cdot)}{\partial(\cdot)}}$ indicates the total derivative and $\overline{\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}}$ represents the total derivative of the z^{th} element of $\mathbf{r}_{n+1}$ with respect to $\boldsymbol{\lambda}_k$ where $k \in [1:n]$ for equation (A.1) and $k \in [1:n-1]$ for equation (A.2). The formulas for the total derivative of $\mathbf{r}_{n+1}$ with respect to α_k and $\boldsymbol{\lambda}_k$ use

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_n}} = \frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_n} \tag{A.3}$$

where the total derivative of $\mathbf{r}_{n+1}$ with respect to α_n corresponds to the only possible path on figure 1 that goes from $\mathbf{r}_{n+1}$ to α_n. Before going further, it is important to notice that the allowed paths are those following the same direction as the arrows in figure 1. Therefore, we have

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \alpha_k}} = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{k+1}}} \frac{\partial \mathbf{r}_{k+1}}{\partial \alpha_k} \tag{A.4}$$

where the chain rule is applied between the total derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{r}_{k+1}$, which depends on every possible path on figure 1 between $\mathbf{r}_{n+1}$ and $\mathbf{r}_{k+1}$, and the straight unique path derivative between $\mathbf{r}_{k+1}$ and α_k with $k \in [1:n-1]$. In a similar way, we derive

$$\overline{\frac{\partial \mathbf{r}_{n+1}^{(z)}}{\partial \boldsymbol{\lambda}_k}}(i,j) = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_{k+1}}}(i,z) \frac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_k}(i,j) \tag{A.5}$$

where the two indices in parenthesis next to every term represent the row and the column of the resulting matrix with i,j,z$\in$[1:2M] and $k \in [1:n-1]$. The left-hand side of equation (A.5) shows that there is a distinct derivative for each element in the vector $\mathbf{r}_{n+1}$ with respect to each (i,j) element in the $\boldsymbol{\lambda}_k$ matrix. The reason why the form of equation (A.5) is slightly different from the one of equation (A.4) is due to the fact that $\boldsymbol{\lambda}_k$ is a matrix instead of being a scalar. We then continue with the total derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{p}_n$

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n}} = \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n} \tag{A.6}$$

which is simply the direct path in Figure 1 between $\mathbf{r}_{n+1}$ and $\mathbf{p}_n$. Things get slightly more complicated for the general form of the derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{p}_k$ with $k \in [2:n-1]$

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_k}} = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_{k+1}}} \left(\frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{p}_k} + \frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{s}_k} \frac{\partial \mathbf{s}_k}{\partial \mathbf{r}_{k+1}} \frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k} \right) + \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{k+1}}} \frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k} \tag{A.7}$$

where the chain rule is first applied between the total derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{p}_{k+1}$, which depends on every possible path on Figure 1 between these two nodes, and the straight path derivative between $\mathbf{p}_{k+1}$ and $\mathbf{p}_k$. In a similar way, the chain rule is applied between the total derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{p}_{k+1}$ and the straight paths derivatives going through $\mathbf{p}_{k+1}$, $\mathbf{s}_k$, $\mathbf{r}_{k+1}$ and $\mathbf{p}_k$ respectively. Finally, the same process is done between the total derivative of $\mathbf{r}_{n+1}$ with respect to $\mathbf{r}_{k+1}$ and the unique straight path derivative between $\mathbf{r}_{k+1}$ and $\mathbf{p}_k$. At that point, the only unknown we are left with is the total derivative of $\mathbf{r}_{n+1}$ with respect to any other $\mathbf{r}$. We begin with

$$\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_n}} = \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_n} + \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n} \frac{\partial \mathbf{p}_n}{\partial \mathbf{s}_{n-1}} \frac{\partial \mathbf{s}_{n-1}}{\partial \mathbf{r}_n} \tag{A.8}$$

where the first possible path is the direct one between $\mathbf{r}_{n+1}$ and $\mathbf{r}_n$ and the second possible path is the one that goes through $\mathbf{r}_{n+1}$, $\mathbf{p}_n$, $\mathbf{s}_{n-1}$ and $\mathbf{r}_n$ respectively. We then continue with

$$\begin{aligned} \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{n-1}}} = \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_n}} \left(\frac{\partial \mathbf{r}_n}{\partial \mathbf{r}_{n-1}} + \frac{\partial \mathbf{r}_n}{\partial \mathbf{p}_{n-1}} \frac{\partial \mathbf{p}_{n-1}}{\partial \mathbf{s}_{n-2}} \frac{\partial \mathbf{s}_{n-2}}{\partial \mathbf{r}_{n-1}} \right) \\ + \frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n} \frac{\partial \mathbf{p}_n}{\partial \mathbf{p}_{n-1}} \frac{\partial \mathbf{p}_{n-1}}{\partial \mathbf{s}_{n-2}} \frac{\partial \mathbf{s}_{n-2}}{\partial \mathbf{r}_{n-1}} \end{aligned} \tag{A.9}$$

where the three possible paths between $\mathbf{r}_{n+1}$ and $\mathbf{r}_{n-1}$ are considered. Finally, we have the general case for $k \in [2:n-2]$

$$\begin{aligned}\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_k}} &= \overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{k+1}}}\left(\frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{r}_k}+\frac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k}\frac{\partial \mathbf{p}_k}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_k}\right)\\ &+\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{p}_n}\left(\prod_{i=n}^{k+1}\frac{\partial \mathbf{p}_i}{\partial \mathbf{p}_{i-1}}\right)\frac{\partial \mathbf{p}_k}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_k}\\ &+\sum_{j=1}^{n-k-1}\left(\overline{\frac{\partial \mathbf{r}_{n+1}}{\partial \mathbf{r}_{n-j+1}}}\frac{\partial \mathbf{r}_{n-j+1}}{\partial \mathbf{p}_{n-j}}\prod_{z=n-j}^{k+1}\left(\frac{\partial \mathbf{p}_z}{\partial \mathbf{p}_{z-1}}\right)\right)\frac{\partial \mathbf{p}_k}{\partial \mathbf{s}_{k-1}}\frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_k}\end{aligned} \tag{A.10}$$

where all possible paths going from $\mathbf{r}_{n+1}$ to $\mathbf{r}_k$ are selected. By differentiating equations (19), (20) and (21), every symbolic derivative can be replaced by its true value

$$\begin{cases}\dfrac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{r}_k}=\mathbf{I},\ \dfrac{\partial \mathbf{r}_{k+1}}{\partial \mathbf{p}_k}=-\alpha_k\mathbf{A}\\ \dfrac{\partial \mathbf{r}_{k+1}}{\partial \alpha_k}=-\mathbf{A}\mathbf{p}_k,\ \dfrac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{r}_k}=\mathbf{A}\\ \dfrac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{p}_k}=\boldsymbol{\lambda}_k,\ \dfrac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{s}_k}=\mathbf{I}\\ \dfrac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_k}=\begin{pmatrix}\mathbf{p}_k^{(1)} & \cdots & \mathbf{p}_k^{(2M)}\\ \vdots & \vdots & \vdots\\ \mathbf{p}_k^{(1)} & \cdots & \mathbf{p}_k^{(2M)}\end{pmatrix}\in\mathbb{R}^{2M\times 2M}\end{cases} \tag{A.11}$$

where $\mathbf{I}$ is a $2M\times 2M$ identity matrix and $\dfrac{\partial \mathbf{p}_{k+1}}{\partial \boldsymbol{\lambda}_k}$ is a matrix with the i^{th} column having a constant value corresponding to the i^{th} element of $\mathbf{p}_k$.

To get better and faster results during the backpropagation process, the adaptation step for α_k and $\boldsymbol{\lambda}_k$ can be set to different values and a momentum technique is used on $\boldsymbol{\lambda}_k$:

$$\boldsymbol{\lambda}_k(\mathrm{t}+1)=\boldsymbol{\lambda}_k(\mathrm{t})-\mu_1\overline{\frac{\partial e}{\partial \boldsymbol{\lambda}_k(t)}}+\beta\left(\boldsymbol{\lambda}_k(\mathrm{t})-\boldsymbol{\lambda}_k(\mathrm{t}-1)\right) \tag{A.12}$$

$$\alpha_k(\mathrm{t}+1)=\alpha_k(\mathrm{t})-\mu_2\overline{\frac{\partial e}{\partial \alpha_k(t)}} \tag{A.13}$$

where μ_1 and μ_2 are the fixed adaptation step, β is the momentum coefficient between 0 and 1 and t represents the current iteration step (epoch) in the backpropagation process.

High Level Synthesis Strategies for Ultra Fast and Low Latency Matrix Inversion Implementation for Massive MIMO Processing

Samuel Sirois, Messaoud Ahmed Ouameur and Daniel Massicotte

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering,
3351, Boul. des Forges, Trois-Rivières, Québec, Canada
Laboratoire des Signaux et Systèmes Intégrés
Chaire de recherche sur les signaux et l'intelligence des systèmes haute performance

Abstract—**In this paper, we discuss the implementation strategies of an explicit matrix inversion technique based on the recursive Gram matrix inversion update (RGMIU) algorithm. These strategies are explicitly chosen to optimise the design for high throughput dictated by the enhanced mobile broad band (eMBB) use case and by the low latency imposed by the ultra-reliable low-latency communications (URLLCs) use case. The RGMIU algorithm is recently proposed to implement the zero forcing (ZF) problem encountered in massive multiple-input multiple-output (MIMO) detection task. We therefore compare and analyse the performance in terms of symbol error rate (SER) against popular implicit and explicit methods such as optimised coordinate descent (OCD), Gauss-Seidel (GS) and Neumann series expansion (NSE) algorithms. To determine the optimal word length, a fixed-point analysis shows that 16 bits are enough to reach a floating-point precision. We, thereafter, use Vivado High-Level Synthesis tools and optimization directives to implement the RGMIU algorithm based on three strategies to infer key insights and design trade-offs from the resource's utilization, latency, throughput and energy efficiency.**

Index Terms— **Massive MIMO, Recursive Gram Matrix Inversion Update (RGMIU), MIMO detection, Zero Forcing (ZF), Fixed-point analysis, high level synthesis, Vivado HLS, throughput, latency.**

I. Introduction

Being a promising concept for future cellular networks, massive multiple-input multiple-output (MIMO) has now made its way to 5G as one of the means to substantially improve both spectral and energy efficiencies [1]. As a matter of fact, base stations (BSs) with 64 fully digital transceiver chains are commercially deployed and the key component of massive MIMO has made its way into the 5G standard [2], [3]. Nevertheless, the authors in [4] have pointed out that massive MIMO implementation continues to be at least as exciting as massive MIMO theory. Massive MIMO is a form of multiuser MIMO where the number of serving antennas at the BS is an order of magnitude larger than the number of user terminals (UTs) served within each radio resource element. Given the large number of antennas, reliance on time division duplex (TDD) channel reciprocity is essential [1].

Under favorable channel conditions and/or as the number of antennas increases, the UTs' channels are mutually orthogonal which makes linear processing (detection and precoding), such as maximum ratio combining (MRC), zero forcing (ZF) and minimum mean square error (MMSE) detection techniques, optimal [5]. The detection/precoding problem based on ZF or MMSE technique is an arithmetic operation with cubic computational complexity in the order of the matrix dimension. To reduce the implementation complexity, matrix inversion approximations such as Neumann series expansion (NSE) is proposed in [6]. Recently, a technique based on Gauss-Seidel (GS) was shown to outperform NSE due to its fast convergence at considerably low computational complexity [7]. However, this comes at the expense of higher latency and lower throughput [7]. It has actually been shown that the NSE performance degrades as the number of UTs increases [8]. To counter the load increase effect, GS can still afford using more iterations while maintaining lower computational complexity, albeit at the expense of reduced throughput [7]. It has therefore been argued to resort to exact matrix inversion [8].

High throughput application specific integrated circuit (ASIC) is designed for the NSE based detector in [9]. The ASIC achieves 3.8 Gbps for 128 antenna BS and 8 users for single carrier frequency division multiple access (SCFDMA). The NSE based detector is also implemented on a Xilinx Virtex-7 FPGA in [6]. The FPGA design achieves 600 Mbps for 128 antenna BS serving 8 UTs (i.e. 128×8 system). A detector based on the conjugate gradient (CG) algorithm has been proposed in [10] and achieved a significant complexity reduction. However, to speed up the convergence rate and improved performance, a hybrid detector based on the CG algorithm and the Jacobi method has been proposed in [11]. The CG-based detector is also implemented in Xilinx Virtex-7 FPGA for a 128×8 system [12]. On the other hand, the GS-based method has been proposed wherein the initial solution is based on the NSE of two terms [7] whereas, its parallel architecture is implemented in [13]. Even through the GS method can reduce the complexity to be $O(K^2)$ [14], however, due to the GS internal sequential iterations dependencies, it is not well suited for parallel implementation [7], [15].

On the other hand, it has also been argued that these centralized processing techniques still impose stringent constraints on the interconnects' bandwidth between the massive MIMO radio heads (RHs) and the central processing unit (CPU). Distributed, or decentralized, massive MIMO processing has been introduced to overcome such limitations [16] and [17]. Unfortunately, the decentralized processing

computational complexity, and hence the energy efficiency, are also of concern [18]. On the other hand, to support ultra-reliable low-latency communications (URLLCs), low latency and high throughput processing is required. As such, we focus on implementing the core matrix inversion technique based on recursive Gram matrix inversion update (RGMIU) method [19].The inversion of the Gram matrix is performed by exploiting matrix inversion update of a matrix in the form of $\mathbf{H}^H\mathbf{H}$ when a new column is added/updated to a complex-valued matrix $\mathbf{H}$ [19]. Even though implementation of the RGMIU technique has been presented in [19], the contribution of this paper is the way the algorithm is implemented. In fact, the RGMIU method is completely unrolled to leverage a maximum throughput performance. We can add that the implementation in [19] was designed for low resources consumption. Also, the order of magnitude of the obtained results can justify the importance of this paper as it will be shown in the next sections. Herein, direct matrix inversion based on Cholesky decomposition is considered as a reference from the performance and computational complexity standpoint.

A comprehensive review, comparison and discussion of the existing linear precoding mechanisms for massive MIMO according to different cell scenarios have been presented in [8]. It also discussed some standing challenges which related to the design of precoding mechanisms and practical implementations. Low complexity precoders suffered from a considerable performance loss, while a complicated precoder design is more difficult to implement practically. On the other hand, an extensive survey on detection algorithms related to massive MIMO system is presented in [20]. The particular focus is on performance and complexity trade-off as well as the practical implementation of detection algorithms.

In this paper,

- We simulate and compare performances in terms of symbol error rate (SER) for four popular algorithms, namely the Gauss-Seidel (GS) implicit method, the optimized coordinate descent (OCD) implicit method with BOX equalization, the Neumann series expansion (NSE) of order 3 explicit approximation method and the RGMIU explicit method. The goal is to introduce the reader to other state-of-the-art algorithms and to show how the RGMIU performs relatively to these algorithms. This is an extension to [19] by adding the OCD method to make the paper self-contained including fixed point simulations.
- The main focus of this work will be on the RGMIU algorithm implementation which is not considered in [19]. We use Vivado High-Level Synthesis tools and optimization directives to implement the RGMIU algorithm based on three strategies to infer key insights and design trade-offs from the resource's utilization, latency, throughput and energy efficiency.

The paper is organized as follows: Section II summarizes the detection problem. Section III presents the simulation results and discusses the performances of the four methods. Section IV shows the implementation results on Vivado HLS in terms of design strategies, fixed-point analysis, resources consumptions, latency/throughput and energy efficiency for the RGMIU algorithm. Finally, a conclusion is drawn.

II. Signal Model

Considering a massive MIMO system with K UTs and M BS antennas, the received signals at the BS can be modeled as

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{1}$$

where $\mathbf{y} \in \mathbb{C}^{M\times 1}$ is a vector containing the signals received by each BS antenna, $\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \cdots \quad \mathbf{h}_K]$ is the matrix channel for a given sub-carrier with every column representing the channel between every antenna and one particular UT so that $\mathbf{H} \in \mathbb{C}^{M\times K}$, $\mathbf{s} = [s_1 \quad s_2 \quad \cdots \quad s_K]^T$ is the vector containing the K symbols sent by the K UTs and $\mathbf{n} \in \mathbb{C}^{M\times 1}$ is a complex gaussian white noise with zero mean and variance σ^2. The Gram matrix and the matched filter vector resulting from the projection of the vector $\mathbf{y}$ onto the column space of $\mathbf{H}$ are compute as follow

$$\mathbf{G} = \mathbf{H}^H\mathbf{H}, \tag{2.a}$$

$$\mathbf{y}_{MF} = \mathbf{H}^H\mathbf{y}, \tag{2.b}$$

where $\mathbf{G}$ is the Gram matrix and $\mathbf{y}_{MF}$ is the matched filter vector. Therefore, the new square system of equations can be written as

$$\mathbf{y}_{MF} = \mathbf{G}\mathbf{s} + \mathbf{H}^H\mathbf{n}. \tag{3}$$

It is possible to explicitly solve equation (3) as follow

$$\hat{\mathbf{s}} = (\mathbf{G})^{-1}\mathbf{y}_{MF} \tag{4}$$

where $\hat{\mathbf{s}}$ is the vector containing the estimated value of the symbols sent by the UTs. In fact, hard or soft data output decision needs to be done on $\hat{\mathbf{s}}$ depending on if forward error correction process is used or not to ensure that the estimated symbols are contained within the same set (constellation) as the one of $\mathbf{s}$. For simplicity, our analysis will be restricted to $\hat{\mathbf{s}}$ by computing the symbol error rate (SER) on hard output decision which consists of making a decision based on the symbol in the constellation set that is the closest to its estimated value $\hat{\mathbf{s}}$ in terms of Euclidian distances.

It is also possible to implicitly solve it by using iterative methods as discussed in the first section. Nevertheless, the procedure that consists of solving equation (3) is known as a zero forcing (ZF) solution. By adding the variance σ^2 to every diagonal entry of the Gram matrix and solving the same system, we could have derived a more robust technique called minimum mean square error (MMSE) problem but since it is often hard to get the value of σ^2 in practice, we chose simplicity and use ZF algorithm. Finally, hard output decision can be written as

$$\hat{s}_i^{hard} = \arg\min_{z\in O} \left|\hat{s}_i - z\right| \quad i = 1,\ldots,K \tag{5}$$

where O is the constellation set. With this metric in hand, it is possible to process the SER by computing the ratio of the

wrong decisions over the total number of symbols sent. As the main focus is to implement the RGMIU algorithm, Table 1 outline such technique for the sake of safe contained reference where the details are found in [19]. To give the reader a better intuition behind this method, the RGMIU algorithm is said to be recursive in the sense that the inverse of the Gram Matrix is recursively built from a smaller Gram matrix inverse that is growing every iteration. This growing matrix corresponds to $\mathbf{B}$ in table 1. Each iteration of the RGMIU algorithm adds a row and a column to $\mathbf{B}$ up until the full inverse of the Gram matrix is obtained. That way, the starting point of the algorithm could be an already known inverse of a one dimension smaller Gram matrix and only one iteration would be required to find the new inverse. This is typically the case when the channel matrix does not change and a new UT is added to the group. This behavior is opposite to most classical iterative algorithms that need to redo a set of iteration all over again when a new UT is considered.

TABLE 1. THE RECURSIVE GRAM MATRIX INVERSION UPDATE (RGMIU) ALGORITHM

INPUT: $\mathbf{G}=\mathbf{H}^H\mathbf{H}$

Precompute the scalar inverse as a starting point:

$\mathbf{B}_1=\left(\mathbf{G}_{1,1}\right)^{-1}$ // $\mathbf{B}_1$ is an 1×1 matrix (scalar)

FOR $m=1$ **to** $K-1$ **DO:**

1. $\mathbf{z}=\mathbf{G}_{m+1,m+1}$ // $\mathbf{G}_{m+1,m+1}$ is the element at row and column $m+1$
2. $\mathbf{y}_1=\mathbf{G}_{1:m,m+1}$ // $\mathbf{G}_{1:m,m+1}$ corresponds to rows 1 to m in column $m+1$
3. $\mathbf{y}_2=\mathbf{B}_m\mathbf{y}_1$
4. $c=1/\left(\mathbf{z}-\mathbf{y}_1^H\mathbf{y}_2\right)$
5. $\mathbf{y}_3=c\,\mathbf{y}_2$
6. $\mathbf{\Gamma}=\mathbf{B}_m+c\,\mathbf{y}_2\mathbf{y}_2^H$
7. $\mathbf{B}_{m+1}=\begin{bmatrix}\mathbf{\Gamma} & -\mathbf{y}_3\\ -\mathbf{y}_3^H & c\end{bmatrix}$ // $\mathbf{B}$ is a growing matrix

END DO

OUTPUT: $\mathbf{B}_K=\left(\mathbf{H}^H\mathbf{H}\right)^{-1}$

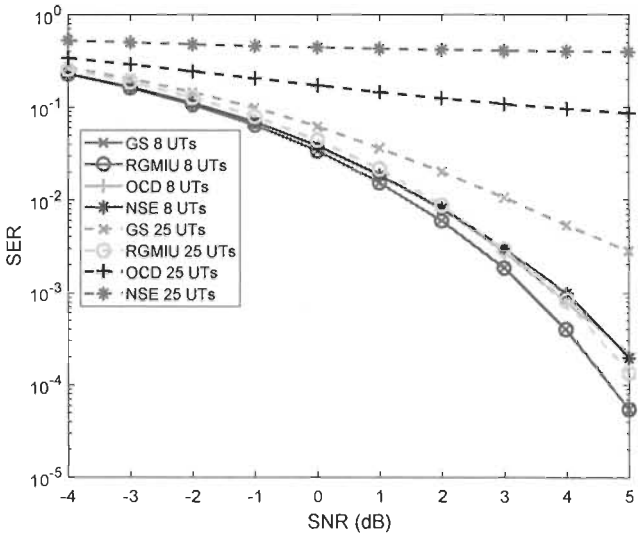
III. SIMULATION RESULT

Our simulations were done with 128 BS antennas and 8 or 25 UTs. The channel matrix is modeled with independent and identically distributed (i.i.d.) random variable with zero mean and unit variance and we used a 64 QAM constellation for the symbols with an average transmitted power of 1.

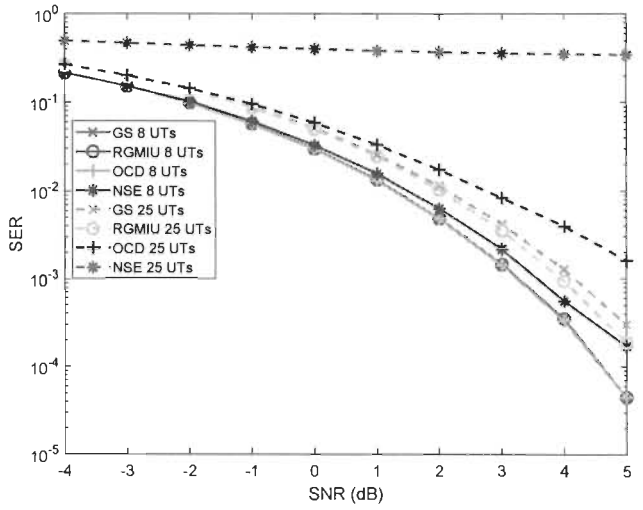
Before, looking at the simulation results, we can predict some behaviors of these algorithms. Firstly, since the GS and the RGMIU are both solving the ZF problem, we can guess that with enough iterations, the GS algorithm will reach the same performances as the RGMIU method. Also, as mentioned in [19], the OCD algorithm has near MMSE performances without having access to the value of σ^2. That being said, we should expect the OCD to have the smallest SER after a certain number of iterations.

Fig. 1 shows the SER for the four algorithms with the GS and OCD having 2 and 3 iterations respectively for 8 and 25 UTs. As expected, for 8 UTs, it takes at least two iterations for the GS algorithm to reach the same performances as the RGMIU algorithm. It also takes three iterations for the OCD method to get the same results as RGMIU and GS. The reason why OCD does not outperform the other algorithms is because the channel matrix $\mathbf{H}$ is not correlated. That way, the diversity at the BS is maximized and the system of equations is easy to solve with a good accuracy. Our goal was simply to get a good idea on how these algorithms perform relatively to one another. On the other hand, the order 3 NSE algorithm never reaches the performances of exact matrix inversion and has a catastrophic SER with 25 UTs.

Of particular interest is the fact that for 25 UTs, OCD and GS need more iterations to reach the performances of the RGMIU. In Fig. 1b, OCD and GS do not reach the SER of RGMIU with three iterations. In general, the number of iterations for implicit methods will depend on the ratio of the number of BS antennas



a)



b)

Fig. 1 SER versus SNR of the GS, NSE, OCD and RGMIU with a) two iterations and b) three iterations for GS and OCD.

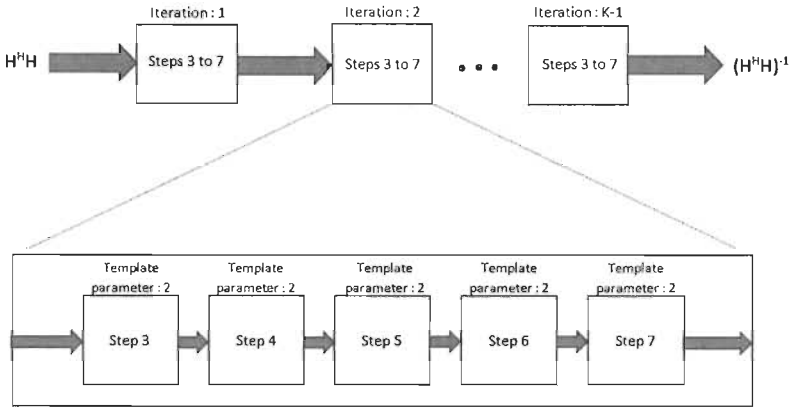


Fig. 2 RGMIU implementation flat architecture

over the number of UTs which is a disadvantage because the complexity of the algorithm is hard to predict. In contrast, the complexity of the RGMIU can be known in advance because it takes as many iterations as the number of UTs minus one to inverse the Gram matrix. Also, the results of the RGMIU will always be accurate because its output is the exact Gram matrix inverse. On the other hand, the order 3 NSE will be accurate only when the ratio of the number BS antennas over the number UTs is high.

In the end, all the presented techniques solve for the ZF detector. Some methods approximate the Gram matrix inverse and others implicitly solve the system of equations. The results tend to show that approximation methods may not solve the ZF problem with good accuracy and that implicit iterative methods performances can vary depending on the number of iterations applied in the process. The over all goal here is to show that direct matrix inversion should be chosen to obtain the best trade-off in terms of performances, complexity and reliability.

From another point of view, the implementation trade-off between algorithms is mainly about the results in terms of throughput versus resources consumptions. A computational intensive algorithm can be implemented with low resources consumption but the throughput results will generally be very low. The opposite can also be true. As a matter of fact, it is shown in [19] that the RGMIU method is in the midfield in terms of computational complexity, being below the NSE technique and above the GS algorithm.

In this sense, RGMIU is partly chosen for its low inter-iteration dependencies which generally reduces resources consumptions regarding the algorithm complexity. In [8], implementations of the CG and the GS can only achieve a throughput of 20 and 48 Mb/s respectively as opposed to the Neumann series expansion technique that can reach a theoretical 621 Mb/s. In all these three algorithms, the preprocessing steps corresponding to compute the Gram matrix is part of the implementation. This shows that the bottleneck of the design in terms of throughput is the core algorithm itself and not the preprocessing steps. On the other hand, the resources consumption on the chosen FPGA for the NSE implementation uses 34% of LUTs, 19% of FFs and 28% of DSP48Es as opposed to the two other methods that barely use above 5% of each of these hardware components.

That said, as it is shown in [7], another GS implementation could reach a throughput of 732 Mb/s with similar resources consumption as the NSE algorithm.

What will be shown in the next sections is that by fully optimizing the RGMIU algorithm to reach excellent throughput performances, the resources consumption for 8 mobile users are more than acceptable (see table 2) with potential throughput results above what is normally expected. The only problem that remains is the implementation of a preprocessing and a post processing unit that will not act as the bottleneck to the design.

IV. Implementation

We used Vivado HLS 2019.1 to implement our designs [21]. All solutions were validated with RTL Cosimulation and we used the xc7vx690t from the Virtex 7 family as the reference FPGA.

A. Implementation strategies

Before directly talking about design strategies, it is important to mention here that we have only implemented the core of the RGMIU algorithm. In other words, the Gram matrix and matched filter vector preprocessing phase is not implemented as well as the post processing multiplication of the matrix inverse with the matched filter vector. We only wanted to optimize the core algorithm because as we will see later, it has the capabilities to reach huge throughput data detection, but the preprocessing and the post processing units cannot keep up with the data rate at which the core can operate without a big increase in resources utilization on FPGA. Our goal is to show the potential of the core algorithm. On the other hand, the complete implementation of the RGMIU algorithm on an ASIC, which can reach higher clock frequencies than FPGA, could be realistic in the sense that the preprocessing and post processing phases could operate at a higher clock frequency than the core algorithm in order to use it at its full capacity.

Three implementation strategies were used [22]. The first one, which we refer as S1, uses a different hardware for every iteration. It is important to note here that the RGMIU algorithm has growing vector and matrix sizes every iteration so that loops boundaries in the code are variable depending on the current iteration. With that in mind, we found out that the best way to translate this to Vivado HLS was to create template

functions in C++ with the template parameter being the loop boundaries so that every time a template function is called with a new template parameter, Vivado HLS understands that it needs to create new hardware for this function call. Every function was written in such a way that it could be pipelined with an initiation interval of one. Without the function template, we would have needed to declare as many functions as there are iterations times the number of steps in one iteration with the only difference being the loop boundaries parameter values. Some may argue that we could have use the FUNCTION_INSTANTIATE pragma instead which is equivalent in some way to the template functions, but we found out that in our design, this directive was interfering with the PIPELINE pragma. We then created a top function which called all the template functions one after the other [22]. This top function was pipelined, and the template functions were inlined to increase the flow of data and to maximize the performances. In total, there were five template functions corresponding to equations (3) to (7) in Table 1 and the value of the template parameter went from 1 to $K-1$. Also, since the loop boundaries are different for every iteration in Table 1, the hardware resources used are also different for every iteration because most of the loops are unrolled by the pipeline directive in the top function. Fig. 2 shows the architecture of the first implementation strategy.

The second strategy (S2) is basically the same as the first strategy except that the design is heavily pipelined with more registers so that the clock signal can reach higher frequencies which in turn enables a higher potential throughput. The downsides of this are the increases in resources consumption and a higher latency. To translate this to Vivado HLS, we just reused the first strategy solution and we set a clock constrain with a smaller period. Even if it is straightforward to pass from S1 to S2, the authors judge that the results are still worth mentioning since the maximum clock frequency can be significantly increased. Otherwise, the general architecture of S2 is the same as the one of S1 depicted in Fig. 2.

The third strategy (S3) is designed in such a way that resources consumption is similar in terms of percentage for the DSP48E, the flip-flops (FF) and the look-up tables (LUT). The downside of this method, as we will see in the Section IV.D, is that the throughput is generally cut in half. To deploy this strategy, we used the ALLOCATION pragma to limit the number of DSP48E in the top function, so that FFs and LUTs are needed to compensate this limitation. The fact that S3 uses less DSP48E can give more room to other implementations that need explicitly this type of resources to reach good performances in term of throughput on the same FPGA. That is not the case with S1 and S2. Also, S3 is the only possible implementation with 12 users. Once again, the general architecture of S3 is the same as the one of S1 depicted in Fig. 2.

To sum up in more details, the beauty of the RGMIU algorithm is that it does not contain major sequential inter-iteration dependencies, as opposed, for example, to the GS method [7]. This allows to implement this method with a small pipeline initiation interval without excessive usage of hardware resources. The smaller the initiation interval, the higher the throughput is, since an initiation interval of one means a new valid input can be processed every new clock cycle. Also, as stated before, the number of iterations that the algorithm needs to go through to obtain the exact inverse of the Gram matrix is equal to the number of mobile users minus one. These simple facts greatly simplify the complexity of the hardware implementation under Vivado HLS.

As stated before, equations (3) to (7) corresponding to one iteration in table 1 can each be described as a simple function in C++ with a template parameter used to indicate the current iteration. Once this is done, each array that is contained in these functions are partitioned with the ARRAY_PARTITION pragma. Then, the whole function is pipelined with the PIPELINE pragma and inlined with the INLINE pragma. The combination of partitioning and pipelining allows Vivado HLS to effectively pipeline every loop inside a given function to maximize its throughput while inlining prevents the creation of bottle necks between each function call. After that, a main top function is created to call all functions corresponding to one iteration of the RGMIU algorithm a number of times equal to the number of mobile users minus one. This main function is also pipelined to ensure smooth data transfers between function calls. In other words, the main idea of the RGMIU method implementation on Vivado HLS was to first sub-divide the algorithm in simple elements and optimized them as much as possible to then glue them together in an efficient way to avoid bottle necks. At this stage the only thing missing is the input/output management. Basically, the input of the design is a FIFO corresponding to every element of the Gram matrix in parallel and the output is corresponding, in a similar manner, to every element of the inverse of the GRAM matrix in parallel. The INTERFACE pragma was used to achieve this.

Nothing fancy was needed to obtain our implementation results. The key was to correctly sub-divise the algorithm and add simple optimization pragmas from the bottom-up. This is where the beauty of Vivado HLS resides. It can be a powerful tool for one who knows how to use it. At the risk of repeating ourselves, the implementation results are the main contributing factors of this paper, even if the path to obtain them remains quite trivial.

The optimization strategies were also quite simple to implement. The first strategy was simply obtained by letting Vivado HLS freely set the clock constrains and the resources allocation. Then, from this first baseline strategy, the second strategy design was obtained by specifying a smaller target clock in the project solution settings. Finally, the third strategy was derived from the first strategy by using the ALLOCATION pragma in the main top function to limit the number of DSP48Es to a maximum value.

That being said, we also tried to implement a fourth strategy which consisted of reusing the same hardware for the last two iterations. The goal was to dramatically reduce the resources consumption since the DSP48E utilization is a cubic function with respect to the number of iterations as we will see in the Section IV.E. When we implemented it, there was indeed a big improvement in the resources consumption, but the throughput was getting too slow so that we have decided to abandon the idea.

TABLE 2. ESTIMATED RESOURCES, LATENCY AND THROUGHPUT

Strategy	Minimum clock period (ns)	Number of UTs	Estimated latency (µs)	Estimated Throughput (Gb/s)	DSP48Es	LUTs	FFs
S1	2.64 ± 0.38	4	0.428	9.09	130 (3%)	12192 (2%)	18065 (2%)
		8	0.966	18.18	1218 (33%)	60216 (13%)	95177 (10%)
S2	1.934 ± 0.13	4	0.404	**12.41**	130 (3%)	14726 (3%)	32829 (3%)
		8	1.039	**24.82**	1218 (33%)	84282 (19%)	210569 (24%)
S3	2.577 ± 0.38	8	0.905	9.31	750 (20%)	84098 (19%)	161753 (18%)
		12	1.495	13.97	2400 (66%)	289930 (66%)	534689 (61%)

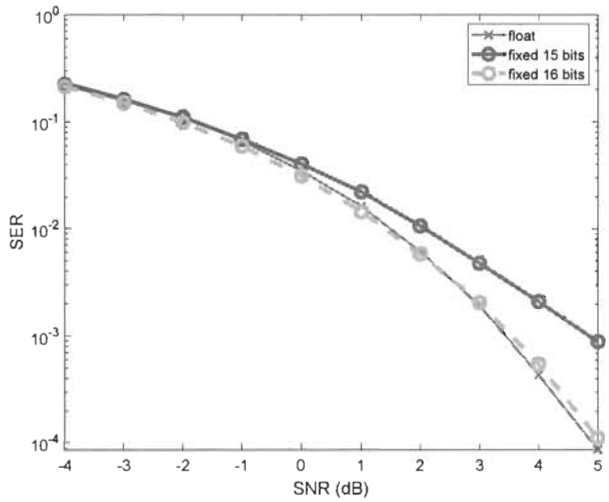


Fig. 3 SER of the RGMIU algorithm for 8 UTs with floating and fixed-point precision.

Before continuing to the next sub-section, one thing we did not try but that could have been interesting is a pipeline interleaving strategy as they did in [8]. Indeed, since the algorithm loop depicted in Table 1 has inter-iteration dependencies, this prevents traditional pipelining except if we completely unroll it (different hardware for every iteration) as we did in S1, S2 and S3. To overcome this problem without a complete unroll, it would have been possible to always reuse the same hardware that represents one iteration and to create several pipeline stages inside it so that the core can accept a new Gram matrix input every clock cycle up until all the pipeline stages are filled. Thereafter, the algorithm would need to finish its $K-1$ iterations before being able to accept a new batch of inputs. Also, it would have been possible to create a hardware that computes more than one iteration to be able to have more pipeline stages in the design and a higher global throughput at the expense of higher resources utilization. This pipeline interleaving strategy is reserved for future work.

B. *Fixed-point analysis*

The fixed-point analysis was done with 16 bits word lengths. Most of the internal variables used an integer part of one signed bit and 15 fractional bits. Fortunately, no shifts were needed to reduce the dynamic range and the common operators in Vivado HLS were able to automatically align the binary point of two variables having different fractional bit width. That being said, since DSP48Es in FPGAs can accept a maximum bit width of 18, our design was not panelized in terms of resources consumption. For all complex value multiplications, the complex multiplier is composed of 4 real multipliers and 2 real adders.

We have been able to reach a good precision such that the difference between the SER curve of floating and fixed-point never exceed 0.5 dB. Indeed, Fig. 3 shows the comparison in terms of SER for floating and fixed-points variables with a bit width of 15 and 16. Clearly, 16 bits is the minimum required bit width to obtain results below 0.5 dB.

C. *Resources utilization estimates*

The resources utilization for all strategies in terms of DSP48Es, FFs and LUTs is depicted in Table 2 for different numbers of UTs. Looking only at S1 and S2, the number of DSP48Es used is independent of the employed strategy. This is totally normal because these are used for multiplication and multiplier-accumulate (MAC) operations which are independent of the level of pipelining. The DSP48Es are only dependent on the number of UTs. This being said, we can derive a formula that connects these two parameters together. By looking at the scheduler of Vivado HLS, we have determined that the required number of DSP48Es, namely $d(l)$, for the l^{th} iteration is defined as follow

$$d(l) = 8l^2 + 4l - 2 \tag{6}$$

With that in mind, the total number of DSP48Es, namely D, is defined by

$$D = \sum_{l=1}^{L} d(l) = \frac{16L^3 + 36L^2 + 8L}{6} \tag{7}$$

where L is the total number of iterations, which corresponds to the number of UTs minus one. Equation (7) shows that the number DSP48Es follows a cubic order which can be a limiting factor for bigger designs with more than 8 UTs. This is why we have implemented S3 to repartition the resources in such a way that we are not limited by the number of DSP48Es for 12 UTs. The resources consumptions of S3 in terms of LUTs and FFs is way higher than S1, but it is much less in terms of DSP48Es. This comes from the fact that Vivado HLS used LUTs and FFs to create the same logic response as DSP48Es. As it will be

seen in the next sub-section, the throughput in S3 is smaller because DSP48Es are designed and optimized to accept a high clock frequency. When the algorithm can use all the DSP48E it needs, the overall throughput can be maximized in contrast as when it cannot. The obtained results of S3 are worth mentioning to show the flexibility of the implementation. On the other hand, as we could expect, S2 utilizes more FFs and LUTs than S1 because it has more pipeline stages. Finally, as mentioned in the implementation strategies Section IV.E, S3 uses resources in a balanced way. In addition to allowing us to implement a design with 12 UTs, it significantly reduces the number of DSP48Es for 8 UTs by a factor of 13%.

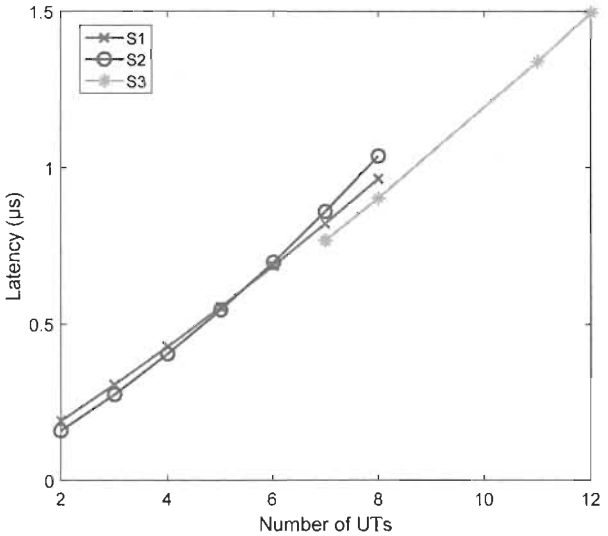


Fig. 4 Latency versus the number of UTs for implementation strategies S1, S2 and S3.

D. *Latency and throughput estimates*

Referring to the results obtained from Vivado HLS post synthesis estimations in Table 2, the latencies between S1 and S2 are similar but the highest throughputs are achieved by S2. Fig. 4 compares the latencies and throughputs for S1, S2 and S3 in a 64 QAM modulation. For 8 UTs, we are talking about a potential latency of 0.966 µs and a potential throughput of 18.18 Gb/s with S1. These number are conditional to the fact that the preprocessing and the post processing units can keep up with this data rate. In a similar fashion, for 8 UTs with S2, it is theoretically possible to reach a latency of 1.04 µs and a throughput of 24.82 Gb/s. It is interesting to note that since the implementations of all strategies are pipelined with a unitary input interval, the minimum clock period estimated by Vivado HLS in Table 2 also corresponds to the Gram matrix inversion throughput period. That way, the estimated throughput in Gb/s can be calculated by multiplying the number of UTs with the number of bits in the constellation (6 bits for 64 QAM) and by dividing the result with the estimated clock period. To continue, both throughput and latency are linear with the number of UTs. This is totally normal for the throughput because the number of UTs is the only variable multiplicative factor. Since the algorithm is completely unrolled (All loops flatten), the limiting factor for the clock frequency is the level of pipelining which is independent of the number of users. The only thing that changes is the resources consumption and the over all latency since 7 iterations are flatten with 8 UTs in contrast of only 3 with 4 UTs. On the other hand, by looking at the design scheduler in Vivado HLS with S1 for example, we concluded that the latency was approximately linear. Indeed, since there is one division of 26 cycles for the initialization of B and that every new iteration there is a new division of 26 cycles (equation (4) in Table 1) and parallel operations that take a latency of approximately 20 cycles for the first few iterations and that grows very slowly, the sum of all these terms gives a rough linear function. Fig. 4 also shows the latency for S3. The resources trade-off had the downside effect of approximately cutting in half the throughput if we compare it with S1. In terms of latencies, S1, S2 and S3 are similar with S3 taking a little less clock cycles. Of particular interest is the case when a new UT is added or when an existing UT has a new channel. Indeed, if a new UT is included, the starting point of the algorithm is the $K \times K$ already computed matrix inverse. In other words, every time a new UT is added, the algorithm only needs to do one iteration before finding the new inverse. Also, as shown in [19], when the channel changes for one UT, only a partial computation equivalent in terms of latency of two times (removing the UT and then adding it with its new CSI) the last iteration of the algorithm needs to be done. Other algorithms need to redo all the calculations to get to the new inverse. Table 3 shows the approximative latency for different configurations for updating the matrix inverse when one UT has a new channel. This approximation is computed by multiplying by 2 the result obtained by the difference between the total latency of two design with $n-1$ and n UTs, where n is arbitrary. The obtained latency gains are not neglectable and can represent a big advantage for time critical applications.

TABLE 3. ESTIMATED LATENCY TO UPDATE ONE UT

Strategy	Number of UTs	Estimated latency to replace one UT (µs)
S1	4	0.243
	6	0.264
	8	**0.285**
S2	4	**0.259**
	6	0.306
	8	**0.352**
S3	8	**0.201**
	12	0.236

TABLE 4. ENERGY EFFICIENCY VS OPTIMIZATION STRATEGY

Strategy	Number of UTs	Power consumption estimates (W)	Energy efficiency estimates (Gb/J)
S1	4	1.16	7.837
	6	2.541	5.367
	8	5.03	**3.615**
S2	4	2.068	6.001
	6	5.306	3.508
	8	11.34	**2.189**
S3	8	6.291	**1.480**
	12	16.809	0.831

E. Energy efficiency estimates

We have been able to compute an estimate of the power consumption of our designs with the help of the Xilinx Power Estimator spreadsheet. Post place and route timing simulation to estimate energy efficiency would be worth in a complete end to end implementation case. The goal here is just to give a rough idea of the power estimates of our design. The main contribution of the paper is still the throughput results obtained with the RGMIU algorithm. Table 4 presents the energy efficiency for our three strategies and for different numbers of UTs. This efficiency is shown in terms of watts (W) and Gigabits per Joule (Gb/J). The first thing to notice is that the energy efficiency is decreasing with the number of UTs no matter which strategy is used. Clearly, since S2 is heavily pipelined, the design clock can reach a higher frequency so that the power consumption is heavily increased. From S1 to S2, for 8 UTs in a 64 QAM modulation, we pass from approximately 3.615 Gb/J to 2.189 Gb/J so that the trade-off for S2 is a higher throughput for a lower energy efficiency and a higher resources consumption. On the other hand, since S3 needs to compensate the fixed limit of DSP48Es with a lot of LUTs and FFs, the power consumption is higher compared to S1. Indeed, we get a 1.480 Gb/J efficiency for 8 UTs in a 64 QAM modulation. The trade-off then becomes a sacrifice of throughput and energy consumption for a more balanced resources utilization. It is important to note here that the efficiency is computed for the core algorithm only. By adding the preprocessing and the post processing units, the energy efficiency would be a little bit less than previously computed.

V. Conclusion

To conclude, we have compared in simulation the performances of four different types of algorithm detection in terms of error symbol rate (SER). We have shown that the performances of iterative algorithms are sensitive to the number of iterations chosen but when it is carefully chosen, the GS algorithm has the same SER performances as the RGMIU method and the OCD algorithm has the potential to get better results due to the fact that it can reach near MMSE performances. On the other hand, NSE algorithm has very poor results when the number of UTs increase. Nevertheless, we have done these simulations to support the point that direct matrix inversion always leads to predictable and accurate results.

Also, we have implemented the RGMIU algorithm on Vivado HLS and have shown the big potential of this method to reach high throughput data detection in a massive MIMO system. The design was done on 16 bits words and performances for three different designs in terms of resources consumption, latency/throughput and energy efficiency were compared to bring out the possible trade-offs.

Unfortunately, it would not be fair to directly compare performances of our implementations with others known in the literature simply because we assume the existence of a preprocessing and a post processing unit that can keep up with the core algorithm data rate. Just by looking at the results, throughputs in the order of magnitudes of Gb/s are obtained as opposed to Mb/s in the literature. Resources consumption is also difficult to directly compare since preprocessing and post processing are not considered in our design.

A more in-depth study needs to be done to accelerate the pre and post processing steps by either approximating the Gram matrix or simply getting around these units.

Our future work will also consist of implementing the pipeline interleaving strategy for RGMIU as well as the GS, OCD and NSE algorithms to present a full comparison between all these methods.

Acknowledgments

This work has been funded by the Natural Sciences and Engineering Research Council of Canada, Prompt, Canadian Foundation for Innovation, the CMC Microsystems and NUTAQ innovation.

References

[1] E. Björnson, L. Sanguinetti, H. Wymeersch, J. Hoydis, and T.L. Marzetta, "Massive MIMO is a Reality – What is Next?: Five Promising Research Directions for Antenna Arrays," *Digital Signal Processing*, vol. 94, 2019, pp. 3–20.

[2] E. Björnson, "A look at an LTE-TDD Massive MIMO product," http://ma-mimo.ellintech.se/2018/08/27/ Accessed 7 July 2020.

[3] P. von Butovitsch, D. Astely, C. Friberg, A. Furuskär, B. Göransson, B. Hogan, J. Karlsson, and E. Larsson, "Advanced antenna systems for 5G networks. Ericsson white paper," https://www.ericsson.com/4a8a87/assets/local/publications/white-papers/10201407_wp_advanced_antenna_system_nov18_181115.pdf (Accessed 7 July 2020

[4] Y. Qu, A. Lozano, and A. Gatherer, "Nine Communications Technology Trends for 2019," Communication society technology news, 2019. https://www.comsoc.org/publications/ctn/nine-communications-technology-trends-2019 (Accessed 7 July 2020)

[5] H.Q. Ngo, "*Massive MIMO: fundamentals and system designs*," Ph.D. Thesis. Linköping University Electronic Press, 2015.

[6] M. Wu, B. Yin, G. Wang, C. Dick, J.R. Cavallaro, and C. Studer, "Large-scale MIMO detection for 3GPP LTE: algorithms and FPGA implementations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, 2014, pp. 916–929.

[7] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X. You, "Efficient architecture for soft-output massive MIMO detection with Gauss-Seidel method," IEEE Int. Symp. on Circuits and Systems, May 2016, pp. 1886–1889.

[8] M. Wu, C. Dick, J.R. Cavallaro, and C. Studer, "High-throughput data detection for massive MU-MIMO-OFDM using coordinate descent," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, 2016, pp. 2357–2367.

[9] B. Yin, M. Wu, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "A 3.8Gb/s large-scale MIMO detector for 3GPP LTE-advanced," Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc., May 2014, pp. 3879–3883.

[10] J. Zhou, Y. Ye, and J. Hu, "Biased MMSE soft-output detection based on Jacobi method in massive MIMO," Proc. IEEE Int. Conference on communication problem-solving, Dec 2014, pp. 442–445.

[11] W. Song, X. Chen, L. Wang, and X. Lu, "Joint conjugate gradient and Jacobi iteration based low complexity precoding for massive MIMO systems," IEEE International Conference on Communication, July 2016, pp. 1–5.

[12] B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "VLSI design of largescale soft-output MIMO detection using conjugate

gradients," IEEE International Symposium on Circuits and Systems, May 2015, pp. 1498–1501.
[13] Z. Wu, Y. Xue, X. You, and C. Zhang, "Hardware efficient detection for massive MIMO uplink with parallel Gauss-Seidel method," International Conference on Digital Signal Processing, August 2017, pp. 1–5.
[14] Z. Zhang, J. Wu, X. Ma, Y. Dong, Y. Wang, S. Chen, and X. Dai, "Reviews of recent progress on low-complexity linear detection via iterative algorithms for massive MIMO systems," IEEE International Conference on Communication, July 2016, pp. 1–6.
[15] X. Qin, Z. Yan, and G. He, "A near-optimal detection scheme based on joint steepest descent and Jacobi method for uplink massive MIMO systems," IEEE Communication Letter, vol. 20, no. 2, Feb. 2016, pp. 276–279.
[16] C. Jeon, K. Li, J.R Cavallaro and C. Studer, "Decentralized Equalization with Feedforward Architectures for Massive MU-MIMO," *IEEE Transactions on Signal Processing*, vol. 67, no. 17, 2019, pp. 4418–4432.
[17] M. Ahmed Ouameur, and D. Massicotte, "Efficient Distributed Processing for Large Scale MIMO Detection," *European Signal Processing Conference (Eusipco)*, A Coruna, Spain, 2-6 Sept. 2019, pp. 1–4.
[18] M. Ahmed Ouameur and D. Massicotte, "Deep Autoencoder for Interconnect's Bandwidth Relaxation in Large Scale MIMO-OFDM Processing", 2019, https://arxiv.org/abs/1907.12613. (Accessed 7 July 2020)
[19] M. Ahmed Ouameur, D. Massicotte, A. M. Akhtar and R. Girald, "Performance Evaluation and Implementation Complexity Analysis Framework for ZF Based Linear Massive MIMO Detection," Wireless Networks Journals, Springer, pp. 1–15, April 2020.
[20] M. A. Albreem, M. Juntti and S. Shahabuddin, "Massive MIMO Detection Techniques: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, 2019, pp. 3109-3132.
[21] Xilinx, Vivado Design Suite User Guide, High-Level Synthesis UG902 (v2019.1) July 12, 2019.
[22] Xilinx, SDx Pragma Reference Guide UG1253 (v2019.1) June 5, 2019.