N° D'ORDRE : 7815

UNIVERSITE PARIS XI
UFR SCIENTIFIQUE D'ORSAY

THESE
Prsentée pour obtenir

Le GRADE de DOCTEUR EN SCIENCES
DE L'UNIVERSITE PARIS XI ORSAY

par Antonín Heřmánek

Sujet: Etude de l'implantation d'algorithmes d'égalisation de prochaine génération

Soutenue le 3.1.2005 devant la Commission d'examen

Mme Sylvie Marcos	Président du jury
Mme Inbar Fijalkow	Rapporteur
M Phillip A. Regalia	Directeur de theèse
M Michel Terré	Rapporteur
M Boris Šimák	Examinateur

Supervisors:

prof. Phillip A. Regalia
Department of Communication, Image and Information processing
National Institute of Telecommunication
Evry, France

Doc. Ing. Boris Šimák
Department of Telecommunication
Czech Technical University in Prague, FEE
Prague, Czech Republic

Acknowledgement

The presented work is a result of my participation on these three institutions: Czech Technical University in Prague, Institute National de Télécomunication in Evry and Institute of Information Theory and Automation of the Czech Academy of Science.

I would like to express my gratitude to all the people I have met during my research. I am truly indebted to Professor Phillip A. Regalia of the Institut National de Télécommunication for taking me under his wing. The majority of the work in this dissertation would not exist without his expertise and general willingness to help. I would like to thanks to Professor Boris Šimák of the Department of Telecommunication of the Czech Technical University for many constructive comments as well as for overall encouragement. The acknowledgement would be far from complete without heartfelt thanks to Jiří Kadlec of the Institute of Information Theory and Automation for his sage advices and the overall long-term support.

The cooperation between the institutions would not be possible without the financial contribution of the French Government which partially supported my studies in France.

Thanks to Jan Schier for editorial help with this manuscript and for his introduction to the belgian beers. The thesis would hardly be possible without the friendly environment in my "home" Department of Signal Processing. Thank you, guys, for all your jokes!

Last but not least, I am deeply indepted to all my family for their incredible support and encouragement during the course of work on this thesis. From the starting line to the kickoff, my family never failed to show.

Abstract

The thesis deals with blind channel equalization using higher-order statistics (HOS). The task of equalization is to remove distortion, which occurs inevitably while the signal propagates from transmitter to receiver, due to multi-path character, attenuation and other adverse effects of the propagation environment. The equalizer is a FIR (Finite Impulse Response) filter, which is – in the case of blind equalization – initialized using an optimization algorithm based on the statistical properties of the signal.

The thesis consists of two main parts: in the first one, the HOS-based equalization algorithms are reviewed, with main focus on the Constant Modulus Algorithms (CMA). Gradient, Algebraic, and Finite-Interval CMA algorithms are discussed in detail and their properties in typical equalization scenarios are investigated. Main contribution of this part is derivation of a Sliding Window FI-CMA algorithm, including detailed investigation of its convergence and numerical properties. The use of the sliding window determines important advantages of this algorithm: ability to track changing environment and lower computational complexity than the standard FI-CMA algorithm (which does not work with the short window used for SW-CMA).

In the second part of the thesis, embedded implementation of the algorithms using Texas Instruments DSP and Xilinx Virtex-series FPGA is investigated. At the beginning of this part, the basic features of both architectures are reviewed and the reader is introduced into the principles of the logarithmic numbering system (LNS), which has been selected to implement the floating point operations, required in the FI-CMA algorithm, in FPGA.

Next, the implementation issues (memory access, limited chip area) are considered. An implementation architecture for the FI-CMA algorithm, optimized for the Virtex-E device, has been designed. Up to our knowledge, an attempt to implement a signal processing algorithm of this complexity in FPGA is rather unique.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbf{a}$	vector $\mathbf{a}$
$\mathbf{e}_\nu$	unit vector with a one a the the ν^{th} position
$\mathbf{A}$	matrix $\mathbf{A}$
$\mathbf{a}^T,\ \mathbf{A}^T$	transpose of vector $\mathbf{a}$ resp. matrix $\mathbf{A}$
$\mathbf{a}^H,\ \mathbf{A}^H$	Hermitian transpose of vector $\mathbf{a}$ resp. matrix $\mathbf{A}$
$\lvert a\rvert$	absolute value of real variable a; modulus of complex variable a
$\lVert\mathbf{a}\rVert$	2-norm of vector $\mathbf{a}$
$\lVert\mathbf{a}\rVert_p$	p-norm of vector $\mathbf{a}$
$\mathbf{a}^{\odot p}$	componentwise p^{th} power of vector $\mathbf{a}$ (also known as Hadamerd power)
$\mathbf{A}^{-1}$	inverse of matrix $\mathbf{A}$
I	identity matrix
$\mathcal{R}\{s\}$	real part of complex number s
$\mathbf{a} \star \mathbf{b}$	convolution of a_i and b_i
$E\{a\}$	expectation of random variable a
$\mathrm{cum}_p\, a$	p^{th} order cumulant of random variable a
σ_a^2	variance of random variable a
m_a	mean value of random variable a

Abbreviation

ACMA	Algebraic CMA
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
BPSK	Binary Phase Shift Keying
CM	Constant Modulus
CMA	Constant Modulus Algorithm
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Devices
CPU	Central Processing Unit
DSP	Digital Signal Processor
FI-CMA	Finite Interval CMA
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
G-CMA	Gradient CMA
GSM	Global System for Mobile communication
ICA	Independent Component Analysis
IP	Intellectual Property
ISI	Intersymbol Interference

LB	Logic Block
LMS	Least Mean Square
LNS	Logarithmic Numbering System
LUT	Look-up Table
MAC	Multiply-and-Accumulate
MAI	Multi-Access Interference
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
MSB	Most Significant Bit
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
PE	Perfect Equalization
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RLS	Recursive Least Square
SIMO	Single Input Multiple Output
SISO	Single Input Single Output
VLIW	Very Long Instruction Word
VLSI	Very Large System Integrations
SNR	Signal to Noise Ration
SRAM	Static RAM
SVD	Singular Value Decomposition
SW-CMA	Sliding Window CMA
ZF	Zero Forcing

Résumé

1 Introduction

Dans les années récentes, nous avons été témoins d'un développement rapide des systèmes de télécommunication, notamment des systèmes sans fils qui se sont répandus très rapidement. Main dans la main avec leurs nombreuses applications vient une recherche intense dans beaucoup de domaines de l'égalisation des canaux, de la séparation de source, de la synchronisation, dans les antennes intelligentes et dans le contrôle de la puissance, pour citer quelques exemples. Le but de cette recherche est développer la performance des systèmes numériques, plus précisément, d'améliorer leur capacité de transmission et de réduire leur consommation d'énergie.

Dans cette thèse, nous nous concentrons sur l'utilisation des statistiques d'ordres supérieurs pour l'égalisation des canaux. La tâche d'équalisation consiste à supprimer la distorsion inhérente propre du canal entre l'émetteur et le récepteur. Cette distorsion est inévitable à cause des propriétés physiques du média de transmission (atténuation, réflexions et glissement de phase à cause d'obstacles le long du canal, caractère multi-chemin de l'environnement et atténuation propre à l'environnement ; le caractère de la distorsion dépend de l'environnement). Le résultat est une superposition de copies du même signal avec des gains et phases différents. Cette superposition donne lieu à une interférence entre symboles transmis, appelées ISI (Inter Symbol Interference). Le degré de distorsion dépend des propriétés de l'environnement : dans une environnement vallonné le délai entre les copies du signal peut atteindre $15\mu s$, alors que dans un environnement urbain seulement $10\mu s$. Pour illustration, dans le

système GSM qui utilise des périodes de symbole de $3.7\mu s$ (c'est-à-dire du 270kbit/s), cela représente un délai de 5 symboles.

Pour empêcher une dégradation sévère de la performance, il est nécessaire de compenser la distorsion des canaux. Cette compensation est effectuée par un filtre du côté récepteur, dans ce qu'on appelle l'égaliseur de canal. Dans le cas où les propriétés du canal sont connues à l'avance, il est possible d'initialiser le filtre avant que la transmission commence. Si le canal n'est pas maîtrisé, les paramètres de l'égaliseur doivent être estimés. Les méthodes standard utilisent ce qu'on appelle des séquences d'apprentissage : l'émetteur émet une séquence de symboles connue par le récepteur. Ce procédé est appelé l'apprentissage. Après l'estimation des paramètres, l'égaliseur est construit et l'égalisation du signal peut commencer.

Dans le cas des systèmes sans fil, les paramètres du canal peuvent varier dans le temps. L'estimation du canal doit alors être répétée avec chaque bloc de données, ce qui résulte, bien entendu, en une perte de la performance de la transmission. Dans le système GSM, pour donner un exemple, la séquence d'apprentissage est constituée de 26 symboles placés au centre de chaque bloc de données de 148 symboles. La perte de performance de la transmission est donc de 18%.

Cette thèse a la structure suivante : dans le reste de l'introduction, nous définissons le système modèle et le concept de l'égalisation. A la fin de ce chapitre, les statistiques d'ordre supérieur (HOS – High Order Statistics) sont brievement rappelées. Dans le Chapitre 2 est présenté l'état delait en la matière de la technique de l'égalisation aveugle. Les quatre techniques les plus importantes sont décrites dans des sections séparées. A la fin du chapitre est effectuée leur comparaison sous l'hypothèse de conditions 'idéales'.

Dans le Chapitre 3, le critère du module constant (CM – Constant Modulus) est discuté plus en détail. Trois algorithmes d'optimisation basés sur la minimisation du CM de la fonction de coût sont décuts et leur comportement typique est montré sur des simulations.

Dans le Chapitre 4 est déduite la fenêtre glissante CMA et le comportement typique est montré sur une simulation. Dans le Chapitre 5 la performance de différents algorithmes basés sur le module constant est comparé pour des canaux à paramètres variables dans le temps. Plusieurs scénarios typiques pour l'évolution des canaux sont présentés. Ces deux chapitres représentent l'essentiel de la thèse pour ce qui est de la partie théorique.

Finalement, les Chapitres 6 et 7 sont orientés vers les problèmes d'implémentation des algorithmes de traitement de signal. Le Chapitre 6 rappel brièvement l'architecture des plates-formes (Xilinx FPGA et Texas Instruments DSP) sur lesquelles l'implémentation d'algorithmes sélectionnés a été effectuée et également l'arithmétique de ces plates-formes. Comme les implémentation FPGA utilisent le système de numérotation logarithmique, qui n'est pas communément connu, il est décrit plus en détail. Le Chapitre 7 présente les résultats de l'implémentation de algorithme du 'Gradient CMA' et des algorithmes à Intervalle Fini CMA ('Finite Interval CMA algorithms'). Ce chapitre représente la contribution essentielle de cette thèse aux problèmes de l'implémentation.

Système modèle

Nous supposons que le système de communication est décrit comme suit : Les symboles à transmettre sont d'abord numériquement modulés et envoyés au récepteur à travers le canal de communication. Les signal modulé est à bande étroite et pour simplifier les expressions nous utilisons la représentation en bande de base. Vu que cette thèse se concentre sur l'algorithme du module constant, les plus importantes modulations sont celles à enveloppe constante : PSK, QPSK, etc. Pour simplifier, nous utilisons que les modulations sans mémoire.

La Matrice du canal

Nous supposons que la réponse impulsionnelle est de durée finie (FIR – Finit Impulsion Response) et à valeurs réelles. La réponse impulsionnelle $h(z)$ est a priori

inconnue. Le bruit blanc Gaussien $w(t)$ est ajouté à la sortie du canal.

La structure de la matrice du canal et du signal reçu dépend du nombre d'entrées et de sorties du canal. Dans cette section, quelques cas typiques (entrée simple/multiple, sortie simple/multiple) vont être décrits. Le modèle à entrée simple représente les systèmes avec un utilisateur émetteur seulement, alors que le module à entrées multiples représente une configuration avec plusieurs utilisateurs du canal ou plusieurs émetteurs interférents. Le modèle à sortie simple est utilisé pour construire le signal de sortie du canal pour des récepteurs à espacement entier (baud-spaced) avec une antenne unique alors que le modèle à sortes multiples pour des systèmes à antennes multiples ou pour des systèmes à espacement rationnel (fractionally spaced systems).

Le concept d'égalisation

Le signal des données, considéré comme aléatoire, est transmis par un canal inconnu. Comme résultat d'une convolution effectuée par ce canal et du caractère multi-utilisateurs de l'environnement de transmission, le signal reçu est détérioré non seulement par le bruit additif $w(t)$ mais aussi par l'interférence inter-symbole (ISI) et par l'interférence multi-accès (MAI).

Le rôle d'un égaliseur est d'enlever cette distorsion pour baisser le pourcentage de symboles erronés. Dans le cas d'un égaliseur idéal la sortie serait égale à la copie retardée de la séquence de symboles à l'entrée. Souvent (dans notre cas également) l'égaliseur a une structure d'un estimateur linéaire à ordre fini, décrit par un vecteur de coefficients d'équalisation $\mathbf{f}$. Ce vecteur est choisi parmi d'un ensemble de vecteurs acceptables (non triviaux) et ceci pour minimiser une certaine fonction de coût. Pour une utilisation future nous définissons les termes de base suivants. L'espace d'équaliseur, l'espace du système la solution à zéro forcé (ZF) et la solution à erreur carré moyenne minimale (MMSE).
Les égalisations à espacement entier et à espacement rationnel sont revu l'existence de ZF et de MMSE est montrée pour les deux cas : bruité ou non.

Les Statistiques d'ordres supérieurs

Dans beaucoup de cas, les propriétés statistiques du second ordre ne suffisent pas pour une bonne déconvolution aveugle. Il existe un groupe d'algorithmes d'ordres supérieurs pour résoudre ce système. Dans le Chapitre 2.3. Nous présentons un algorithme développé par Shalvi et Weinstein qui utilise la maximisation du cumulant d'ordre supérieur (kurtosis) pour résoudre l'équation. Dans ce chapitre est effectué un court rappel des statistiques d'ordres supérieurs et la définition des cumulants est redonnée. Egalement sont discutées les propriétés des cumulants pour les signaux qui sont une combinaison de séries aléatoires identiquement distribuées (comme la sortie d'un canal sans bruit).

2 L'égalisation aveugle

Traditionnellement, les systèmes de communication utilisent des égaliseurs adaptatifs avec des séquences d'apprentissage. Par séquence d'apprentissage, nous voulons parler d'une séquence de symboles connue à l'avance par le récepteur. Cette séquence d'apprentissage est construite d'une façon à ce quelle soit facilement identifiable et elle a la forme d'un burst de synchronisation. L'égaliseur avec une séquence d'apprentissage fonctionne en deux modes : Le mode apprentissage quand les coefficients d'égalisation sont mis à jour en utilisant la connaissance de la séquence et le mode égalisation quand le signal est filtré et envoyé au module de décision. Il est évident que l'envoi de la séquence d'apprentissage décroît la capacité réelle du système de communication.

A l'inverse, l'égalisation aveugle est basée sur la connaissance a priori de certaines propriétés du signal et n'utilise pas de séquence d'apprentissage. Il existe plusieurs types d'algorithmes d'égalisation aveugle. Dans ce qui suit nous donnons un rappel succinct de chacun d'eux. Suit une comparaison des deux les plus populaires : les algorithmes basés sur le cumulant (principalement Shalvi-Weinstein) et de l'algorithme à module constant (également connu comme l'algorithme de Godard).

Les types suivants d'algorithmes d'égalisation aveugle peuvent être rencontrés:

- *LMS dirigé par la décision*: l'algorithme des moindres carrés moyeus (Least Mean Squares) est probablement le plus populaire des algorithmes d'égalisation adaptative non aveugle. Le LMS "classique" a cependant besoin d'une séquence d'apprentissage. Lucky [38] a montré que si l'erreur d'estimation est suffisamment petite pour que les décisions soient correctes "la plupart du temps" , le signal voulu peut être remplacé par la sortie du module de décision (l'estimateur du symbole de s_n).

- *Statistiques de second ordre (SOS)*: Les algorithmes aveugles basés sur les SOS utilisent la connaissance de la matrice de covariance ou de l'auto-covariance du signal transmis pour adapter les coefficients du filtre. Dans le cas à espacement entier, les SOS du signal de sortie suffisent généralement pour identifier la phase d'un système linéaire invariant. Cependant, si le prélèvement de données rationnel est utilisé, le signal présente une cyclostationnarité qui pourrait être utilisée pour trouver la phase d'un modèle de canal inconnu en utilisant uniquement les statistiques du signal reçu (cf [65], [66], [64], [39] par exemple). Il a été montré récemment que les algorithmes aveugles basés sur les SOS convergent avec moins d'observations que les algorithmes basés sur les techniques d'estimation HOS.

- *Joint channel and data estimation*: Ces algorithmes sont basés sur le critère de la probabilité maximum (ML – maximum likelihood). Les estimateurs de la réponse impulsionnelle du canal h et de la séquence de données transmise trouvés grâce à ML sont des estimateurs qui maximisent la fonction de jointe de probabilité et densité $p(r|h, s)$. Ces algorithmes consistent en général en une estimation itérative de la réponse impulsionnelle du canalh_{ML} et en une détection de symboles utilisant l'algorithme de Viterbi. Cet algorithme présente deux inconvénients majeurs : Premièrement, l'estimateur récursif deh_{ML} est demande beaucoup de calculs et deuxièmement, l'estimateur de h est souvent peu precis.

- *La technique Bussgang:* Cette classe d'algorithmes d'égalisation aveugle est basée sur la procédure itérative stochastique du gradient. Pour générer le signal désiré, une certaine (dans la plupart des cas sans mémoire) non-linéarité est appliquée à la sortie de l'égaliseur linéaire. Cette classe d'algorithmes est utilisée principalement pour une séquence de données à l'entrée qui est caractérisée par une fonction de distribution de probabilité non-Gaussienne. Ces algorithmes sont alors distingués principalement par le choix de la fonction non-linéaire. La mise à jour des coefficients de l'égaliseur est assuré par un algorithme adaptatif de type LMS d'où proviennent les principaux avantages et inconvénients de cette méthode: D'un côté, les algorithmes de type LMS sont très simples à implémenter, de l'autre côté, la convergence est relativement lente en moyenne et la convergence n'est pas garantie.

La Condition de l'égalisation aveugle parfaite

Nous définissons l'égalisation parfaite (PE – Perfect Equalization) comme la solution à zéro forcé. Il existe plusieurs conditions sous lesquelles les algorithmes HOS ont une convergence démontrée. Ces conditions sont formalisées comme suit : Chaque source du signal $\{s^l\}$ est une séquence de symboles inter indépendants à moyenne nulle à variance σ^2 et à sub-Gaussian pdf, toutes les sources sont indépendantes entre elles et le canal $\mathbf{H}$ est de rang égale à la dimension de son espace et sans bruit. Dans le cas où les algorithmes sont étudiés dans un environnement bruité, nous assumons un bruit Gaussien centré qui est indépendant de toutes les sources du signal.

Le Critère de Godard

Le critère du module constant (CM) est probablement le plus populaire et le critère le plus étudié dans la conception d'égaliseurs et de beamformers de pour les systèmes de communication à sources où les symboles sont inter indépendants. Grâce à sa popularité, il a été adopté pour l'implémentation de beamformers basé sur le CM pour le HDTV. Historiquement, il existe deux travaux de base qui présentent l'approche

du CM : Godard a proposé son critère [16] en 1980; Des résultats similaires ont été dérivé d'une façon indépendante par Traichler et Agee en 1983 [71]. A l'inverse du critère du MSE, Godard a proposé un critère qui pénalise la déviation en magnitude d'une valeur prédéfinie de la sortie de l'égaliseur. Pour des sources CM qui entrent dans les conditions de PE, toute solution ZF est en même temps une solution CM.

Comme il sera discuté plus loin, d'un point de vue de l'implémentation, l'avantage majeur du CMA est l'existence d'un algorithme simple à gradient similairement au populaire LMS. Plus de détails sur l'algorithme CM vont être donnés dans un section dédiée.

Le Critère basé sur le cumulant

L'utilisation des cumulants comme critère pour la déconvolution aveugle et/ou pour la séparation de source de données non-Gaussiennes est naturelle compte tenu de leurs propriétés. Le développement des algorithmes aveugles de type HOS a une longue histoire : les premiers travaux dates de 1958. Utiliser les cumulants comme une mesure de la non-Gaussienté est un choix naturel puisque pour une variable aléatoire Gaussienne, tous les cumulants d'ordre supérieur à 2 sont des zéros et cette classe d'algorithmes est basée sur la recherche des paramètres$\mathbf{f}$, ayant le cumulant d'ordre k aussi loin que possible du zéro. Un groupe important de critères basés sur les cumulants utilise le cumulant du $4^{ième}$ ordre appelé kurtosis (κ_y) ou kurtosis normalisé (ρ_y).

Les critères proposés sont universel au sens où ils n'imposent aucune autre restriction sur la fonction de distribution (seulement la non-Gaussieneté des sources) et peuvent être par conséquent appliqués à une large gamme de problèmes de déconvolution. Il n'y a pas de faux maximaux locaux qui correspondraient à des solutions locales indésirables. Cela implique que la procédure de recherche du gradient qui résout la maximisation à contraintes va converger vers la solution désirées peu importe les données d'initialisation. Il a été démontré que les maximaux correspondent à la solution ZF avec un délai ν différent et à un facteur de gain complexe

près.

Analyse en Composantes Indépendantes et l'algorithme du point fixe rapide

L'analyse en composantes indépendantes est une technique statistique basée sur la minimisation de l'information mutuelle qui est appliquée principalement dans les domaines de la déconvolution/égalisation et dans l'extraction de propriétés (feature extraction). Elle représente un domaine indépendant dans le traitement statistique du signal. Nous la mentionnons dans notre liste parce que, sous certaines conditions, cette méthode est étroitement liée aux méthodes HOS et quelques résultats peuvent être directement utilisés dans d'autres méthodes de type HOS.

Cette méthode est basée sur la minimisation de l'information mutuelle, c'est-à-dire sur la maximisation de la negentropie. L'algorithme du point fixe est une procédure itérative de recherche du gradient. L'approximation de la negentropie utilise différentes non-linéarités. Le choix de la non-linéarité est le problème clé dans l'optimisation de la performance. La performance de l'algorithme pour une non-linéarité donnée dépend sur les paramètres statistiques du signal source. Les non-linéarités typiques sont: u^4, $\log(\cosh(u))$, $e^{u^2/2}$ et/ou$1/(u^2+1)^2$. L'avantage majeur de cet algorithme est qu'il est indépendant du fait si la source est sub-Gaussienne ou super-Gaussienne et marche dans le cas mixte.

Regalia et Kofidis ont encadré la taille du pas dans l'algorithme du point fixe dans un intervalle où la convergence monotone vers l'extremum local est assurée. Il résout les cas des sources sub- et super-Gaussienne séparément, mais ne résout pas le cas mixte. Il résout également le pas optimum pour le cas sub-Gaussien.

L'algorithme super exponentiel

L'algorithme super-exponentiel a été originellement proposé par Shalvi et Weinstein [57]. Il a été développé dans le domaine des systèmes pour trouver la réponse optimum du système pour minimiser la ISI mais sans aucun critère ou/et limitations fonctionnelles sur les données reçues. Le principe de base dans le développement de

cet algorithme est de considérer la réponse idéale du système comme une solution ZF, par exemple $\mathbf{q} = [0 \dots 0\ 1\ 0 \dots 0] = \mathbf{e}_\nu$. Alors, si le terme dominant de la réponse du système $\mathbf{q}$ est unique, l'exposant Hadamard $\left(\frac{\mathbf{q}}{q_\nu}\right)^{\odot m}$ tend vers la solution ZF quand m tend vers l'infini. L'algorithme a été proposé à partir de cette observation.

Le calcul de la procédure demande, cependant, la connaissance de la réponse impulsionnelle du canal. Comme la réponse est inconnue dans un système réel, la procédure a une signification surtout théorique. Dans [57] a été proposé un algorithme pour égaliseur ajustables basé sur les cumulants joints des données reçues et de la sortie.

L'avantage majeur de cette solution est la complexité de calcul et convergence rapide. D'un autre côté, l'algorithme peut diverger avec une suite de données finie et une SNR finie.

Relations

Bien que chacune des méthodes mentionnées utilise un critère ou/et concept différent, sous certaines conditions elles sont étroitement liées et constituent un groupe d'algorithme référencé comme la famille des fonctions contraste $f_{2p}(q)$ caractérisées par:

$$f_{2p}(q) = \left(\frac{\|q\|_{2p}}{\|q\|_p}\right)^{2p}$$

Dans cette section, l'équivalence entre ces concepts est montrée sous des conditions simplificatrices comme un canal sans bruit (rang complet, pas de racines double ou comme des sources sub-Gaussienne à symboles inter-indépendants.

3 Algorithmes à CM

Dans ce chapitre, nous allons revoir les propriétés de la fonction coût du module constant dans plus de détail et décrire différentes méthodes d'optimisation des algorithmes CM, y compris leur performance d'égalisation et leur convergence dans des scénarios de base. La position des points stationnaires a été étudiée dans beaucoup

de travaux pour beaucoup de types de sources. Comme la fonction de coût du CM est critère très non-linéaire, son analyse est difficile et souvent possible que sous des conditions simplificatrices. C'est pourquoi la plupart des travaux prennent comme supposition un canal sans bruit et un égaliseur à longueur infinie. On peut déduire de la définition de la fonction de coût du CM qu'il s'agit d'un critère multi-modal. Deux conditions qui donnent le gradient nul correspondent à un maximum local à $\mathbf{q} = 0$ et à une série de minimaux locaux et points stationnaires résultant de $\left|\mathbf{q}^T\mathbf{s}\right|^2 = \gamma$. De l'analyse présentée dans [4]et dans [24] résulte que:

- il y a $2N$ minimaux locaux et sous les conditions PE tous ces minimaux correspondent à la réponse du système à ZF. De plus, dans un cas sans bruit, tout ces minimaux sont équivalents au sens de la MSE et ils sont des minimaux globaux. Chaque minimum correspond à un délai de phase et un délai dans le temps différent.

- il y a$3^N - 2N - 1$ points d'inflextion (stationnaires), où chacun d'eux correspond à la réponse du système avec $n = 1 \ldots N$ éléments non nuls qui sont de la même magnitude.

- il y a un seul maximum local avec une origine $\mathbf{q} = 0$. Cette fausse solution est un point instable et dans une implémentation réelle ne constitue pas un problème puisqu'il suffit d'initialiser $\mathbf{f}$ à une valeur non nulle.

- il y a une relation forte entre l'égaliseur CM et le filtre Wiener, du moins quand la SNR s'approche de l'infini. Il a été montré dans le cas bruité que la solution CM est dans le voisinage de la MMSE.

Quand les conditions PE ne sont pas satisfaites, une algorithme proprement optimisé peut converger vers u ne bonne ou une mauvaise solution, et cela dépend de la surface de la fonction de coût au point d'initialisation.

CMA à Gradient

L'avantage majeur du critère CM est qu'il existe un algorithme de recherche du gradient simple et très similaire au très connu LMS. Le principal désavantage est que, dépendamment de la disposition du vecteur propre de la matrice du canal, l'algorithme CMA peut avoir une convergence très lente. Typiquement, la convergence peut avoir une taux de 10^4 iterations. Pour améliorer la vitesse de convergence, il a été proposé d'utiliser une pré-traitement (souvent du pré-blanchissage)

CMA algébrique

Les algorithmes CM basé sur le gradient souffrent des inconvénients suivants : 1) il existe de bonnes et mauvaises solutions compte tenue de la MSE résiduelle (aucune méthode n'est connue pour assurer la convergence vers la bonne solution), 2) le taux de convergence est dépendant de l'initialisation et 3) dans le cas MIMO, le nombre de source CM est imprévisible

Le travail de Van der Veen et de Paulraj [74] [76] était motivé par ces questions ouvertes. Ils ont trouvé une solution algébrique au problème de la séparation CM : ils ont remplacé le problème de la recherche aveugle des coefficients de l'égaliseur par le problème équivalent de la séparation aveugle des sources. Dans la CM algébrique (ACMA), tous les poids des vecteurs sont trouvés simultanément en utilisant ue série d'analyses des matrices de données. La procédure d'initialisation est pas nécessaire (et donc le problème de l'initialisation disparaît).

Les simulations montrent que cette méthode n'égalise pas toutes les sources avec la même qualité (dans le sens MSE ou BER). Quand la puissance du bruit augmente, la défférence entre la qualité d'estimation entre les séquences identifiées à la sortie augmente fortement. De plus, l'algorithme requièr la connaissance des sources CM qui doivent être estimées. Si la méthode décrite dans le travail originel [74] [76] est utilisée, la performance des algorithmes se dégrade beaucoup. L'algorithme et complexe d'un point de vue calcul (il utilise plusieurs décompositions SVD) et demande une grande précision (SVD et procédure de diagonalisation jointe). Toute économie faite dans la

taille en bits de la représentation des données résulte en une grande dégradation de la performance de l'algorithme.

La CMA à intervalle fini

La FI-CMA est une version de la CMA avec du traitement en bloc. L'opérateur "fenêtre de temps" est appliqué aux données reçues et plusieurs itérations de la mise à jour de l'égaliseur sont appliquées à la matrice courante des données pour obtenir la sortie de l'égaliseur. Il a été montré dans[43] que la minimisation de la fonction de coût de la FI-CMA est équivalente à la minimisation de la fonction de coût invariante par changement d'échelle qui est dépendante seulement des moments du quatrième et second ordre de la sortie de l'égaliseur. L'utilisation de la décomposition QR de la matrice de données d'entrée simplifie le calcul du gradient de la fonction de coût équivalente. Finalement, l'algorithme décent à traitement en bloc ressemble à l'algorithme du point fixe ICA avec la non-linéarité u^4. Grâce à cela, la taille optimale du pas peut être calculée grâce aux données à la sortie de l'égaliseur seulement. Les simulations montre que l'algorithme converge rapidement après quelques itérations seulement (souvent moins que 10).

Nous pouvons en conclure que l'avantage majeur de cet algorithme est sa vitesse de convergence et le fait que la taille optimale du pas est connue et dépend uniquement des données. D'un autre côté, comme il s'agit d'un algorithme à traitement en bloc, il y besoin d'utiliser des blocs successifs se chevauchant et donc une procédure de comparaison et reconstitution après l'égalisation.

4 CMA à fenêtre glissante

La FI-CMA a une complexité de calcul bien plus petite que l'algorithme ACMA alors que la convergence vient plus rapidement par comparaison avec les algorithmes de type gradient. De l'autre côté, comme il s'agit d'un algorithme de traitement en bloc, il peut être utilisé que pour des canaux invariants dans le temps (ou variant

très lentement), et demande une séparation en blocs se chevauchant et donc un post-traitement d'assemblage. Ce calcul supplémentaire donne la condition sur N d'être aussi grand que possible. Malheureusement, des blocs de taille N plus grands donnent lieu à une complexité de calcul plus grande et demande une variance du canal dans le temps encore plus lente.

Pour bénéficier de l'analyse FI-CMA (plus précisément du fait qu'il existe une expression analytique de la taille du pas optimal) et de ces propriétés comme la convergence rapide et complexité de calcul acceptable, nous avons dérivé sa version à fenêtre glissante. Elle a les propriétés suivantes: 1) elle ne demande pas le chevauchement entre les blocs et donc ne demande pas de post-traitement, 2) elle marche pour des environnements qui varient lentement dans le temps, 3) elle atteint la vitesse de convergence de la FI-CMA et 4) elle permet à la taille N de la fenêtre d'être aussi petite que possible pour réduire la complexité de calcul par symbole décodé

Nous nous permettons de rappeler que pour l'algorithme FI-CMA, la matrice $\mathbf{Q}$ (résultat de la décomposition QR) est utilisée au lieu de la matrice des données en entrée. Donc, nous pouvons corriger la décomposition QR directement des données en entrée. En liaison avec la structure de la matrice des données en entrée $\mathcal{R}$, il est possible d'utiliser doit une correction par colonnes soit une correction par lignes.

L'erreur d'arrondissement de la décomposition QR et la perte de l'orthonormalité de la matrice $\mathbf{Q}$ ont a été testés dans les simulations. Les résultats montrent que les deux erreurs ont une valeur moyenne nulle. La variance de l'erreur de déviation de la décomposition est $\sigma_e^2 = 1.1 \cdot 10^{-27}$ et la variance de la perte de l'orthonormalité est $\sigma_o^2 = 1.8 \cdot 10^{-28}$. Nous pouvons donc conclure que l'erreur commise par l'utilisation de la décomposition QR est acceptable.

Les simulations montrent que dans les deux variantes (correction par colonnes ou par lignes) ont un comportement similaire. L'influence de la taille de la fenêtre a été également testée. Le résultat montre que l'algorithme est capable de converger pour une taille de la fenêtre beaucoup plus petit que la FI-CMA.

5 La Performance des algorithmes pour des canaux variant dans le temps

Dans les sections précédentes, nous avons discuté les algorithmes CMA à gradient, FI-CMA et SW-CMA. Ici, nous comparons leur performance pour des canaux dont les paramètres varient dans le temps. A partir des résultats des simulations, cinq scénarios typiques sont donnés. Pour la variation dans le temps un modèle auto-régressif a été utilisé. Les valeurs initiales des paramètre du canal ont été générée aléatoirement.

On peut voir dans les simulations que le SW-CMA marche bien dans presque tous les cas présenté et dans les deux cas SW-CMA originale et SW-CMA simplifié et peut donc égaliser des canaux à paramètres variant dans le temps. FI-CMA peut s'en sortir, quant à lui, quand les variations sont petites et à valeur moyenne nulle compté sur un intervalle relativement court. Finalement, le G-CMA est le plus sensible aux variations du canal. Comme pour cet algorithme la valeur du pas optimal est dépendante des paramètres du canal, les variations des ces paramètres peuvent résulter dans une comportement instable d'un algorithme qui a déjà convergé.

6 Les Problèmes de l'implémentation matérielle

Dans ce chapitre, les plates-formes matérielles utilisées pour les méthodes avancées des algorithmes DSP sont revuezs et les problèmes d'implémentation, notamment liés à la virgule flottante, vont être discutés.

L'Histoire des plate-formes DSP

Beaucoup de problèmes résolus dans les télécommunications, comme l'égalisation, codage ou détection de l'audio et de la vidéo sont complexe du point de vue calcul. Le besoin du calcul en temps réel écarte la possibilité d'utiliser les processeurs programmables universels. Pour lutter avec ces besoins, les architectures matérielles doivent offrir en même temps une grande capacité au niveau du flux de données et

de grandes possibilités au niveau de complexité de calcul par données reçues. Les paramètres offerts par des processeurs standard ne suffisent pas pour ces applications. Les plate-formes d'implémentation qui remplissent ces besoins sont divisées en deux groups : processeurs DSP dédiés et solutions matérielles.

Dans le années 80 le premier processeur de traitement de signal numérique (DSP – Digital Signal Processor) a été introduit sur le marché. En comparaison avec les processeur universels, il était construit pour faire plusieurs opérations en un seul cycle d'instruction. L'architecture d'un DSP typique est optimisé pour des opérations comme "*multiply-and-accumulate*" et contient des unités dédiées pour des opérations comme "*barrel shift*", opérations sur les adresses, etc. Typiquement, il a une consommation d'énergie plus petite que les processeurs standards.

Les options pour l'implémentation matérielle sont d'une part les circuits pour des applications spécifiques (ASIC) et d'autre part des modules matériels programmables (CPLD, EPGA). Les architectures ASIC atteignent la performance demandée en utilisant des structures de calcul optimisées pour l'application donnée. Néanmoins, même si la technologie ASIC offre la plus grande performance et la plus petite consommation d'énergie, son coût élevé de la conception est acceptable seulement dans le cas d'une production massive. Ce désavantage se révèlera bien entendue également lors de mises à jour (nouvelles fonctions à implémenter ou correction d'erreurs commises). Cette technologie est donc relativement chère.

L'Architecture des processeurs DSP

Pour les implémentations décrites dans cette thèse, le module TMS320C6711 de Texas Instruments a été utilisé. Il a une grande performance au niveau de la DSP à virgule flottante avec une architecture VLIW (Very Long Instruction Word). Le module éxécute jusqu'à huit instructions par cycle. Son architecture consiste en huit unités fonctionnelles qui calculent avec 32 registres universels à mots en 32 bits. Ces huit blocs fonctionnels contiennent deux multiplieurs et huit unités d'arithmétique logique.

Les instructions sont traitées dans les deux chemins de données (A et B) et chacun

contient quatre unités fonctionnelles et 16 registres universels. Le module utilise une pipeline multi-état avec trois stades : Lecture, Décodage et Exécution où chacun de ces stades se divise en plusieurs états. Les stades de Lecture et de Décodage prennent toujours le même nombre de cycles (sauf si l'état "*hold*", utilisé pour lire de la mémoire externe, survient), alors que l'Exécution prend un à dix cycles.

L'Architecture FPGA

La structure d'un module FPGA peut être divisée en deux parties. L'application est construite en utilisant des ressources logiques et éventuellement par des unités matérielles adaptées. La configuration de la matrice logique, des macros et de leur interconnections est fait grâce à l'interface de la mémoire de configuration.

L'application est exécuté sur le FPGA en utilisant une matrice de ressources logiques. C'est une structure régulière d'éléments logiques de base, appelés blocs logiques (LBs), de macros matériels (mémoires à blocs, multiplieurs, etc.) et de leur interconnections.

La structure concrète des blocs logiques varie d'un module à un autre. En général, elle contient un générateur de fonction, un élément de stockage et quelques fonctionnalités en plus. Le générateur de fonction est généralement implémenté en utilisant une table LUT (look-up table) à n entrées. L'élément de stockage peut être configuré à tout type de registre ou tout type de bascule à niveaux. Les blocs logiques sont normalement groupé en utilisant des ponts d'interconnexion rapides ce qui permet l'implémentation de fonctions logiques plus complexes.

La structure de routage est une partie très importante d'un FPGA parce le routage configurable peut introduire un délai supplémentaire au délai logique. Ces délais peuvent excéder 60% du délai logique du chemin complet. Les FPGAs modernes utilisent une structure d'interconnexion hiérarchique combinée avec quelques aiguillages pour augmenter la capacité de routage et limiter les délais.

Dans l'implémentation finale, le module Xilinx Virtex a été utilisé. Le bloc logique de l'architecture Xilinx Virtex est référencé comme un *slice*. Il inclue deux générateurs

de fonctions busés sur une LUT à quatre entrées, deux éléments de stockage configurables et le circuit logique. Deux *slices* regroupés forment un bloc logique configurable (CLB – configurable logic block) qui est la brique de base de la matrice logique. Chaque CLB contient deux tampons à trois états qui peuvent piloter les bus propres du circuit intégré. Le module Virtex offre des blocs SRAM double-port (appelés BlockRAMS) pour implémenter un mémoire propre efficace. Les interfaces d'entrée-sortie placés autour de la matrice CLB intègrent trois éléments de stockage pour décroître le délai des signaux entrant et sortant.

L'Arithmétique

Le choix de l'arithmétique utilisée pour implémenter un algorithme à DSP est d'une grande importance. Les arithmétiques à nombres entiers ou à virgule fixe sont faciles d'implémentation et beaucoup de modules utilisent ce type d'arithmétique. D'un autre côté, seulement une variété limitée d'algorithmes de traitement du signal peut être implémentée dans cette arithmétique, parce que la dynamique des données représentables est relativement étroite. En plus, concevoir un algorithme implémentable avec une arithmétique à virgule fixe est relativement coûteux.

La plupart des algorithmes à DSP utilisent les nombres à virgule flottante. Les modules utilisant cette arithmétique sont souvent plus chers que ceux à virgule fixe, parce que le matériel des unités à virgule flottante demande plus de logique et donc consomme plus de silicone. Dans notre implémentation, nous avons utilisé une DSP à virgule flottante et le FPGA Xilinx. Pour l'implémentation en DSP, le système à virgule flottante propre du module, IEEE, a été utilisé. et dans la version FPGA, nous avons utilisé le système alternatif de représentation logarithmique des nombres (LNS).

La représentation algorithmique des nombres, qui n'est pas très répandue, consiste en la valeur image de deux points opposés $\log_2 |x|$, où x est la valeur à représenter et $|\cdot|$ est l'opérateur de la valeur absolue et du bit représentant le signe du nombre. Les logarithmes à base 2 sont utilisé même si en principe, n'importe quelle

base pourrait être représentée. La première valeur signée est divisée en une partie entière, qui a toujours 8 bits et en une partie rationnelle dont la longueur dépend de la précision des données. Notons que la multiplication, la division et la racine carré sont des opérations qui peuvent être implémentée comme des additions, soustractions et déplacement à droite donc comme en mode virgule fixe. Le point faible de l'arithmétique LNS est dans l'implémentation de l'addition et de la soustraction, qui demande l'évaluation de la fonction non-linéaire $\mathcal{F}(r) = \log_2(1 \pm 2^{j-i})$. Dans notre conception, nous avons utilisé une technique développée dans [5] pour fournir une précision comparable à la IEEE single precision floating point. L'utilisation de l'algorithme de correction d'erreur et de déplacement d'ordre réduit la taille de le LUT nécessaire pour l'évaluation de la fonction mentionnée. Pour l'implémentation FPGA, nous utilisons la librairie intégrée LNS développée sous le projet HSLA à la IST. Le cœur implémente toutes les opérations arithmétiques dans les cas de précision 32-bits et 19-bits.

7 Les Résultats de l'implémentation

Dans ce chapitre est décrite l'implémentation concrète des algorithmes G-CMA et FI-CMA. Le but de ce chapitre est de fournir une preuve sous forme de prototype qui montre qu'il est possible d'utiliser des modules courants pour implémenter des calculs complexe rencontrés dans le monde de la télécommunication.

Pour l'implémentation finale, le G-CMA et le FI-CMA ont été choisis. Le G-CMA est l'algorithme le plus simple, habituellement utilisé comme une référence. L'algorithme FI-CMA a été choisi comme l'algorithme adapté à la précédemment cité représentation logarithmique des nombres. Le processeur TMS320C6711 de Texas Instruments a été utilisé pour l'implémentation DSP et la série de modules Xilinx Virtex (soit XCV2000E, soit XC2V6000) a été utilisée pour l'implémentation matérielle.

L'Implémentation de G-CMA

L'algorithme G-CMA a été implémenté seulement sur le FPGA. Cet algorithme est très similaire aux algorithmes de type LMS et a une architecture très simple. Les éléments de base de cet algorithmes sont : 1) Un processeur MAC (Multiply-and-Accumulate) calcule la sortie de l'égaliseur, 2) calculs d'erreur et multiplication scalaire d'un vecteur et 3) soustraction de vecteur. Comme l'unité d'addition LNS a une latence de 8 cycles, la partie à optimiser est l'unité MAC qui est complexe en addition. Pour l'optimisation, nous supposons la taille du coefficient du vecteur être $N > 16$ c'est-à-dire deux fois la latence du l'unité d'addition LNS. L'architecture du MAC a été proposé pour chaque N.

Comme l'algorithme G-CMA contient presque autant d'additions que de multiplications, nous n'avons pas profité de l'économie au niveau des multiplications du système LNS. D'un autre côté l'arithmétique LNS profite d'une consommation d'énergie plus petite et d'une optimalisation plus facile du design, puisqu'il n'y a qu'une seule unité à longue latence de calcul.

Le design FPGA a été implémenté en une précision de 32-bits et 19-bits dans la suite Celoxica DK2 en utilisant le langage Handel-C. Le design a été optimisé grâce au logiciel Simplify Pro de Symplicity, Inc. et compilé avec Xilinx ISE 6.1.03.

L'Implémentation de FI-CMA et de SW-CMA

L'algorithme FI-CMA a été implémenté en utilisant les deux DSP et FPGA. Pour implémenter l'algorithme, les unités de calculs suivantes étaient nécessaires : 1) le *processeur QR*, calculant une série de rotations de Givens et 2) le *processeur de mise à jour de l'égaliseur*, calculant une ou plusieurs itérations de la mise à jour de l'égaliseur. Pour le design général de l'architecture d'implémentation le facteurs suivant devaient être considérés : les dimensions de matrices, restrictions dues à l'accès en parallèle à la mémoire, (dual ported / dual accessed RAMs) et le nombre limité d'unités d'exécution.

Deux différentes architecture FPGA pour l'implémentation de l'algorithme ont été

proposées. La première, architecture initiale, a été développée manuellement, alors que la deuxième a été optimisé en utilisant les méthodes de la programmation entière linéaire et de la programmation de cycles de base (Basic Cyclic Scheduling).

En considérant les résultats, il est possible de conclure que l'utilisation de l'arithmétique LNS amène un bénéfice significatif pour les deux choses : les calculs QR et les initialisations de l'égaliseur basé sur le FI-CMA. Notamment, comme dans l'arithmétique LNS n'est présente qu'une seule unité avec une longue latence de calcul, l'optimisation du code est plus simple et il est possible d'arriver à une efficacité de 100%. Aussi, il faut mettre le l'accent sur le fait que cette latence est la même dans les deux précisions (32/19 bits) contrairement à d'autres implémentations à virgule flottante ou la latence dépend de la précision.

Quant à l'algorithme Basic Cyclic Scheduling, il a été montré qu'il peut aider dans l'optimisation de l'utilisation des ressources dans une architecture FPGA et qu'il peut être utilisé pour optimiser des applications avancées à DSP.

Finalement, les mesures de performance montrent que notre implémentation FPGA de l'algorithme FI-CMA dépasse l'implémentation sur un DSP à virgule flottante. Pour être correct, nous devons noter que pour l'implémentation sur DSP, nous n'avons pas utilisé d'optimisation de scheduling. Egalement, le code DSP n'utilise pas complètement les ressources d'une façon parallèle et il devrait être réécrit dans une langage de plus bas niveau (comme un assembleur linéaire, par exemple).

8 Conclusion

Cette thèse traite d'algorithmes de traitement de signal avancés pour communications et de problèmes d'implémentation associés au sujet. Nous nous sommes concentrés sur les algorithmes pour égalisation aveugle. Plusieurs méthodes différentes de l'égalisation aveugle ont été étudiées et leur relations ont été identifiées. Finalement, nous nous sommes concentrés sur les méthodes basées sur le critère du module constant (CM).

Trois algorithmes de type CMA, probablement les plus cités, ont été revus –

CMA à Gradient, CMA Algébrique et CMA à Intervalle Fini – et leur propriétés ont été démontrées sur des simulations. Basé sur les résultats de ces simulations, nous concluons que même si le G-CMA est indiscutablement très populaire, notamment grâce à la facilité de son implémentation, sa performance est relativement faible (en comparaison avec les deux autres algorithmes). Alors que, dans cet algorithme, le paramètre de la taille du pas a une influence sur l'algorithme, sa valeur optimale est dépendante des paramètres du canal et son initialisation impropre peut mener à la divergence de l'algorithme.

Le second algorithme, A-CMA, assure simultanément la déconvolution de toutes les sources reçue dans le signal et, utilisant la procédure de diagonalisation jointe, recherche l'optimum global de la fonction de coût du CM. Cette méthode ne requièrt pas le paramètre de la taille du pas et n'est pas sensible à une mauvaise initialisation. D'un autre côté, comme il utilise plusieurs décompositions SVD de grandes matrices, une précision importante dans la représentation des données est nécessaire et la méthode est très complexe d'une point de vue calcul. Elle ne découvre pas toutes les sources avec la même qualité, en relation avec la fonction de coût résiduelle. De plus, la différence entre la fonction de coûte résiduelle entre les différentes sources est influencée par la valeur de la SNR.

Finalement, nous avons présenté les propriétés de l'algorithme FI-CMA. Il représente un compromis entre les besoins de calcules des algorithmes G-CMA et A-CMA. Dans la phase initiale, il utilise la décomposition QR de la matrice de données en entrée ; c'est poursuivi par le calcul itératif de l'égaliseur. Ce calcul est basé sur la méthode itérative du gradient mais la matrice orthonormale $\mathbf{Q}$ est utilisée à la place, avec comme résultat une rapidité de convergence rapide de l'algorithme (unités d'itération). La valeur de la taille du pas optimal (optimal du point de vue de la rapidité de convergence) est connue sous une forme analytique. Nous avons testé l'algorithme pour deux valeurs de la taille du pas ; le premier a résulté en une convergence plus rapide, le second donne une fonction de coût résiduelle CM plus petite et converge légèrement plus lentement.

Finalement, nous avons proposé SW-CMA en tant que version modifié de l'algorithme FI-CMA. Il applique une fenêtre glissante sur les données en entrée et pour toute fenêtre, il calcule une ou plusieurs itérations de la mise à jour des coefficients, comme dans FI-CMA. Pour la décomposition QR, il utilise une mise à jour QR au lieu du calcu direct d'une matrice de données sources. Deux méthodes de mise à jour de la décomposition ont été présentées : par colonnes et par lignes. Il a été montré que SW-CMA atteint les mêmes propriétés de convergence que FI-CMA et ces besoins en calculs sont également similaires. De plus, il peut opérer avec des matrices de données plus petites.

L'analyse de l'erreur causé par la mise à jour QR a été présentée. L'erreur dans la décomposition QR et la perte de l'orthonormalité de la matrice $\mathbf{Q}$ a été également testée. Les résultats montrent que les deux erreurs ont une valeur moyenne nulle et sont de module acceptable.

Le comportement de G-CMA, FI-CMA et SW-CMA pour des canaux variants dans le temps a été testé. Les résultats pour cinq différents canaux, représentants des situations typiques, ont été présentés. Les résultats montrent que même si l'algorithme FI-CMA est de la sorte calcul en bloc, il est capable d'assurer la déconvolution du canal qui varie lentement dans le temps. Le G-CMA, quand il est initialisé proprement, peut en faire de même et peut suivre les variations du canal. Mais à cause de ces variations il peut également diverger même s'il a précédemment convergé. Finalement, le SW-CMA se comporte bien dans presque tous les cas présentés. A la différence avec le G-CMA, il a un taux de convergence rapide et l'algorithme peut suivre les variations du canal dans le temps avec succès. Comme le SW-CMA utilise des matrices de données plus petites (qui correspondent à une taille plus petite de la fenêtre), il est capable d'égaliser les canaux qui varient relativement rapidement dans le temps.

Dans la dernière partie de la thèse, nous nous concentrons sur les architectures d'implémentation pour le traitement de signal numérique. Les implémentations sur deux différentes plate-formes (DSP et FPGA) sont présentées. L'architecture de deux

modules modernes a été revue : Texas Instruments TMS320C6711 DSP et Xilinx Virtex FPGA. Dans cette partie sont présentées également différentes arithmétiques à virgule flottante pour différentes représentation des nombres. Notamment, la représentation logarithmique des nombres a été présentée en détail.

Finalement, les résultats d'implémentation des algorithmes G-CMA et FI-CMA ont été présentés. Les deux designs FPGA ont utilisé la librairie LNS. L'architecture de l'unité Multiply-and-Accumulate, qui est l'unité la plus coûteuse au niveau de la complexité de calcul pour les algorithmes G-CMA, est proposée. Finalement, les résultats des architectures FPGA proposées pour l'arithmétique LNS en 32- et 19-bits pour Xilinx Virtex2000E sont résumés. Comme l'algorithme G-CMA contient autant d'additions que de multiplications et pas de divisions ni racines carré, l'utilisation de l'implémentation LNS n'apporte pas beaucoup d'avantages.

Dans l'architecture FI-CMA, nous avons premièrement proposé une première version faite à la main. Cette architecture a été alors optimisée en utilisant le Basic Cyclic Scheduling. L'architecture a été optimisé en tenant compte de l'utilisation de l'unité d'addition LNS, comme c'est l'unité la plus coûteuse dans l'architecture, pour des tailles particulières des matrices $\mathbf{Q}$ et $\mathbf{w}$ et pour un nombre particulier d'unités d'addition (une unité). L'optimisation a résulté en une utilisation de l'additionneur maximale (100%) dans les deux parties de l'algorithme : la décomposition QR et la mise à jour des coefficients. Il a été également noté que l'architecture optimisée est efficace seulement pour le même nombre d'unités que pour lequel l'optimalisation a été effectuée et les limites des la taille des matrices sont ainsi posées. Pour être concrets, en baissant la taille de la matrice $\mathbf{Q}$ ou en utilisant plus d'unités on pourrait arriver à une efficacité plus petite avec des cycles vides.

Dans le cas de l'algorithme FI-CMA, l'arithmétique LNS apporte un avantage significatif, puisqu'il y a beaucoup plus de multiplications que d'additions et il y a beaucoup d'opération de puissante sur des éléments vectoriels dans l'algorithme. Il a été montré que l'implémentation LNS sur FPGA dépasse la performance des versions modernes de DSPs à virgule flottante. Ce qui est plus, comme l'arithmétique LNS n'a

qu'une seule unité à latence plus grande que 1, il est beaucoup plus facile d'optimiser l'algorithme.

Chapter 1

Introduction

In the recent years, we have witnessed fast development of communications systems – especially the wireless systems which have expanded extremely fast. Most modern systems are based on the principles of digital communications. Hand in hand with their widespread application comes an intense research in many fields of digital signal processing: in the areas of channel equalization, source separation, synchronization, smart antennas and power control, to give but a few examples. The goal of this research is to enhance performance of the digital systems, namely to improve their transmission capacity and to reduce power consumption.

In this thesis, we focus on using higher order statistics for channel equalization. The task of equalization is to remove distortion inhered while propagating from transmitter to receiver. This distortion occurs inevitably due to physical properties of the transmission media (signal fading, reflections and phase shift due to the obstacles along the channel, multi-path character of the environment, and attenuation of the propagation environment; the character of the distortion depends on the type of environment). As a result, the received signal contains a superposition of copies of the same signal with different gains, phases and delays. This superposition results in interference between the subsequently transmitted symbols – an effect called Inter Symbol Interference (ISI). The degree of distortion depends on the properties of the

environment: in a hilly terrain, the delay spread between the copies of the transmitted signal is up to $15\mu s$, while in urban area, it amounts to $10\mu s$. For illustration, in the GSM system, which uses symbol period $3.7\mu s$ (i.e. 270kbit/s), this represents delay of 5 symbols.

To avoid serious performance degradation, it is mandatory to compensate the channel distortion. This compensation is performed by a filter at the receiver part - in the so called channel equalizer. In the case that the channel characteristics are known beforehand, it is possible to initialize the filter before the transmission is started. If the channel is not known, the equalizer parameters have first to be estimated. Standard methods use so called training sequence: the transmitter sends a block of symbols which are a-priori known at the receiver side. The parameters are estimated, using this sequence and the symbols that were received. This process is called training. After the estimation phase, the equalizer is constructed and the the equalization of the received signal can be performed.

In the case of the wireless systems, the channel parameters may vary in time. The channel estimation has then to be repeated with each data block to be transmitted, which, in sequel, results in transmisson capacity loss. In the GSM systems, to give an example, the training sequence takes 26 symbols in the central part of a block of 148 symbols. This results in 18% throughput loss.

The thesis has the following structure: in the rest of this chapter, we define system model and the equalization concept. At the end of this chapter, the Higher order statistics (HOS) is shortly reviewed. In Chapter 2, the state of the art in the blind equalization techniques is presented. The four most important approaches are described in separate sections. At the end of the chapter, their relations under the "ideal" conditions are shown.

In Chapter 3, the Constant Modulus criterion is discussed in more detail. Three optimization algorithms based on minimization of CM cost function are described and their typical behavior is shown on simulations.

In Chapter 4, the Sliding Window CMA is derived and its typical behavior is

demonstrated on simulations. In Chapter 5, the performance of different Constant Modulus algorithms is compared for channels with time varying parameters. Several typical scenarios for channel evolution are presented. These two chapters represent the essence of the thesis in the theoretical part.

Finally, Chapters 6 and 7 are oriented towards the implementation issues of the signal processing algorithms. Chapter 6 gives a brief review of the architecture platforms (Xilinx FPGA and Texas Instruments DSP) on which the selected algorithms were implemented and of the arithmetics for these platforms. Since the FPGA implementations use the Logarithmic Numbering System, which is not widely known, it is described in more details. Chapter 7 presents the results of the implementing the Gradient CMA and Finite Interval CMA algorithms. This chapter represents the core contribution of the thesis to the implementation issues.

1.1 System model

We assume that the communication system is described by model presented in Figure 1.1: symbols to be transmitted are first digitally modulated and sent to the receiver through the communication channel. The modulated signal is a narrow-band signal (in terms of the baseband signal bandwidth to the carrier frequency ratio) i.e., the signal bandwidth is much smaller than the carrier frequency. In the GSM system, a signal with 200 kHz bandwidth is modulated by 900 MHz carrier. Since the transmitted signal contains always both the information and the carrier part, it is useful to simplify the expressions to be used and to work with their equivalent baseband representation. In the sequel, we define the baseband representation of the digitally modulated signal and the communication channel.

Any real-valued, time-continuous digitally modulated signal can be expressed in several forms. Generally, it can be written as:

$$s(t) = a(t)\cos(2\pi f_c t + \phi(t)) \tag{1.1}$$

where the $a(t)$ is the signal pulse shape, f_c is the carrier frequency a $\phi(t)$ is the phase;

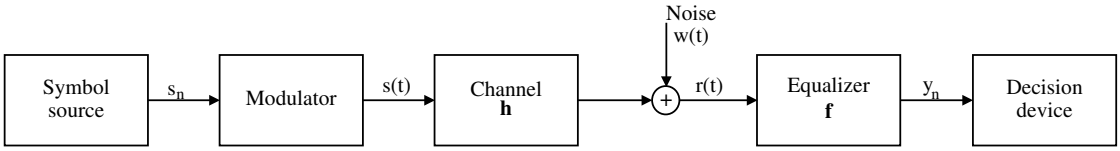


Figure 1.1: Simple model of a communication system

the signal $a(t)$ is also called the signal envelope. By expanding the cos() term in (1.1) we obtain the following form of the equation:

$$s(t) = s_i(t)\cos(2\pi f_c t) - s_q(t)\sin(2\pi f_c t) \tag{1.2}$$

where the low-frequency signals $s_i(t)$ and $s_q(t)$ are called the in-phase and the quadrature component. Another representation expresses the modulated signal as the real component of a complex signal:

$$s(t) = \Re\left\{s_l(t)e^{i2\pi f_c t}\right\} \tag{1.3}$$

where $\Re$ means real part of a complex signal and $s_l(t) = s_i(t) + is_q(t)$ is called the complex envelope of the narrow-band real signal $s(t)$ and is used as a baseband signal equivalent. Similarly, the baseband equivalent of linear bandpass channel with impulse response $h(t)$ is defined as:

$$h(t) = 2\Re\left\{h_l(t)e^{i2\pi f_c t}\right\} \tag{1.4}$$

where $h_l(t)$, which is obtained as the inverse Fourier transform of (one-sided) signal spectrum, creates complex-valued equivalent low-pass system (see [41]). The input of the receiver is expressed as a response of a linear bandpass system to the bandpass input signal. The received signal – which is again a bandpass signal – is obtained by convolution of the channel impulse response with the input signal and has the form:

$$r(t) = \int_{-\infty}^{+\infty} s(t)h(t-\tau)d\tau \tag{1.5}$$

With some algebraic manipulations, it can be shown (see [41]) that the baseband equivalent of received signal is convolution of low-pass input equivalent signal with an equivalent low-pass system model.

$$r_l(t) = \int_{-\infty}^{+\infty} s_l(t) h_l(t-\tau) d\tau \tag{1.6}$$

In the rest of this thesis, we will work only with the equivalent baseband models and we will use the symbols $h(t)$, $s(t)$ and $r(t)$ for the baseband equivalents.

Note: Also the bandpass stationary stochastic processes (additive noise) have their equivalent representations which will be used in the text. For the sake of brevity, they will not be presented here (see again [41]).

In the following text, we define the baseband representation of the digitally modulated signal. Taking into account that this thesis focuses on the Constant Modulus algorithm, the most important modulations are those with constant envelop – PSK, QPSK etc. For simplicity, we use only memoryless modulations.

The analog base-band continuous-time digitally modulated source signal is described as:

$$s(t) = \sum_{i=0}^{+\infty} s_i \psi(t - iT) \tag{1.7}$$

where T is the symbol period, $\{s_i\}$ is the source symbol sequence, where one symbol s_i is transmitted once in every baud interval T and ψ is analog pulse with a finite time support (T). As was shown earlier, the received baseband signal from (1.6) is defined by a convolution integral:

$$r(t) = \int_{-\infty}^{+\infty} s(\tau) h(t-\tau) d\tau \tag{1.8}$$

In the receiver, the incoming signal is digitally sampled. Depending on the sampling rate T_s, the system is called baud rate if $T_s = T$ or fractionally spaced if $T_s = T/L$. The received signal, sampled at time nT_s, is written as:

$$r_n = r(nT_s) \qquad n = 0, 1 \ldots \infty$$

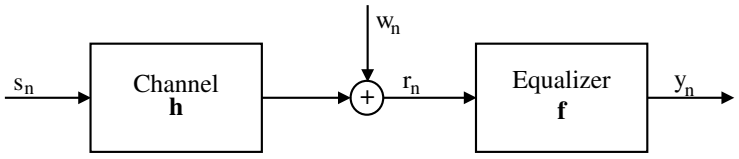


Figure 1.2: Discrete baseband communication model

Using this discretization on other signals in the communication system model, it can be transformed to the form shown in Figure 1.2 – the discrete baseband communication model.

1.2 Channel Matrix

We assume the channel impulse response to be finite in duration (FIR) and real-valued. The channel impulse response $h(z)$ is assumed to be a priori unknown, causal and bounded-input bounded-output (BIBO) stable. Gaussian (white) noise $w(t)$ is added to the channel output.

The structure of the channel matrix and of the received signal depends on the number of the channel inputs and outputs (i.e., sources and antennas or oversampling factor). In the following sections some typical cases (single/multiple input, single/multiple output) will be described. The single input model represents systems with only one communicating user, whereas the multi-input model represents configuration with several users and/or multiple sources of interference. The single output model is used to form the channel output signal for baud-spaced receivers with one antenna, whereas the multiple output model describes channel output of a system with an antenna array and/or of fractionally spaced systems.

1.2.1 Single output data model

In single-input/single-output (SISO) channel model, signal from one user is transmitted over the channel and, at the receiver side, the output signal is received by one antenna and sampled at baud (symbol) interval T_s (this setting fulfills the Nyquist theorem). These systems, with the sampling frequency corresponding to the symbol interval, are called the baud-rate systems.

The user data sequence $\{s_n\}$ is viewed as an independent identically distributed random process. The data stream is sent through a discrete-time channel with impulse response $h(z)$:

$$h(z) = \sum_{k=0}^{N_h-1} h_k z^k \tag{1.9}$$

where h_k is k^{th} element of the discrete channel impulse response and N_h is the channel length. The discrete-time output signal of the SISO channel has the form:

$$r_n = \sum_{k=0}^{N_h-1} h_k s_{n-k} + w_n = \mathbf{h}^T \mathbf{s_n} + w_n \tag{1.10}$$

where s_n is source symbol transmitted at discrete time n, w_n is an additive Gaussian noise and $\mathbf{h}$ and $\mathbf{s}_n$ are column vectors of the following structure:

$$\mathbf{h} = [h_0 \ \dots \ h_{N_h-1}]^T \qquad \mathbf{s}_n = [s_n \ s_{n-1} \ \dots \ s_{n-N_h-1}]^T \tag{1.11}$$

In the case when several users are involved in communication in the same time (as is usual in modern wireless communication), both the desired symbol sequence s_n and K sequences from other users $s_n^{(i)}$ each pass through separate linear channel before being received. By expanding equation (1.10) the output of multiple-input/single-output (MISO) channel is expressed as:

$$r_n = \mathbf{h}^T \mathbf{s}_n + \mathbf{h}_{(1)}^T \mathbf{s}_n^{(1)} + \ \dots \ + \mathbf{h}_{(K)}^T \mathbf{s}_n^{(K)} + w_n = \mathbf{h}^T \mathbf{s}_n + \underbrace{\sum_{i=1}^{K} \mathbf{h}_{(i)}^T \mathbf{s}_n^{(i)}}_{\text{MAI}} + w_n \tag{1.12}$$

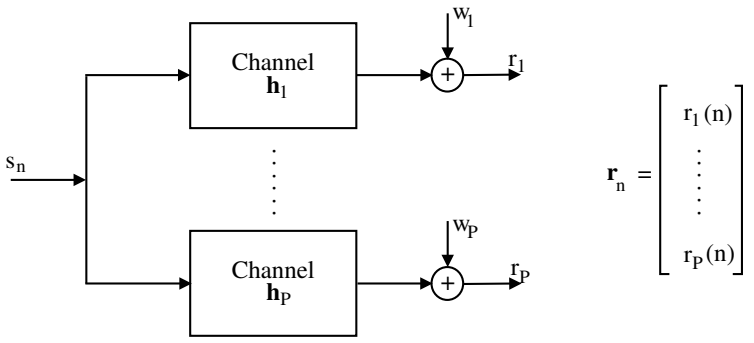


Figure 1.3: Single input multiple output data models

where $\mathbf{s}^{(i)}$ means the symbol sequence of the i^{th} user and $\mathbf{h}_{(i)}$ is the channel impulse response corresponding to the i^{th} user. Sequence $\mathbf{s}$ is the sequence of interest, sequences $\mathbf{s}^{(i)}$ from other users represent new source of noise, referred to as Multiple Access Interference (MAI).

1.2.2 Multiple output data model

A typical system with multiple channel outputs is the base station for mobile communication where an antenna array is used to receive signal. In this case, the input of the receiver is a P dimensional observation vector $\mathbf{r}_n$. The single-input/multiple-output (SIMO) (system with only one user) channel model is shown in Figure 1.3 with output signal of the form:

$$\mathbf{r}_n = \mathbf{H}\mathbf{s}_n + \mathbf{w}_n \tag{1.13}$$

where $\mathbf{s}_n$ is the symbol source vector, which is defined in the same way as in the SISO case, $\mathbf{w}_n$ is the noise column vector, $\mathbf{H}$ is a $P \times N_h$ matrix representing the channel impulse response, P is the number of antennas and N_h is the channel length. The channel matrix $\mathbf{H}$ of the channel model shown in Figure 1.3 has the following

structure:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \vdots \\ \mathbf{h}_P^T \end{bmatrix}$$

where $\mathbf{h}_i$ represents the impulse response vector of the ith channel/antenna defined by 1.11.

The multiple-input/multiple-output system model results from extension of the SIMO model (1.13) to the case with several users. From (1.13) follows:

$$\mathbf{r}_n = \sum \mathbf{H}_{(i)} \mathbf{s}_n^{(i)} = \mathbf{H}_A \mathbf{s}^A$$

where $\mathbf{H}_k$ is a $P \times N_f$ matrix and $\mathbf{H}_A$ contains channel impulse response of all users/channels and has the form:

$$\mathbf{H}_A = \begin{bmatrix} \mathbf{h}_A^T(0) \\ \mathbf{h}_A^T(1) \\ \vdots \\ \mathbf{h}_A^T(P) \end{bmatrix} \tag{1.14}$$

$\mathbf{h}_A(i)$ represents the i^{th} channel impulse response for all users defined as in the previous case.

1.3 Equalization concepts

The data signal – which is considered to be of random nature – is passed through an unknown channel. As a result of convolution with this channel and of multi-user character of the transmission environment, the received signal is distorted not only by additive noise $w(t)$ but also by Inter Symbol Interference (ISI) and Multiple Access Interference (MAI).

The goal of an equalizer is to remove the distortion and to decrease the symbol rate error. In the case of an ideal equalizer, the output would be equal to the delayed

version of the original input data sequence. Often (also in our case) the equalizer has structure of linear estimator of finite order, described by vector of equalizer coefficients $\mathbf{f}$. This vector is chosen from a set of all acceptable (non trivial) vectors to minimize certain cost function. For later use, we define some basic terms.

Definition 1. Let N_f be the fixed equalizer length. We define an equalizer space F as the linear space of all equalizers with length N_f.

Definition 2. For equalizer $\mathbf{f}$, we define channel-equalizer system response in the absence of noise as a system response $\mathbf{q}$, and define system space as the set of all possible system responses of equalizers in the equalizer space F.

From the vector valued observation sequence $\{\mathbf{r}_n\}$, the receiver generates a sequence of linear estimates y_n of $s_{n-\nu}$, where ν is an unknown delay. Using the equalizer coefficient vector $\mathbf{f}_n$, the estimates y_n are formed as:

$$y_n = \sum_{i=0}^{N_f-1} f_i^H r_{n-i} = \mathbf{f}^H \mathbf{r}_n \tag{1.15}$$

Assuming that $f(z)$ is BIBO stable the impulse response coefficients of the channel-equalizer cascade $q_n(z) = f^H(z)h(z)$ can be written as:

$$q_n = \sum_{i=0}^{N_f-1} f_i^H h_{n-i} \tag{1.16}$$

Using this formula and the definition of the system response (in the sense of Definition 2), it is possible to rewrite the output of linear estimator in the form:

$$y_n = \sum_i q_i s_{n-i} = \mathbf{q}^T \mathbf{s}_n \tag{1.17}$$

An ideal equalizer, as discussed before, is called the Zero-Forcing equalizer and is described by the following definition:

Definition 3. A zero forcing (ZF) solution for a particular channel $\mathbf{h}$ is an equalizer $\mathbf{f}$ who's system response is pure delay, i.e., $\mathbf{q} = [0 \dots 0\ 1\ 0 \dots 0] = \mathbf{e}_\nu$

Algorithms minimizing the symbol error ratio directly (algorithms based on the Viterbi alg.) have unfortunately high computational complexity. This represents severe limitation for implementation, mainly because of the processing time required for adaption and/or decoding. Because of that, the data throughput is significantly limited.

The symbol error ratio can be decreased indirectly by using some statistical criterion to adapt the equalizer. Under certain assumptions on the properties of the additive noise, Wiener receiver, which minimizes the mean square error (MSE), gives local extreme of maximum likelihood (ML) and maximum a-posteriori probability (MAP) criterion. This is the reason for adopting MSE criterion as a performance measure.

Definition 4. A minimum mean squared error (MMSE) solution for delay ν is an equalizer whose mean squared error between the output and the source signal delayed by νT is minimized over $\mathcal{F}$

$$\mathbf{f}_{\mathrm{MMSE}} = \arg\min_{f\in\mathcal{F}} J_{\mathrm{MMSE}}(\mathbf{f}) \tag{1.18}$$

$$J_{\mathrm{MMSE}}(\mathbf{f}) = E\left(\|e_n\|^2\right) = E\left(\|y - s_\nu\|^2\right) \tag{1.19}$$

The ZF and Wiener solution for the Baud rate and fractionally spaced systems for single user scenario will be shown in the sequel.

1.3.1 Baud Spaced Equalization

If the received signal is sampled at symbol interval T and is received by only one antenna the system is referred to as baud-spaced – this scenario corresponds to the SISO model. The sampled received signal r_n has the form of convolution sum (1.10) symbolically written as $r_n = \mathbf{s}_n \star \mathbf{h}$, where $\star$ denotes convolution. The global system response $\mathbf{q}$ is created as a cascade of two linear systems and can be expressed as a convolution of its impulse responses, i.e., $\mathbf{q} = \mathbf{h} \star \mathbf{f}$. This convolution can be

represented as an inner product of the channel convolution matrix H associated with impulse response vector $\mathbf{h}$ and the equalizer vector $\mathbf{f}$

$$\mathbf{q} = H\mathbf{f} \tag{1.20}$$

For the channel impulse response of length N_h the channel convolution matrix H has the Toeplitz structure:

$$H = \begin{bmatrix} h_0 & & & 0 \\ h_1 & h_0 & & \\ \vdots & h_1 & \ddots & \\ h_{N_h} & \vdots & \ddots & h_0 \\ & h_{N_h} & \ddots & h_1 \\ & & \ddots & \vdots \\ 0 & & & h_{N_h} \end{bmatrix}$$

Using the system response $\mathbf{q}$, The equalizer output can be expressed as:

$$y_n = \mathbf{f}_n^T \mathbf{r}_n = \mathbf{f}_n^T H^T \mathbf{s}_n = \mathbf{q}_n^T \mathbf{s}_n$$

Assuming that there is no noise, the result of the ZF solution will be such vector of equalizer coefficients $\mathbf{f}$ that the system impulse response is pure delay, $q = \delta_i$. In the Z-domain, $q(z) = h(z)f(z) = z^{-\nu}$ for an integer ν. For a finite length equalizer, the system of equations (1.20) is always overdetermined with respect to the equalizer coefficients $\mathbf{f}$. Obviosly, for the baud-spaced case, there is no polynomial $f(z)$ that satisfies the equality for a non-trivial polynomial $h(z)$. Hence, for a BS system there does not exist a ZF solution.

For the design of the MMSE solution, let us first assume that there is a zero mean additive white Gaussian noise (AWGN) with variance σ_w^2 present. The MMSE solution is given by pseudo-inversion of the matrix H in (1.20) or by location of the extreme points of the same formula by means of gradient and Hessian calculation or by simple completion to the square of (1.19). From (1.17), the expression for error

signal is:

$$e_n = \mathbf{q}^T\mathbf{s}_n + \mathbf{f}^T\mathbf{w}_n - \mathbf{e}_\nu^T\mathbf{s}_n = \mathbf{s}_n^T\left(H\mathbf{f} - \mathbf{e}_\nu\right) + \mathbf{w}^T\mathbf{f}$$

Under the above mentioned assumption about noise and the assumption that the source signal is an i.i.d. process jointly uncorrelated with variance σ_s^2, the MSE criterion (1.19), normalized to signal variance, reads:

$$J_{\mathrm{MSE}}(\mathbf{f}) = (H\mathbf{f} - \mathbf{e}_\nu)^H(H\mathbf{f} - \mathbf{e}_\nu) + \mathbf{f}^H\mathbf{f}\frac{\sigma_w^2}{\sigma_s^2} \tag{1.21}$$

This formula indicates that the equalizer coefficient $\mathbf{f}_{\mathrm{MMSE}}$ minimizing J_{MSE} for system delay ν is:

$$\mathbf{f}_{\mathrm{MMSE}} = (H^HH + \frac{\sigma_w^2}{\sigma_s^2}\mathbf{I})^{-1}H^H\mathbf{e}_\nu \tag{1.22}$$

It follows that for $\mathbf{f}_{\mathrm{MMSE}}$, residual MSE error is present and it is a function of the system delay $\mathbf{e}_\nu$

$$J_{\mathrm{MSE}}(f_{\mathrm{MSE}}, \nu) = \mathbf{e}_\nu^H\left(\mathbf{I} - H(H^HH + \frac{\sigma_w^2}{\sigma_s^2}\mathbf{I})^{-1}H^H\right)\mathbf{e}_\nu \tag{1.23}$$

To summarize, the MMSE solution represents a compromise between zeroing the ISI of system response and enhancing the noise from an equalizer setting which has high noise gain. In the absence of noise, the MMSE solution approximates a delayed inverse of the channel. Channels with zeros near the unit circle have a relatively long inverse. It is known that these channels therefore pose conditioning problems to the baud-spaced equalizer design.

1.3.2 Fractionally Spaced Equalization

The discrete time multi-rate channel model from Figure 1.4 is obtained by uniformly sampling $r(t)$ at an integer fraction of the symbol period T_s/P. Sampling at the rate higher than T is represented by the oversampling block at the input of the multi-rate system model in Figure 1.5. After equalization, decimation is needed at the output of equalizer to generate T-spaced symbol estimates.

If the temporal spreads of the T-spaced and the fractionally-spaced equalizer (FSE) are held to the same value, the FSE obviously consumes more computation

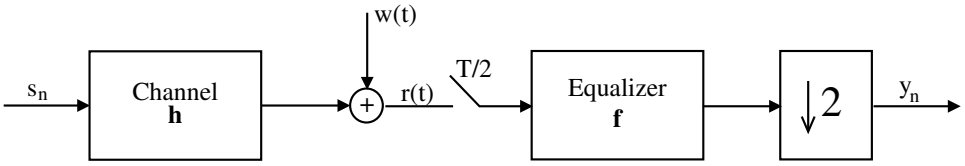


Figure 1.4: Baseband FS system model

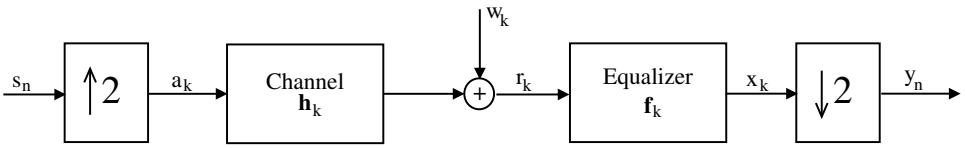


Figure 1.5: FS multi-rate model

than a T-spaced design. So why the fractionally spaced sampling? Even though it requires more computations, it simplifies the rest of the demodulator design and allows to enhance performance of the demodulator.

The actual input ratio to the FSE is usually governed by a variety of hardware considerations. The rate must be high enough to satisfy the sampling theorem, but lowering the rate reduces the computational requirements. The most common choice is to sample the input signal at $2f_B$, making the filter tap spacing equal to $T/2$. For simplicity and without loss of generality, we will solve all equations for $T/2$-spaced equalization. All result are extensible to the more general T/P case.

The received analog signal $r(t)$ is over-sampled at rate $T/2$, so the discrete fractionally-spaced received signal has the form:

$$r_k = r(k\frac{T}{2}) = \sum_{i=0}^{N_{h_{T/2}}-1} s_i h_{T/2}(k\frac{T}{2} - iT) \tag{1.24}$$

where $h_{T/2} = [h(0)\ h(T/2)\ h(T)\ h(3/2T)\ \ldots\ h((N_{h_{T/2}}-1)T)]$ is $T/2$ sampled discrete channel impulse response and k is $T/2$ spaced discrete time. We can express equation

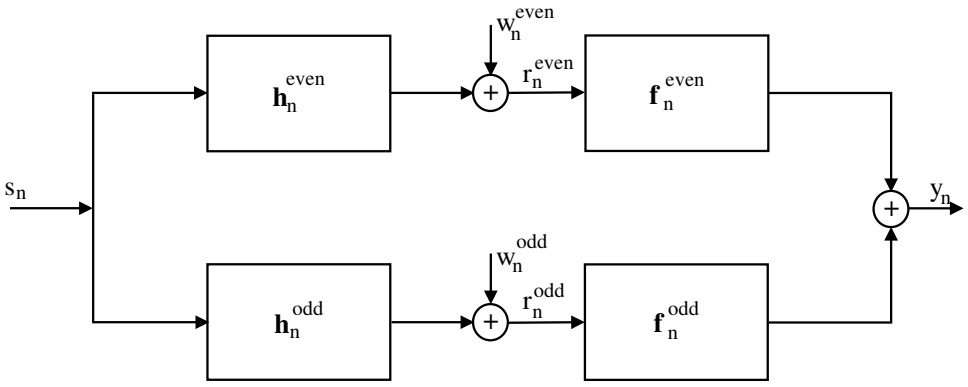


Figure 1.6: FS Multi-Channel model for T/2 spacing

(1.24) for the even and odd samples (sampled at nT and $nT + T/2$, respectively) as:

$$r_{2k} = \sum_{i=0}^{(N_{h_{T/2}}-1)/2} s_i h_{2k-2i} = \mathbf{s} \star \mathbf{h}^{even} = r_n^{even} \tag{1.25}$$

$$r_{2k+1} = \sum_{i=0}^{(N_{h_{T/1}}-1)/2} s_i h_{2k-1-i} = \mathbf{s} \star \mathbf{h}^{odd} = r_n^{odd} \tag{1.26}$$

Note that the even samples are also called on-baud and the odd ones the fractionally-spaced samples.

It follows directly from (1.24) and (1.15) that the equalizer output before decimation is:

$$y_k = \sum_{i=0}^{N_f-1} f_i r_{k-i} \tag{1.27}$$

Let us assume that decimation will throw away all odd samples. The decimated equalizer output can then be expressed by use of even and odd input samples as:

$$\begin{aligned} y_n = y_{2k} &= \sum_{i=0}^{(N_f-1)/2} f_{2i} r_{2k-2i} + f_{2i-1} r_{2k-1-2i} \\ &= \mathbf{f}^{even} \star \mathbf{r}_n^{even} + \mathbf{f}^{odd} \star \mathbf{r}_n^{odd} \end{aligned} \tag{1.28}$$

This shows equivalence between the multi-rate model from Figure 1.5 and the

multichannel/filter bank symbol-rate model given in Figure 1.6, with system impulse response given by:

$$\mathbf{q} = \mathbf{h}_{even} \star \mathbf{f}_{even} + \mathbf{h}_{odd} \star \mathbf{f}_{odd}$$

In that case, to find the zero forcing solution, we solve the following equation:

$$\mathbf{h}^{even} \star \mathbf{f}^{even} + \mathbf{h}^{odd} \star \mathbf{f}^{odd} = \delta_{\nu} \tag{1.29}$$

which is equivalent in the z-transform to the equation

$$h_{even}(z) \star f_{even}(z) + h_{odd}(z) \star f_{odd}(z) = z^{-\nu} \tag{1.30}$$

The above equation is known as the Bezout equation and has unique solution $f(z)$ when h^{even} and h^{odd} do not share any common roots and when deg $f(z) =$ deg $h(z)-1$. When the degree condition is not satisfied,we may have multipe solutions (deg $f(z) =$ deg $h(z)$) or no solution (deg $f(z) <$ deg $h(z)$). To express the ZF solution in the close form, we have to derive the structure of the channel convolution matrix H. The system impulse response $\mathbf{q}$ can be expressed as:

$$\mathbf{q} = H\mathbf{f} = \left[H^{even} H^{odd}\right] \begin{bmatrix} \mathbf{f}^{even} \\ \mathbf{f}^{odd} \end{bmatrix} \tag{1.31}$$

where H^{even} and H^{odd} are convolution matrices corresponding to $\mathbf{h}^{even}$ and $\mathbf{h}^{odd}$. Rearranging $\mathbf{f}$ to vector with separated even and odd coefficients represents a basis permutation. Hence, we can apply inverse permutation on the compound matrix $\left[H^{even} H^{odd}\right]$ to obtain the structure of the channel convolution matrix H from (1.31). The resulting matrix will have the following form:

$$H = \begin{bmatrix} h_1 & h_0 & & & & & 0 \\ h_3 & h_2 & h_1 & h_0 & & & \\ \vdots & \vdots & h_3 & h_2 & \ddots & & \\ h_{N_{h_{T/2}}-2} & h_{N_{h_{T/2}}-1} & \vdots & \vdots & \ddots & h_1 & h_0 \\ & & h_{N_{h_{T/2}}-1} & h_{N_{h_{T/2}}-2} & \ddots & \vdots & \vdots \\ 0 & & & & \ddots & h_{N_{h_{T/2}}-1} & h_{N_{h_{T/2}}-2} \end{bmatrix} \tag{1.32}$$

Equation (1.31) has solution for $\mathbf{q} = \mathbf{e}_\nu$ if the convolution matrix H has full row rank. This requirement implies that matrix H has as many columns as rows (in other words, that it is a square matrix). That gives us constraint on the length of equalizer $N_f = N_{h_{T/2}} - 2$ for even $N_{h_{T/2}}$ (resp. $N_f = N_{h_{T/2}} - 1$ for odd $N_{h_{T/2}}$). If these conditions are satisfied the convolution matrix H is invertible and the ZF solution for delay ν has the form:

$$f_{\mathrm{ZF}} = H^{-1}\mathbf{e}_\nu \tag{1.33}$$

When the channle length is not known, the MMSE solution for the FS-equalizer can be derived analogously to that for the baud-rate system. As before, there is no constraint on the convolution matrix H nor on the equalizer length N_f. For delay ν, the zero-forcing solution has the form:

$$f_{\mathrm{MMSE}} = (H^H H + \frac{\sigma_w^2}{\sigma_s^2} I)^{-1} H^H \mathbf{e}_\nu \tag{1.34}$$

Note that in the absence of noise, the MMSE solution corresponds to the ZF solution, when H is invertible.

1.4 Higher Order Statistic

In many cases, "standard" second order statistical properties are not sufficient for successful blind deconvolution. There is a group of algorithms that uses higher order statistics to solve this problem. In chapter 2.3 we present an algorithm developed by Shalvi and Weinstein which uses maximization of a higher order cumulant (kurtosis) to solve the criterion. In this chapter, we present a short review of higher order statistics and of the definition of cumulants.

Stationary random time series can be viewed as being generated by a sequence of statistically independent samples of an underlying generating random variable. Any random variable X is mathematically described by its probability density function (pdf) $f_X(x)$. This pdf can be estimated from the set of samples of the random variable

$\{x_1, x_2 \ldots x_n\}$. In many cases the pdf of the random variable can be characterized by a few statistical parameters such as moments and cumulants. The n^{th} order moment of random variable X is specified by

$$E\{X^n\} = \int_{-\infty}^{+\infty} x^n f_X(x) dx \qquad \text{for} \quad n = 1, 2, 3, \ldots \tag{1.35}$$

where $E\{\}$ denotes the expectation operator.

If the n^{th} order moment exists (i.e., is finite), it follows, then all moments of order smaller than n also exist. Well known and widely used is the first order moment, which is known as mean value of a random variable X and is often denoted by m_x.

The n^{th} central moment of random variable X is defined as:

$$\mu_n = E\{[X - m_x]^n\} = \int_{-\infty}^{+\infty} (x - m_x)^n f_X(x) dx \tag{1.36}$$

Moments and central moments are identical for zero mean random process i.e., if $m_x = 0$. The second order central moment is called variance (referred to as σ^2) and is a measure of how dispersed is the mass around its mean.

The moment generating function of the random variable X is defined as Fourier transform of complex conjugate of the probability density function and it is formally given by

$$\Phi_X(\omega) = \int e^{j\omega x} f_X(x) dx = E\{e^{j\omega x}\} \tag{1.37}$$

The moment generating function possesses all the properties of Fourier transform. For linear systems, three of its properties are of particular importance:

$$\begin{array}{rclcrcl} Y & = & X + a & \Longleftrightarrow & \Phi_Y(\omega) & = & e^{j\omega a}\Phi_X(\omega) \\ Y & = & aX & \Longleftrightarrow & \Phi_Y(\omega) & = & \Phi_X(a\omega) \\ Y & = & X_1 + X_2 & \Longleftrightarrow & \Phi_Y(\omega) & = & \Phi_{X_1}(\omega)\Phi_{X_2}(\omega) \end{array} \tag{1.38}$$

where a is a scalar constant and X_1 and X_2 are independent random variables. $\Phi(\omega)$ possesses some additional properties due to basic nature of the probability density function. Since probability density function has unit area and is real it follows that

$$\begin{array}{rcl} \Phi_X(0) & = & 1 \\ |\Phi_X(\omega)| & \leq & 1 \\ \Phi_X^*(\omega) & = & \Phi(-\omega) \end{array}$$

Using the moment generating function, the n^{th} moment is defined by the Taylor series expansion around the origin $\omega = 0$ as:

$$\frac{d^n \Phi_X(\omega)}{d\omega^n}|_{\omega=0} = (j)^n E\left\{X^n\right\}$$

For analysis of linear combination of statistically independent random variables, natural logarithm of the moment generating function is widely used. This logarithm is called the cumulant generating function and is defined as:

$$\Psi_X(\omega) = \ln(\Phi_X(\omega)) = \ln\left(E\left[e^{j\omega x}\right]\right) \tag{1.39}$$

The properties of the cumulant generating function directly follow from the properties defined in (1.38):

$$\begin{array}{rclcrcl} Y &=& X + a & \Longleftrightarrow & \Psi_Y(\omega) &=& j\omega a + \Psi_X(\omega) \\ Y &=& aX & \Longleftrightarrow & \Psi_Y(\omega) &=& \Psi_X(a\omega) \\ Y &=& X_1 + X_2 & \Longleftrightarrow & \Psi_Y(\omega) &=& \Psi_{X_1}(\omega) + \Psi_{X_2}(\omega) \end{array} \tag{1.40}$$

where a, X_1 and X_2 are defined as before. The n^{th} order cumulant is then defined similarly as in the case of moments by the Taylor series expansion around the origin $\omega = 0$:

$$\mathrm{cum}_n(x) = (-j)^n \frac{d^n \Psi_X(\omega)}{d\omega^n}|_{\omega=0} \tag{1.41}$$

Unfortunately, cumulants cannot be estimated directly by summation or integration operation on the random sequence samples. To find them, it is necessary either to derive them from the probability density function or to estimate the moments first and calculate them using the cumulant-moment relation. Cumulants and moments (resp. central moments) are clearly related as is seen by use of pdf function in their definition. For a zero-mean pdf the three central moments and the corresponding cumulants are identical, while, starting from the fourth order, they differ. In the sequel, we express the first four cumulants as a function of central moments (valid

not only for the zero mean distributions):

$$\begin{aligned} \mathrm{cum}_1(x) &= m_x = \mu_1 \\ \mathrm{cum}_2(x) &= \sigma^2 = \mu_2 \\ \mathrm{cum}_3(x) &= \mu_3 \\ \mathrm{cum}_4(x) &= \mu_4 - 3\mu_2^2 \end{aligned}$$

Note: Contrary to moments, cumulants (except of $\mathrm{cum}_1(x)$) are invariant to the shift of origin (offset).

Note: For a zero mean Gaussian random variable all cumulants but $\mathrm{cum}_2(x)$ are zero.

$$\mathrm{cum}_2(y_i) = E(y_i^2) = \sigma^2 \qquad \mathrm{cum}_4(y_i) = E(y_i^4) - 3E(y_i^2)^2 \tag{1.42}$$

In a particular case, random variable Y can be expressed as a combination of k random, statistically independent variables of the form:

$$Y = \sum_{i=0}^{k-1} a_i X_i$$

Because of statistical independence of X_i, the moment generating function of Y is equal to the product of the scaled moment generating functions of variables X_i:

$$\Phi_Y(\omega) = \prod_{i=0}^{k-1} \Phi_{X_i}(a_i\omega)$$

where the scaling a_i of $\Phi_{X_i}(a_i\omega)$ follows directly from the properties defined in (1.38). Now, applying a natural logarithm on the moment generating function of Y, we obtain directly the expression for the cumulant generating function:

$$\Psi_Y(\omega) = \sum_{i=0}^{k-1} \Psi_{X_i}(a_i\omega) \tag{1.43}$$

From the cumulant definition (1.41) the n^{th} order cumulant is defined as the n^{th} derivative of the cumulant generating function at origin $\omega = 0$. It follows directly

(using differentiation) that for the n^{th} order cumulant of random variable Y which is a linear combination of k statistically independent random variables it holds:

$$\mathrm{cum}_n(y) = \sum_{i=0}^{k-1} a_i^n \, \mathrm{cum}_n(x^{(i)}) \tag{1.44}$$

where $\mathrm{cum}_n(x^{(i)})$ is the n^{th} order cumulant of a random variable X_i

1.4.1 Cumulants and linear system response

In the signal processing context, the received signal is often expressed as the response of a linear system to an independent and identicaly distributed input signal. Let us assume a linear time-invariant system such as in Figure 1.2, with zero noise. The output variable y_n is represented by a convolution sum as:

$$y_n = \sum_i h_i x_{n-i}$$

The output sequence $\{y_i\}$ is a sequence of generally dependent samples of the random variable Y. The nature of this dependency is dictated by a convolution sum of the input signal and the system impulse response h_i. From the results of (1.43) and (1.44) it follows that for the output signal of the linear system the cumulant generating function has the form:

$$\Psi_Y(\omega) = \sum_i \Psi_X(h_i \omega) \tag{1.45}$$

where $\Psi_X(\omega)$ is the cumulant generating function of the input signal and h_i is i^{th} sample of the discrete impulse response. This expression shows that the cumulant generating function $\Psi_Y(\omega)$ is dependent on the impulse response of a linear system $\mathbf{h}$. In the sequel, for nth order cumulant of random variable Y it states:

$$\mathrm{cum}_n(y) = \mathrm{cum}_n(x) \sum_i h_i^n \tag{1.46}$$

The equation (1.46) shows that if $\mathrm{cum}_n(x)$ is finite, $\mathrm{cum}_n(y)$ is also finite. For even values of n, the equation can be rewritten as $\mathrm{cum}_n(y) = \mathrm{cum}_n(x) \, \|\mathbf{h}\|_n^n$, and, since any

non-trivial linear system $\mathbf{h}$ has positive n-norm, it follows that any non-zero even-order cumulant cannot be zeroed by any non-trivial linear operation. Moreover, the resulting cumulant will share the sign of the excited cumulant. This properties are not valid for odd order cumulants.

In the equalization context, cumulants are used to select the desired system response $\mathbf{q} = a \cdot \mathbf{e}_\nu$. This means that the generating function of excitation and the response time series are scalar multiples so that $Y = aX$ with some non-zero scalar a. It follows from the cumulant generating function properties (1.40) that $\Psi_Y(\omega) = \Psi_X(a\omega)$ and that

$$\mathrm{cum}_n(y) = \|\mathbf{h}\|_n^n \,\mathrm{cum}_n(x) = |a|^n \,\mathrm{cum}_n(x) \qquad \forall\, n \text{ such that } \mathrm{cum}_n(x) \neq 0 \tag{1.47}$$

Equality $\|\mathbf{h}\|_n^n = |a|^n$ for variable n is met if and only if all but one element of $\mathbf{h}$ are zero, i.e. $\mathbf{h} = a\mathbf{e}_\nu$. This shows that the maximization of $|\mathrm{cum}_n(x)|$ corresponds to the ZF solution (up to a multiplication by constant).

Chapter 2

Blind Equalization

Traditionally, communication systems have utilized adaptive equalizers with training sequences. By training sequence we mean a sequence of symbols known in advance to the receiver. This training sequence is designed in such way that it can be easily identified by the receiver and it has a form of a synchronization burst. The equalizer with a training sequence works in two modes: the adaptation mode, when the equalizer coefficient are updated using the knowledge of the training sequence and the received signal or equalization mode, when the data is filtered and sent to the decision device. It is evident that sending the training sequence decreases the throughput of a communication system (in the GSM system, to give an example, it is approximately 20% of the data packet size).

Blind equalization, on contrary, is based on a-priori knowledge of some signal properties and it does not utilize a training sequence.

There exist several types of blind equalization algorithms. In the following text, we will give a short review of each of them. This will be followed by a comparison of the two most popular types, the Cumulant based algorithms (mainly Shalvi-Weinstein) and of the Constant Modulus algorithm (also known as the Godard Algorithm).

The following types of the blind equalization algorithms can be distinguished:

- *Decision directed LMS*: the least mean squares (LMS) algorithm is perhaps the

most popular one for non-blind adaptive equalization. The reason of its popularity is that it is simple to implement and that most people are well familiar with this algorithm. The "classical" LMS needs a training signal. However, Lucky [38] showed that if the estimation error is sufficiently small so that the decisions are correct "most of the time", the desired signal can by replaced by the output of a decision-making device (symbol estimate of s_n). It is evident that the DD-LMS is useful only in systems with low value of SNR and ISI.

- *Second Order Statistic (SOS)*: the SOS-based blind algorithms use the knowledge of the covariance or auto-covariance matrix of the transmitted signal to adapt the filter coefficients. In a baud-spaced case, the SOS of the output signal are in general insufficient to identify the phase of a linear time invariant system. For this reason, many traditional blind algorithms use higher order moments of the received signal. However, if fractional sampling is used, the received signal exhibits cyclostationarity, which could be used to resolve the phase of an unknown channel model using purely the statistics of the received signal. Tong et al. developed numerous blind indirect equalization algorithms based on estimation of the autocorrelation function of the received signal (see [65], [66], [64]). Moulines et al. [39] exploited the orthogonality of signal and noise subspaces of the received correlation matrix. Nevertheless, most of these algorithms are computationally expensive in actualization of the correlation matrix and/or in matrix decomposition (QR, SVD etc.). It has been shown recently that the SOS-based blind equalization algorithms converge with fewer number of observations than the HOS-based estimation techniques. Hence, an algorithm based on SOS should provide convergence rate benefits over the algorithm based on HOS.

- *Joint channel and data estimation*: These algorithms are based on the *maximum likelihood* (ML) criterion. The ML estimates of channel impulse response h and transmitted data sequence s are the estimates that maximize the joint probability density function $p(r|h, s)$. If the data sequence s is known, the ML

estimation of the channel parameters can be expressed as $h_{ML} = (\mathcal{S}\mathcal{S}^T)^{-1}\mathcal{S}^T r$. $\mathcal{S}$ is the input data matrix with similar structure as in (3.21). If the impulse response h is known, the optimal ML detection of the given data sequence s is performed by trellis search using the well known Viterbi algorithm. When neither s nor h are known, the ML estimation uses averaging over the data sequence. This results in an iterative estimation of h_{ML} and symbol detection using the Viterbi algorithm. This algorithm has two major drawbacks: firstly, the recursive estimation of h_{ML} is computationally intensive, and secondly, the estimate of h is often not sufficiently good. These methods were studied in many papers – good examples are [1], [30] and [55].

- *Bussgang technique*: This class of blind equalization algorithms is based on the stochastic-gradient iterative procedure. To generate the desired signal some (in most cases memoryless) nonlinearity is applied to the output of a linear equalizer. This class of algorithms is used mainly for an input data sequence with non-Gaussian probability distribution function. These algorithms then differ mainly in the choice of the nonlinear function. They include the algorithms proposed by Godard [16](the well-known CMA) and Sato [48]. The equalizer coefficient tracking is provided by an adaptive LMS-type (or steepest descent) algorithm. From this selection of algorithm follow the main advantages and drawbacks of the method: the LMS-type algorithms are very simple to implement. On the other hand, the convergence characteristics depend on the autocorrelation function of the received data and the convergence is relatively slow on average. The second drawback follows from the fact that we minimize a non-convex cost function. As a consequence, the global convergence is not guaranteed.

In the sequel, the perfect equalization conditions are defined and four HOS equalization techniques are presented, namely the Godard criterion, the cumulant-based criterion and the mutual information concept. Finally, their equivalence and/or relation is discussed in separate sections.

2.1 Perfect Blind Equalization condition

In the context of blind equalization, we define Perfect Equalization (PE) as the zero-forcing solution (see Definition 3), i.e. the signal is perfectly equalized when $y_n = s_{n-\nu}$. There are several assumptions under which the HOS algorithms are proved to converge. These assumptions are related to the source statistics and channel characteristics and are formalized as follows:

- Each source signal $\{s^l\}$ is an independent identically distributed zero mean sequence with variance σ^2.
- Each $\{s^l\}$ is sub-Gaussian with normalized kurtosis $\rho < 3$ in real valued case.
- All sources are mutually independent
- There is no noise at the channel output
- The channel convolution matrix $\mathbf{H}$ is of full column rank

In the PE scenario, there is no additive noise at the channel output. In the case when the algorithms are studied in the noisy environment, there are the following assumptions about noise:

- Sources are independent from the noise vector w_n
- Noise is zero mean, Gaussian process with covariance matrix $E[w_n w_n^H] = \sigma^2 I$

2.2 Godard Criterion

The Constant Modulus (CM) criterion is probably the most popular and best studied criterion in the design of blind equalizers/beamformers for communication systems with sub-Gaussian i.i.d. sources. Thanks to its popularity, it has been adopted for implementation of the CM-based beamformers for HDTV. Historically, there are two basic works giving the basis of CM approach: Godard proposed his criterion [16]

in 1980; similar results were independently derived by Traichler and Agee in 1983 [71]. In contrast to the MSE criterion, Godard proposed a criterion which penalizes deviation in magnitude of the equalizer output from a fixed value:

$$J_p = \frac{1}{2p} E\left[\left(|y_n|^p - \gamma\right)^2\right] \;; \qquad p > 1 \tag{2.1}$$

where γ is chosen as a function of the source alphabet and the integer p is user-chosen. Specifically, Godard showed that for the particular case of sub-Gaussian sources, J_p is minimized by system response generating zero ISI, that choice of $\gamma = E\left[|s_n|^{2p}\right] / E\left[|s_n|^p\right]$ ensures existence of local minima of J_p and that these minima correspond to the perfectly equalized system responses. Choosing $p = 2$, the cost function (2.1) becomes the same as that used by Treihler and Agee in [71] and is known as the Constant Modulus (CM) criterion. In sequel we denote it as J_{CM}:

$$J_{CM} = \frac{1}{4} E\left[\left(|y_n|^2 - \gamma\right)^2\right] = \frac{1}{4} E\left[\left(\left|\mathbf{f}^T \mathbf{r_n}\right|^2 - \gamma\right)^2\right] \tag{2.2}$$

where $\gamma = \frac{E\left[|s_n|^4\right]}{E\left[|s_n|^2\right]^2}$.

For CM sources that meet the PE conditions, any ZF solution is, in the same time, a CM solution. Zeng et al. [81] have shown the relation between the CM and Wiener solution and that some of the CM solutions are located near the MMSE solution with low MSE. As will be described in detail in Chapter 3, the CM solution is not unique because the CM cost function is multimodal. When the PE conditions are not fully satisfied (for example, the properties of the channel matrix are different) the CM solutions are not identical with respect to the residual MSE. Depending on the value of residual MSE, we distinguish so called *good* and *bad* CM solution. To avoid that the CM solution is *bad*, proper initialization is needed.

As will be discussed later, from a implementation point of view, the main advantage of CMA is the existence of a simple gradient descent algorithm of a similar form as the popular LMS. More details about CM algorithm will be given in Chapter 3.

2.3 Cumulant based criterion

The use of cumulants as a criterion for blind deconvolution and/or source separation of non-Gaussian data is a natural consequence of the properties presented in Chapter 1.4.1. Development of the HOS blind algorithms has very long history and one of the early works dates back to 1958. To use cumulants as a measure of non-gaussianity is a natural choice since for a Gaussian random variable, all cumulants of order > 2 are zero, and this class of algorithms is based on seeking for parameters $\mathbf{f}$ giving the k^{th} order cumulant as far from zero as possible (see 1.46 and 1.47)

$$\mathbf{f}_{opt} = \max_{\mathbf{f}} |\text{cum}_n(y)| \quad \text{where} \quad y_n = \mathbf{f}^T \mathbf{r}_n$$

An important group of the cumulant-based criteria uses the fourth order cumulant named kurtosis (κ_y) or normalized kurtosis (ρ_y).

In 1978 Wiggins [77] defined the Minimum Entropy Deconvolution problem and proposed an objective function, which maximizes the normalized kurtosis of the output. Gray showed in 1979 [18] that the proposed objective function is just a special case of the general criterion of the form

$$J_{s,r} = \frac{E\left[|y_n|^r\right]}{\left(E\left[|y_n|^s\right]\right)^{r/s}} \equiv \frac{\text{cum}_r(y_n)}{\left[\text{cum}_s(y_n)\right]^{r/s}} \tag{2.3}$$

where r and s are nonnegative integers and $r + s \geq 3$. This criterion is evaluated in terms of the non-normalized cumulants of order r and s. In practical situations, parameter r is chosen to be 3 or 4. This is because the larger cumulant order used, the larger the variance of the designed equalizer and the computational complexity.

Donoho [10] used minimum entropy deconvolution approach and partial order of non-Gaussianity and first analyzed the properties of constrained kurtosis cost function as a special case of (2.3) i.e. with $s = 2$, $r = 4$ and its relations to the MSE criterion. His approach is based on the properties (1.46) and (1.47) of linear combination of the random variables. The same result was independently derived by Shalvi and Weinstein in [56], where they have shown the existence of perfect blind equalizer using (2.4) for independent identically distributed sources, invertible channels and no

additive noise.

$$J_{SW} = \frac{\mathrm{cum}_4(y_n)}{[\mathrm{cum}_2(y_n)]^2} \tag{2.4}$$

The cumulant based criteria are also known in the literature as a family of contrast functions $f_{2p}(f) = \left(\frac{\|f\|_{2p}}{\|f\|_p}\right)^{2p}$, which directly follows from the properties of normalized cumulants applied to linear time invariant systems. The proposed criteria are universal in the sense that they do not impose any other restriction on the input probability distribution (only non-Gaussianity of sources) and can therefore be applied to a wide range of deconvolution problems. There are no spurious local maxima that correspond to unwanted local solutions. This implies that the gradient search procedure solving the constrained maximization will converge to the desired solution regardless of initialization. On the other hand, the criterion is flat in the vicinity of its saddle points which may result in slow convergence of the gradient search algorithm. Under practical condition of finite data, additive noise and finite equalizer length, these saddle points may turn to local maxima. It was shown that the maxima points correspond to the ZF solution with different delay ν and up to complex valued gain factor.

Criteria based on the normalized cumulants are scale invariant since

$$J(\beta\mathbf{f}) = \left(\frac{\|\beta\mathbf{f}\|_{2p}}{\|\beta\mathbf{f}\|_p}\right)^{2p} = \frac{\beta^{2p}}{\beta^{2p}}\left(\frac{\|\mathbf{f}\|_{2p}}{\|\mathbf{f}\|_p}\right)^{2p} = J(\mathbf{f})$$

holds for any nonzero scalar β. It should be noted that the normalized cumulant criterion is immune to the additive Gaussian noise since $\mathrm{cum}_n(x_{gauss}) = 0$ for any $n > 2$. Also, since it preserves system phase, it can be used for non-minimum phase systems. Many papers [45] [12] [11] show the existence of the perfect blind equalization under the PE conditions and the influence of violation of the PE conditions on the estimator performance [54] [46].

2.4 Independent Component analysis and Fast fixed-point algorithm

Independent component analysis is a statistical technique based on the mutual information approach/minimization with the main applications in the domains of the blind source separation, blind deconvolution/equalization and feature extraction. It represents an independent research area in statistical signal processing. We mention it in our review because under some assumptions this method is closely related to the HOS methods and some results can be used directly in other HOS algorithms.

First works on the ICA concept were published by Jutten and Herault in [27] and by Comon in [7]. It is shown in [7] how to obtain a general formulation of ICA without making assumptions on the underlying data model, only using the mutual information as a natural measure of dependence between random variables. To define mutual information, let us start with the definition of the differential entropy H as

$$H(y) = -\int f(\mathbf{y}) \log(f(\mathbf{y}))d\mathbf{y} \tag{2.5}$$

where $\mathbf{y}$ is a random vector with probability density function $f(\mathbf{y})$. The differential entropy can be normalized to give rise to the definition of the negentropy which is invariant to any linear transformation. The negentropy $J(\mathbf{y})$ is defined as

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{2.6}$$

where $\mathbf{y}_{gauss}$ is a Gaussian random variable with the same covariance as $\mathbf{y}$. It can be seen from (2.6) that negentropy can be also interpreted as a meassure of non-gaussianity of a random process. Using negentropy, one can define mutual information I between the elements y_i of vector $\mathbf{y}$ as

$$I(y_1, y_2, \ldots, y_n) = J(\mathbf{y}) - \sum_i J(y_i) \tag{2.7}$$

Because negentropy is invariant to invertible linear transformation, it is obvious from (2.7) that minimizing the mutual information is equivalent to finding directions in which the negentropy is maximized.

In derivation of the fixed-point ICA algorithm [21] the approximation based on maximum entropy principle of the form

$$J(y) \approx c\left[E\left\{G(y_i)\right\} - E\left\{G(\nu)\right\}\right] \tag{2.8}$$

where G is some non-quadratic function, ν is zero mean unit variance Gaussian variable and c is an irrelevant constant, was used. This gives the new objective function:

$$J(\mathbf{f}) = \left[E\left\{G(\mathbf{f}^T\mathbf{r_n})\right\} - E\left\{G(\nu)\right\}\right] \tag{2.9}$$

where $\mathbf{f}$ is – in our sense – the equalizer tap vector constrained so that $E\left[(\mathbf{f}^T\mathbf{r})^2\right] = 1$. Finally in [21], [22] the fixed point algorithm based on Newtons iterations of the following form was derived.

$$\begin{aligned} \mathbf{f}' &= E\left[\mathbf{r}g(\mathbf{f}^T\mathbf{r})\right] - E\left[g'(\mathbf{f}^T\mathbf{r})\right]\mathbf{f} \\ \mathbf{f} &= \frac{\mathbf{f}'}{\|\mathbf{f}'\|} \end{aligned} \tag{2.10}$$

where $g(\cdot)$ is first derivative of $G(\cdot)$ and $g'(\cdot)$ is its second derivative. For our purpose, the gradient ascent form has higher importance since it could be used to show the relations with the other mentioned algorithms. Is has the form:

$$\begin{aligned} \mathbf{f}' &= \mathbf{f}_k + \mu_k\left[E\left[\mathbf{r}g(\mathbf{f}_k^T\mathbf{r})\right] - \beta_k\mathbf{f}_k\right] \\ \mathbf{f}_{k+1} &= \frac{\mathbf{f}'}{\|\mathbf{f}'\|} \end{aligned} \tag{2.11}$$

where $\beta_k = E\left[\mathbf{r}g(\mathbf{f}_k^T\mathbf{r})\right]^T\mathbf{f}_k$ and μ_k is a step-size parameter.

The choice of nonlinearity G in (2.10) is the key problem in the performance tuning. The algorithm performance for a given nonlinearity G is dependent on the statistical parameters of the source signals. Typical nonlinearities are u^4 (giving raise to the kurtotic type of measure), $\log(\cosh(u))$, $e^{u^2/2}$ and/or $1/(x^2+1)^2$. The main advantage of this algorithm is that it is independent of whether the source is sub-gaussian or super-gaussian and works also for mixed case (i.e. some sources are sub-gaussian other super-gaussian). On the other hand, it assumes that the mixing

matrix $\mathbf{H}$ is square invertible and noise free case only, which – in the real systems – is hardly achieved.

In [47], the robustness of the objective function (2.9) to the undermodeled case (more sources than sensors = non-square mixing matrix) is studied. In [47], Regalia and Kofidis have shown the step-size bounds of iterative procedure (2.11) which ensure monotonic convergence to the local maxima/minima. It solves sub- and super-gaussian sources separately using step descent/ascent procedure, but does not solve the mixed case. It also solves the optimal step size for the sub-Gaussian case.

2.5 Super exponential algorithm

The super-exponential algorithm was originally proposed by Shalvi and Weinstein [57]. It was developed in the system domain to set up the optimal system response and to minimize the ISI but without any criteria and/or objective function requirements on the received data. The key point in the development of the algorithm is that the ideal system response is a ZF solution i.e. $\mathbf{q} = [0 \dots 0\ 1\ 0 \dots 0] = \mathbf{e}_\nu$. Then, if the dominant term q_ν of system response $\mathbf{q}$ is unique, the Hadamard exponent

$$\left(\frac{\mathbf{q}}{q_\nu}\right)^{\odot m}$$

approaches the ZF solution as m tends to the infinity. From this observation, the iterative algorithm proposed in [57] of the form:

$$\mathbf{q}' = \mathbf{q}_i^{\odot p}(\mathbf{q}_i^*)^{\odot q} \tag{2.12}$$

$$\mathbf{q}_{i+1} = \frac{\mathbf{q}'}{\|\mathbf{q}'\|} \tag{2.13}$$

where i is an iteration number and $'$ is used for the intermediate value, converges asymptotically in i to the optimal system response for any $p + q > 1$. The iterative procedure (2.12) converges rapidly because one coefficient approaches the unity while all others decrease very quickly. The computation of this procedure, however, requires the knowledge of the channel impulse response. Since this response is unknown in a

real system, the form (2.12) has mainly theoretical meaning. In [57], an algorithm for adjustable equalizer based on joint cumulants of received data and output of the equalizer was proposed. This algorithm has the following form:

$$\mathbf{f}' = \mathbf{R}_{\mathbf{r}}^{-1}\mathbf{d}_{\mathbf{r}y}^{(i)} \tag{2.14}$$

$$\mathbf{f}_{i+1} = \text{cum}\{y_n : p, y_n^* : q, r_n\} \tag{2.15}$$

where $\mathbf{R}_{\mathbf{r}} = E\left[\mathbf{r}\mathbf{r}^H\right]$ is an $N_f \times N_f$ data covariance matrix of $\mathbf{r}$ and $\mathbf{d}_{\mathbf{r}y}$ is a $1 \times N_f$ joint cumulant vector

$$\mathbf{d}_{\mathbf{r}y} = \text{cum}(y_n : p, y_n^* : q, \mathbf{r}_n)$$

in which p and q are nonnegative integers such as $p + q \geq 3$. Since, in practice, the finite length equalizer is used, the proposed procedure (2.14) is approximative in the super-exponential convergence rate. In practice, only the versions with $p = q = 1$ and $p = 2,\ q = 1$ are used.

The main advantage of this solution is small computational complexity (since $\mathbf{R}_{\mathbf{r}}^{-1}$ is computed only once and the iteration procedure has complexity $\mathcal{O}(L)$) and fast convergence. On the other hand, the algorithm may diverge with finite data length (used for $\mathbf{d}$ and $\mathbf{R}_{\mathbf{r}}$ estimates) and finite SNR. The properties of the super-exponential algorithm under non-optimal conditions (i.e. finite equalizer order and SNR) are studied in [44], [46].

2.6 Relations

Although each of the above mentioned methods uses different criterion and/or concept, under some assumptions they are closely related and create a group of algorithms referred as a family of contrast functions $f_{2p}(q)$ characterized by

$$f_{2p}(q) = \left(\frac{\|q\|_{2p}}{\|q\|_p}\right)^{2p}$$

The basic assumptions of their equivalence are the noise-free channel and sub-Gaussian i.i.d. sources. In this section, we would like to review shortly these equivalences.

2.6.1 Equivalence of the Shalvi-Weinstein and Godard criterion

The close relation between the CMA and SW criterion was first noticed by Li [36] and later rigorously proven by Regalia [42]. The relation can be expected since CMA could be expressed in terms of cumulants:

$$\begin{aligned} J_{CM} &= E\left[\left(|y|^2-\gamma\right)^2\right] = E\left[|y|^4\right] - 2\gamma E\left[|y|^2\right] + \gamma^2 \\ &= \mathrm{cum}_4(y) + 3\sigma_y^2 - 2\gamma\sigma_y^2 + \gamma^2 \end{aligned}$$

where the substitutions $E\left[|y|^4\right] = \mathrm{cum}_4(y) + 3E^2\left[|y|^2\right]$ and $\sigma_y^2 = E\left[|y|^2\right]$ have been used.

To show more rigorous equivalence (following [42]), we will start by expressing the parameter vector $\mathbf{f}$ in the polar form as $y_n = \mathbf{f}^T\mathbf{r}_n = f_r\overline{\mathbf{f}}^T\mathbf{r}_n$, using unit-norm vector $\overline{\mathbf{f}}$ and a radial term f_r. We define Θ as a set of all possible unit norm parameters $\overline{\mathbf{f}}$ and the output of the unit norm filter $\overline{y}_n = \overline{\mathbf{f}}^T\mathbf{r}_n$. Then the CMA criterion can be rewritten in terms of f_r and Θ as:

$$J_{CM}(f_r,\Theta) = E\left[\left(\gamma - f_r^2\left|\overline{\mathbf{f}}\right|^2\right)^2\right] \tag{2.16}$$

The optimal radial term follows directly from (2.16) by setting the derivative of (2.16) with respect to f_r to zero. It is given as $f_{r_{opt}}^2 = \frac{\gamma E\left[|\overline{y}_n|^2\right]}{E\left[|\overline{y}_n|^4\right]}$. Substituting this term back to (2.16) we obtain the CMA cost function of the form

$$\begin{aligned} \overline{J}_{CM}(\Theta) \triangleq J_{CM}(f_r,\Theta)|_{f_r^2=f_{r_{opt}}^2} &= \gamma^2\left(1 - \frac{E^2\left[|\overline{y}_n|^2\right]}{E\left[|\overline{y}_n|^4\right]}\right) \\ &= \gamma^2\left(1 - \frac{1}{2 + \frac{\mathrm{cum}_4(\overline{y}_n)}{E^2\left[|\overline{y}_n|^2\right]} + \frac{E\left[\overline{y}_n^2\right]E[(\overline{y}_n^*)^2]}{E^2\left[|\overline{y}_n|^2\right]}}\right) \end{aligned} \tag{2.17}$$

For real equalizer output, this equation (2.17) is reduced to :

$$\overline{J}_{CM}(\Theta) = \gamma^2\left(1 - \frac{1}{3 + J_{SW}(\Theta)}\right)$$

where the polar decomposition was used in definition of $J_{SW}(\Theta)$. In the case of complex equalizer output, if the output $\overline{y}_n$ is circular for all choices of $\overline{\mathbf{f}}$, then $E\left[\overline{y}_n^2\right] = E\left[(\overline{y}_n^*)^2\right] = 0$ and (2.17) is reduced to :

$$\overline{J}_{CM}(\Theta) = \gamma^2 \left(1 - \frac{1}{2 + J_{SW}(\Theta)}\right)$$

This shows that the spherical error surface $\overline{J}_{CM}$ is a simple deformation of the J_{SW} criterion. Moreover, the deformation is monotonic and order preserving (i.e. $\overline{J}_{CM}(\Theta_1) < \overline{J}_{CM}(\Theta_2) \Longleftrightarrow \overline{J}_{SW}(\Theta_1) < \overline{J}_{SW}(\Theta_2)$). As a consequence of the latter, stationary points in the Θ space coincide as well as their type (local/global minima/ maxima, saddle-point).

This equivalence holds for the noise free case for real equalizer output and for complex circular equalizer output. On the other hand, if the circularity of the complex equalizer output is not satisfied, J_{CM} and J_{SW} can have different convergence points.

Note that CM and SW criteria can still differ in their practical implementation. This is because of two main reasons. First, since their objective functions are related by deformation, the eigenvalue spread of the Hessian matrix in the vicinity of minima/maxima points may differ and their local convergence speed may differ as well. In the presence of noise (channel noise and/or round-off error noise), both methods can lead to misadjustments which do not need to be equivalent.

2.6.2 Super-exponential algorithm and F_{2p} contrast function equivalence

Regalia and Mboub have shown in [44], [45] and [46] that the super-exponential algorithm belongs to the family of contrast functions $f_{2p}(q) = \left(\frac{\|q\|_{2p}}{\|q\|_p}\right)^{2p}$. In the original work, the authors examine the properties of group of blind equalization criteria in the undermodeled and/or noisy environment case. To show that the super-exponential algorithm belongs to this group, with some degree of simplification, we will restrict our discussion to the noise free case only. Furthermore, we assume that the equalizer length is sufficient (which means that the ZF solutions are attainable).

Let us start by expressing the generalized cumulant cost function $f_{2p}(\mathbf{q})$ in the system space as:

$$f_{2p}(\mathbf{q}) = \mathrm{cum}_{2p}(y_n)\frac{\|\mathbf{q}\|_{2p}^{2p}}{(\|\mathbf{q}\|^2)^p} = \mathrm{cum}_{2p}(y_n)\frac{\sum_i q_i^{2p}}{(\sum_i q_i^2)^p}$$

where the subscript i is used for the i^{th} element of the system response vector $\mathbf{q}$. Calculating the gradient of $f_{2p}(\mathbf{q})$ with respect to $\mathbf{q}$, we obtain:

$$\nabla f_{2p}(\mathbf{q}) = 2p\left(\mathrm{cum}_{2p}(y_n)\mathbf{q}^{\odot 2p-1} - f_{2p}(\mathbf{q})\mathbf{q}\right)$$

Using this result, we can directly obtain the gradient search procedure of the form

$$\begin{aligned}
\mathbf{q'}_k &= \mathbf{q}_k \pm \frac{\mu_k}{2p}\nabla f_{2p}(\mathbf{q}_k) \\
&= \mathbf{q}_k \pm \mu_k\left(\mathrm{cum}_{2p}(y_n)\mathbf{q}^{\odot 2p-1} - f_{2p}(\mathbf{q})\mathbf{q}\right) \qquad (2.18)\\
\mathbf{q}_{k+1} &= \frac{\mathbf{q'}_k}{\|\mathbf{q'}_k\|_2} \qquad (2.19)
\end{aligned}$$

By setting the step-size parameter μ_k equal to $\mu_k = \frac{1}{|f_{2p}(\mathbf{q}_k)|}$ the equation (2.18) will be simplified to:

$$\mathbf{q'}_k = \pm\,\mathrm{cum}_{2p}(y_n)\mathbf{q}_k^{\odot 2p-1}$$

which can be recognized as the super-exponential algorithm in the system space. This shows that the super-exponential algorithm can be seen as a gradient search procedure based on the Godard criterion with a specific value of the step size parameter.

2.6.3 Fast-ICA and Godard

Although ICA creates a separate/independent group of algorithms, under some assumptions it can be closely related to the $f_{2p}(\mathbf{q})$ family of algorithms. In the sequel, we suppose that the input data are pre-whitened so that $E[\mathbf{r}\mathbf{r}^T] = I$ and we suppose a sub-Gaussian source. Then, for the nonlinearity $G(x) = \frac{1}{4}x^4$, we get its derivative $g(x) = x^3$. Now consider the steepest descent algorithm (2.11) (descent because of

sub-Gassian source); with the selected nonlinearity, it assumes the form:

$$\begin{aligned}
\mathbf{f}' &= \mathbf{f}_k + \mu_k \left(E\left[\mathbf{r} g(\mathbf{r}^T \mathbf{f}_k)\right] - \beta_k \mathbf{f}_k \right) \\
&= \mathbf{f}_k + \mu_k \left(\sum_n \mathbf{r}_n y_n^3 - \left[\sum_n \mathbf{r}_n y_n^3 \right]^T \mathbf{f} \cdot \mathbf{f} \right) \\
&= \mathbf{f}_k + \mu_k \left(\mathcal{R}^T \mathbf{y}^{\odot 3} - \underbrace{\mathbf{f}^T \mathcal{R}}_{\mathbf{y}^T} \mathbf{y}^{\odot 3} \mathbf{f} \right) \\
&= \mathbf{f}_k + \mu_k \left(\mathcal{R}^T \mathbf{y}^{\odot 3} - \|\mathbf{y}\|_4^4 \, \mathbf{f} \right)
\end{aligned} \tag{2.20}$$

where $\mathcal{R}$ is input (pre-whitened) data matrix as defined in (3.21). Equation (2.20) can be recognized as the Finite-Interval CMA developed by Regalia in [43] - see section 3.4. In [47], the step size bounds for the gradient search procedure for ICA (2.11) and the optimal step size value were developed. Thanks to the equivalence of fast-ICA with FI-CMA for the nonlinearity $G(x) = x^4$, the results of [47] can be directly applied for FI-CMA.

Chapter 3

CM Algorithms

We have shown in the previous chapter some of the best known and developed HOS blind equalization techniques. It was demonstrated that at least three of them optimize the same criterion. For our implementation, we have decided to focus on the CMA-based algorithms. The reason is that their theoretical development seems to have stabilized and they are ripe for implementation.

In this chapter we will review the properties of the Constant Modulus cost function in more detail and describe several different CM optimization algorithms, including their equalization performance and convergence in basic scenarios. It will help us to set up the background necessary for derivation of the sliding window CMA algorithm in the next chapter.

3.1 CM properties

In this section we shall discuss various properties of the CM cost function, such as the surface (in the system space), number and position of extrema in the noiseless and noisy case. The influence of common channel roots and correlated source will be also demonstrated.

The position of the stationary points was studied in many papers for different sources. Since J_{CM} is (highly) non-linear criterion, its analysis is difficult and often

only possible under some simplifying assumptions. That is why most of related papers use assumptions of noiseless channel and infinite equalizer length [4] [24] [32] [37] . Their position can be expressed by calculating the gradient of J_{CM} (2.2) with respect to the equalizer coefficient vector $\mathbf{f}$ and setting it to zero:

$$\begin{aligned} \nabla_{\mathbf{f}} J_{\mathrm{CM}} &= \mathbf{H} E\left[\left((\mathbf{q}^T\mathbf{s}_n)^3 - \gamma \mathbf{q}^T\mathbf{s}_n\right)\mathbf{s}_n\right] = 0 \\ &\Rightarrow E\left[\left((\mathbf{q}^T\mathbf{s}_n)^3 - \gamma \mathbf{q}^T\mathbf{s}_n\right)\mathbf{s}_n\right] = E\left[(\mathbf{q}^T\mathbf{s}_n)\left(\left|\mathbf{q}^T\mathbf{s}_n\right|^2 - \gamma\right)\mathbf{s}_n\right] = 0 \quad (3.1) \end{aligned}$$

It is seen from (3.1) that the J_{CM} cost function is a multi-modal criterion. Two conditions giving the zero gradient correspond to one local maxima at $\mathbf{q} = 0$ and to a set of local minima and saddle points resulting from $\left|\mathbf{q}^T\mathbf{s}\right|^2 = \gamma$. From analysis given in [4] and [24] follows that:

- there are $2N$ local minima and under the PE condition all these minima correspond to the ZF system response. Moreover, in the noiseless case all these minima are equivalent in the MSE sense and they are global minima. Each minimum corresponds to a different phase shift and time delay.

- there are $3^N - 2N - 1$ saddle points, where each saddle point corresponds to the system response with $n = 1 \ldots N$ non-zero elements which are all of the same magnitude.

- there is only one local maximum in the origin $\mathbf{q} = 0$. This false solution is an unstable point and in practical implementation does not establish real problem, since it is sufficient to initialize $\mathbf{f}$ to some nonzero value.

- there is a strong relation between the CM equalizer and the Wiener filter, at least when the SNR approaches infinity. It was shown that in the noisy case, the CM solution lies in the neighborhood of MMSE.

The shape of the cost function surface has non-negligible influence on the behavior of the stochastic gradient search algorithm, as has been extensively analyzed in many papers [13] [25] [26] [81]. When the PE conditions are not satisfied, an appropriate

optimization algorithm can converge to a 'good' or 'bad' solution, depending on the cost function surface and the initialization point. As is seen from Figures 3.1 - 3.5, some solutions have higher MMSE ('bad' solutions) than others ('good' solutions). In the sequel, we show the effect of different sources of distortion on the J_{CM} cost function. In the examples the real-valued FS channel, two-tap equalizer and zero mean, i.i.d. BPSK source signal will be assumed. In the plots, the position of the CM minima is marked by crosses (+) whereas the position of the MSE minima is marked by circles (o).

CM cost function under perfect equalization condition In the case when the perfect equalization conditions are satisfied (noiseless scenario, channel matrix $\mathbf{H}$ has full column rank and is invertible), there exists the ZF solution (see section 1.3), i.e., there exists an equalizer for which the system impulse response is $\mathbf{e}_i = [0 \dots 0\ 1\ 0 \dots 0]$. The J_{CM} cost surface and contour plot corresponding with this case are shown in Figure 3.1 and 3.2. It can be seen that the shape of the cost surface is symmetric with respect to the origin and that it has multiple minima (is multi-modal). Moreover, the MSE minima show the same symmetry and their location coincides with the minima of J_{CM}. It follows that the J_{CM} and MSE cost functions are closely related in the neighborhood of local minima. Note that both J_{CM} and MSE optima give the optimal ZF equalizer because the residual of the cost function equals zero.

Influence of additive noise on the J_{CM} cost surface The deformation of J_{CM} cost surface caused by an additive white Gaussian noise is shown in Figure 3.3. It can be seen that the shape of the cost surface gets "narrower", that is, the location of local minima is now closer to origin, and the J_{CM} function has non-zero minima with different residual values. This implies that the gradient search procedure can converge to 'good' or 'bad' local minima with low resp. high residual value. The MSE and J_{CM} minima do not coincide, nevertheless the deformation caused by additive noise is small enough so that their minima lie in close proximity.

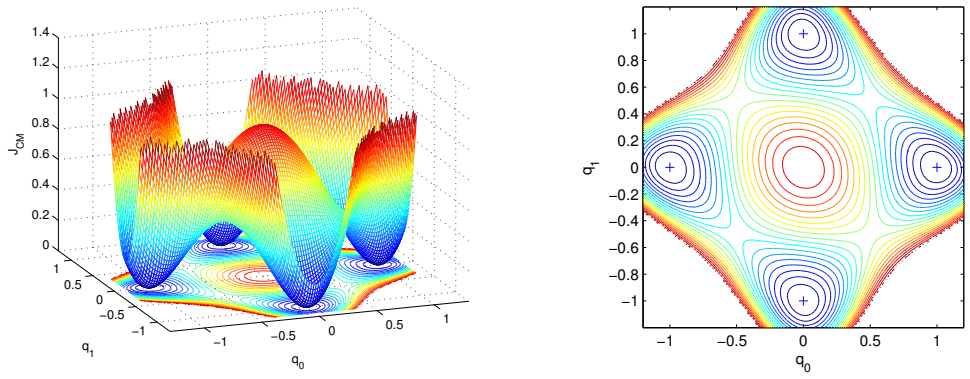


Figure 3.1: J_{CM} surface and contour under perfect equalization condition in system space

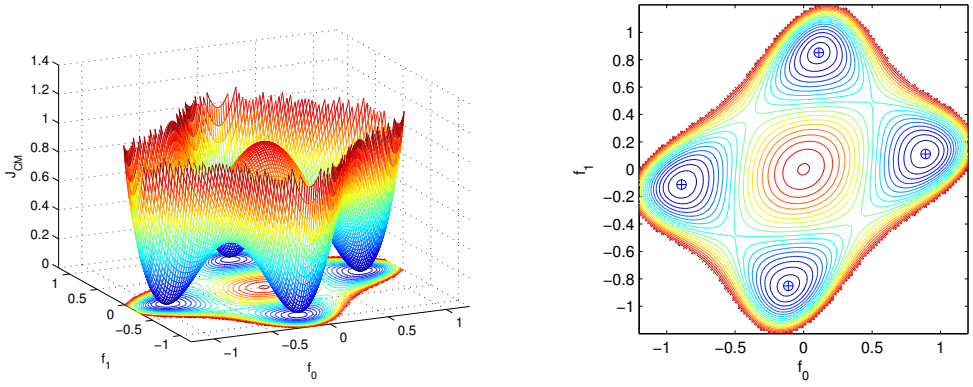


Figure 3.2: J_{CM} surface and contour under perfect equalization condition. Equalizer space.

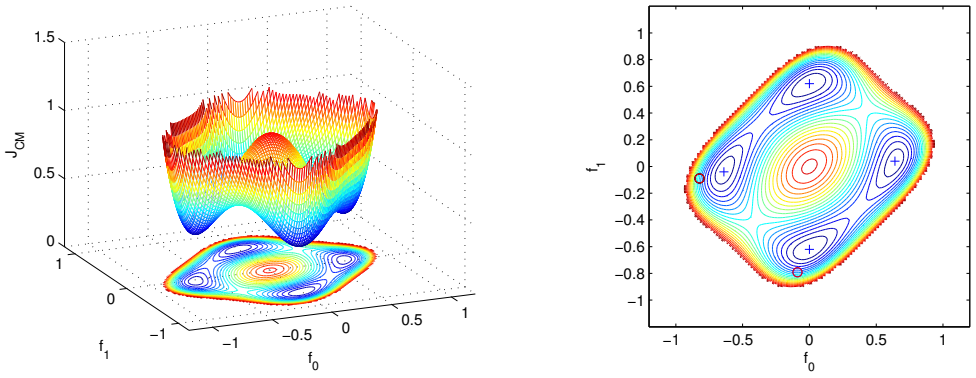


Figure 3.3: J_{CM} surface and contour in equalizer space with 10dB AWGN

Common channel roots It is assumed, in perfect equalization, that the channel convolution matrix is invertible. However, in the case of fractionally spaced equalization, if the sub-channels share common roots, the channel convolution matrix becomes singular and the performance of equalization algorithms significantly degrades. It was shown in [14] and [15] that the fractionally spaced CMA equalizers are still able to equalize reasonably. In Figure 3.4, the roots of the T/2 spaced channel impulse response and the corresponding baud spaced sub-channels with nearly common roots can be seen. The T/2 spaced channel impulse response has symmetric roots with respect to the origin.

Correlated source The case of J_{CM} cost surface with the correlated source signal (non i.i.d.) is shown in the Figure 3.5. In digital communications, this kind of source correlation may occur as a result of source coding (convolution codes, differential encoding). Correlating the signal makes the source more Gaussian-like and its kurtosis tends to zero (see Chapter 1.4.1). It can be seen from the surface plot, that since the source remains non-Gaussian the position of local minima does not change but the shape of J_{CM} cost surface is flattened, making the convergence of the CM minimization procedure slower.

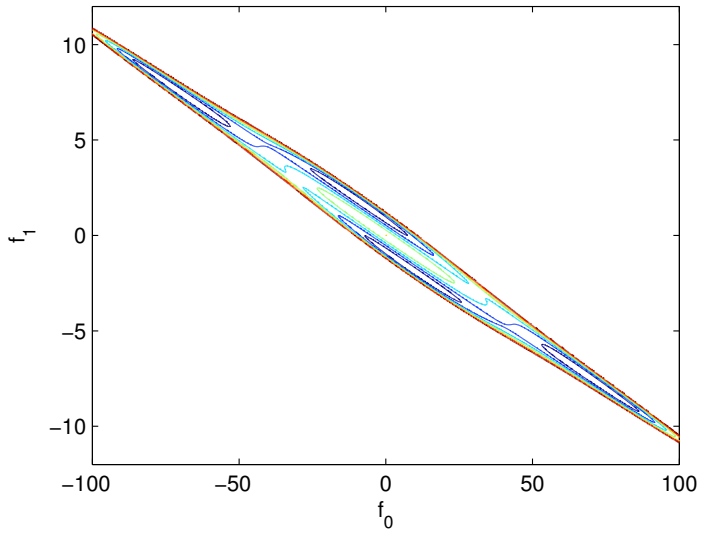


Figure 3.4: J_{CM} surface deformation caused by nearly common channel roots. Equalizer space, no noise.

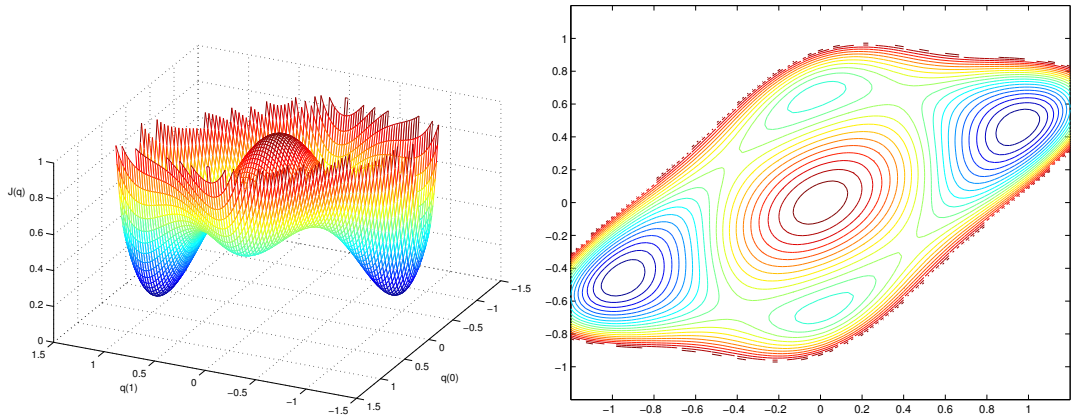


Figure 3.5: Shape of the J_{CM} surface caused by correlated source. No noise.

3.2 Gradient CMA

The main advantage of the CM criteria J_{CM} is that there exists a simple gradient search algorithm of the form:

$$\mathbf{f}(k+1) = \mathbf{f}(k) - \mu \nabla_{\mathbf{f}}(J_{\mathrm{CM}})$$

where k is a discrete time index and $\nabla_{\mathbf{f}}(J_{\mathrm{CM}})$ is a gradient of $J_{\mathrm{CM}}(\mathbf{f})$ by $\mathbf{f}$. Calculating the gradient directly from (2.2) we obtain:

$$\begin{aligned} \nabla_{\mathbf{f}}(J_{\mathrm{CM}}) &= \frac{1}{2} E\left[\left(|y_n|^2 - 1\right) \cdot \nabla_{\mathbf{f}}(\mathbf{f}^H \mathbf{r}_n^* \mathbf{r}_n^T \mathbf{f})\right] && (3.2) \\ &= E\left[\left(|y_n|^2 - 1\right) \mathbf{r}_n^* \mathbf{r}_n^T \mathbf{f}\right] && (3.3) \\ &= E\left[\left(|y_n|^2 - 1\right) y_n) \mathbf{r}_n^*\right] && (3.4) \end{aligned}$$

Replacing the true gradient by its instantaneous gradient estimate we obtain the recursive update formula:

$$\mathbf{f}_{n+1} = \mathbf{f}_n - \mu \left((|y_n|^2 - 1) y_n \mathbf{r}_n^*\right) = \mathbf{f}_n - \mu e_n \mathbf{r}_n^* \tag{3.5}$$

which has form similar to the popular LMS algorithm. Depending on the eigenvalue spread of the channel matrix, the CMA algorithm is known to have very slow convergence, typically the convergence rate is approximately 10^4 iterations (see Figures 3.6 to 3.8).

To improve the convergence speed, it has been proposed to use some preprocessing. In [33] the input signal is pre-whitened by an adaptive linear predictor. The error signal Δr_n between the linear predictor output signal $\hat{\mathbf{r}}_n$ and the input signal $\mathbf{r}_n$ is used as the input of the standard gradient CMA algorithm. In [33], linear predictor coefficients $\mathbf{w}$ are updated by the following formula:

$$\begin{aligned} \Delta r_n &= r_n - \mathbf{w}_n^T \mathbf{r}_n && (3.6) \\ \mathbf{w}_{n+1} &= \mathbf{w}_n + \beta \Delta r_n \mathbf{r}_n && (3.7) \end{aligned}$$

where the β is a step size parameter. This method significantly improves the convergence rate with small additional computation (since the computational complexity of

(3.6) is linear with the length of $\mathbf{w}$). In [8] the received signal is first pre-filtered by linear predictor whose coefficients are obtained from input auto-covariance matrix by solving the Yule-Walker equation. Then, the following iterative update procedure is used:

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu(I \otimes \mathbf{f}_n^T)\mathbf{Q}_\mathbf{r}(\mathbf{f}_n^* \otimes I)\mathbf{f}_n - 4\mathcal{R}_\mathbf{r}\mathbf{f}_n \tag{3.8}$$

where $\otimes$ denotes the Kronecker product, $\mathcal{R}_\mathbf{r}$ is the estimated input signal covariance matrix and $\mathbf{Q}_\mathbf{r} = \frac{1}{N}\sum_n \mathbf{r}_n^T\mathbf{r}_n \otimes \mathbf{r}_n^T\mathbf{r}_n$ is the estimated quadricovariance matrix with N being the number of samples used for estimation. Note that although this algorithm was designed as a block processing one, using forgetting and recursive covariance update known from the RLS algorithms, it could be used in a recursive form.

3.2.1 Simulation examples

The typical behavior of the gradient CMA algorithm and its variants is shown in Figures 3.6 to 3.8. All results were obtained using the double precision calculations in Matlab. We have considered a system with single input/two outputs FIR channel of the 10^{th} order with randomly chosen coefficients and linear equalizer of length 9. The input data stream $\mathbf{s}_n$ is a random (i.i.d.) binary sequence. In these examples only noise free cases are presented.

The results of the gradient CMA algorithm are presented in Figures 3.6 to 3.8 for three different settings of the step size μ. As can be seen, the convergence speed of the algorithm is very dependent on the value of the step size parameter. Unfortunately, its optimal value depends on the parameters of the channel, which are unknown. An improper step size value can even cause the algorithm to diverge.

In Figure 3.6 the case with the "optimal" step size $\mu = 0.025$ is shown. It can be seen that the algorithm converges after several thousands iterations, but the ISI eye is open already after several hundreds iterations. The residual error in steady state is $J_{CM} = 0.0224$ with the variance $\sigma^2 = 0.0013$. Note that setting the step size slightly higher (for example, $\mu = 0.025001$) will cause the algorithm to diverge. This

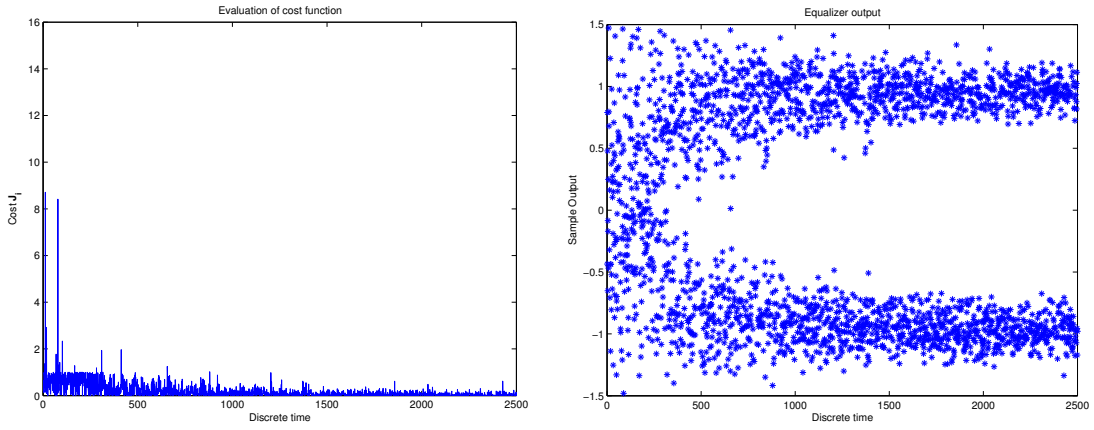


Figure 3.6: Evolution of cost function and equalizer output of gradient CMA algorithm with "optimal" step size $\mu = 0.025$

represents significant problem in practical implementations with finite data precision.

In Figure 3.7 the case with the step size $\mu = 0.01$ is shown. This value still ensures reasonable convergence while avoiding the risk of divergence caused by round-off errors. It can be seen that the convergence speed is bit slower in comparison with the previous case (approximately ten thousands iterations) and the ISI eye is open after approx. thousands iterations. In this example, the residual error in steady state is $J_{CM} = 0.0601$ with variance $\sigma^2 = 0.0111$.

Lowering the step size parameter will cause the convergence speed to slow down. The case for $\mu = 0.005$ can be seen in Figure 3.8 with steady state residual error $J_{CM} = 0.1512$ and variance $\sigma^2 = 0.0341$. We can conclude that the algorithm convergence speed is very dependent on the value of the step size and channel parameters and that – even for the noiseless case – the residual output error is relatively high.

3.3 Algebraic CMA

As mentioned earlier, the gradient-based CM algorithms suffer from the following drawbacks:

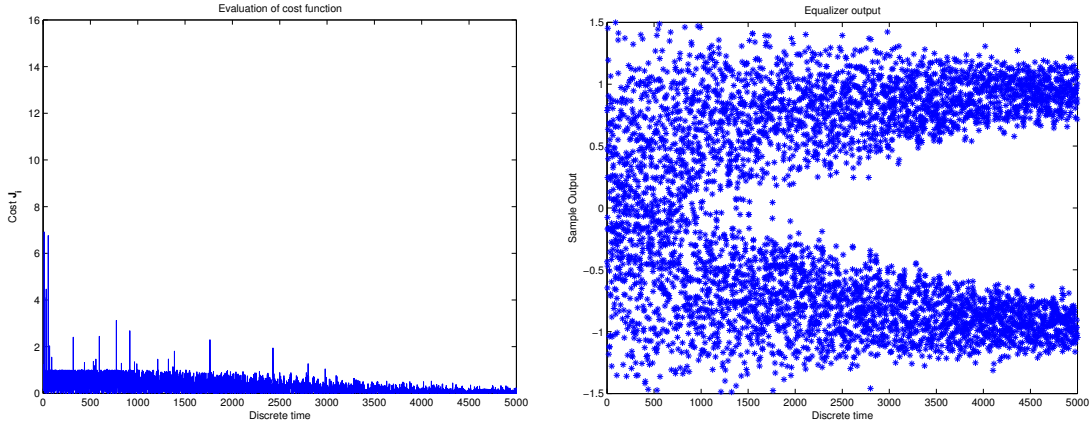


Figure 3.7: Evolution of cost function and equalizer output of gradient CMA algorithm with step size $\mu = 0.01$

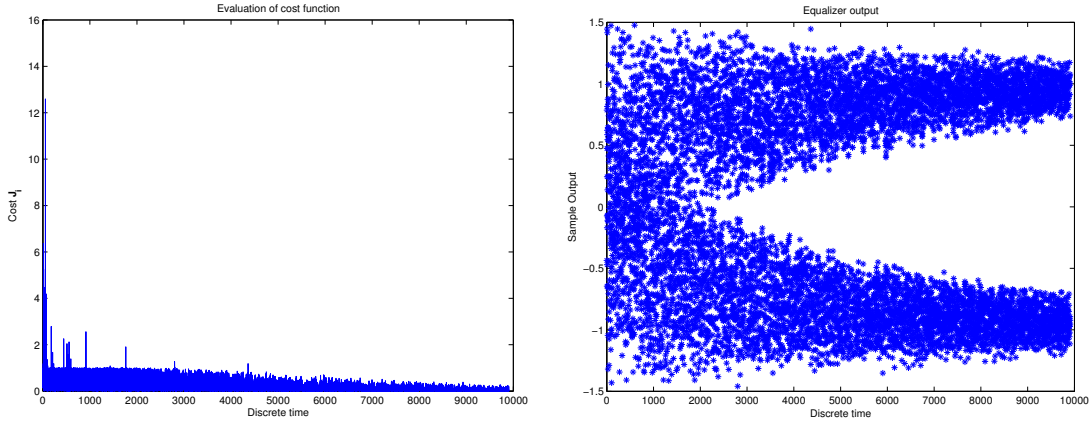


Figure 3.8: Evolution of cost function and equalizer output of gradient CMA algorithm with step size $\mu = 0.005$

- there exist 'good' and 'bad' solutions with respect to residual MSE. No method is known to ensure the convergence to the 'good' solution

- the convergence rate is dependent on the initialization value

- in MIMO case, the number of CM sources is unpredictable

The work of Van der Veen and Paulraj [74] [76] was motivated by these open questions. They have found an algebraic solution to the CM separation problem: they replaced the problem of finding the equalizer coefficients blindly with the equivalent problem of blind source separation. In the algebraic CM algorithm (ACMA), all weight vectors are found simultaneously using a set of diagonalizations of data matrices. An initialization procedure is not needed (hence, the initialization problem is removed).

When reviewing the ACMA, we consider the MIMO communication scheme with d independent CM sources to be transmitted over a communication channel. Suppose the number of antennas is M where $M > d$ and the input data block size fulfils the $N \geq d^2$ inequality. Derivation of ACMA in [74],[76] starts by postulate that the problem of finding optimal equalizer weights is equivalent to finding all independent signals $\mathbf{s}_i, \quad i = 1 \ldots d$ that satisfy the condition:

$$\mathbf{s}_i \in \text{span}(\mathcal{R}) \qquad \mathbf{s}_i \in (CM)$$

where the (CM) is a set of all constant modulus signals and span$(\mathcal{R})$ denotes a row span of the input data matrix $\mathcal{R} = \mathbf{H}_A[\mathbf{s}_1 \ \ldots \ \mathbf{s}_d]^T$.

Let $\mathcal{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$ be singular value decomposition (SVD) of the input data matrix $\mathcal{R}$. If there are d independent sources, the rank of matrix $\mathcal{R}$ is equal to d and consequently, diagonal matrix $\mathbf{\Sigma}$ has d non-zero singular values. Let us define the corresponding part of matrix $\mathbf{V}$ as $\hat{\mathbf{V}} \in C^{d \times N}$, where the rows of $\hat{\mathbf{V}}$ define the orthonormal basis of

Given $\mathcal{R}$ the optimal coefficients $\mathbf{W}$ are computed as follows:

1. Estimate the number of sources $d = \text{rank}(\mathcal{R})$
   - compute SVD of data matrix $\mathcal{R} = \mathbf{U\Sigma V}$
   - estimate number of sources d from $\mathbf{\Sigma}$
   - construct orthonormal basis matrix $\hat{\mathbf{V}}$
2. Solve the linear system $\mathbf{P}\hat{\mathbf{w}} = \mathbf{1}$
   - Construct $\mathbf{P}$
   - compute $\mathbf{G}$ using the QR factorization of $[\mathbf{1}|\mathbf{P}]$
   - compute SVD of $\mathbf{G}$ and get orthonormal basis of its null space $\{\hat{\mathbf{w}}_i\}_1^d$
3. Solve the simultaneous diagonalization problem
   - construct $\hat{\mathbf{W}}_i = \text{vec}^{-1}(\hat{\mathbf{w}}_i)$
   - find $\mathbf{T}$ to jointly diagonalize $\hat{\mathbf{W}}_i$ as $\hat{\mathbf{W}}_i = \mathbf{T\Lambda}_i\mathbf{T}$
   - scale each column of $\mathbf{T}$ to unit norm and calculate a coefficient matrix $\mathbf{F} = [\mathbf{f}_1 \ \dots \ \mathbf{f}_d] = \mathbf{U\Sigma}^{-1}\mathbf{T}$

Figure 3.9: ACMA summary

row span of $\mathcal{R}$. The condition is then rewritten as:

$$\mathbf{s}_i \in \text{span}(\mathcal{R}) \iff \mathbf{s}_i \in \mathbf{f}^H\hat{\mathbf{V}} \tag{3.9}$$

$$\mathbf{s}_i = [s_1, s_2 \dots s_N] \in CM \iff$$

$$\left[|s_1|^2, |s_2|^2, \dots, |s_N|^2\right] = \left[\mathbf{f}^H(\mathbf{v}_1\mathbf{v}_1^H)\mathbf{f}, \dots, \mathbf{f}^H(\mathbf{v}_N\mathbf{v}_N^H)\mathbf{f}\right] = [1\,1 \ \dots \ 1] \tag{3.10}$$

where $\mathbf{v}_k$ is the kth column of $\hat{\mathbf{V}}$. Using (3.9) and (3.10), we obtain a set of equations, which has to be satisfied for all independent linear vectors $\mathbf{f}$ of the form:

$$\mathbf{f}^H\mathbf{P}_k\mathbf{f} = 1 \quad k = 1, 2, \dots, N \tag{3.11}$$

where the $\mathbf{P}_k = \mathbf{v}_k \mathbf{v}_k^H$ is a $N \times N$ matrix. The quadratic form (3.11) can be linearized when written in terms of the Kronecker product:

$$\mathbf{f}^H \mathbf{P}_k \mathbf{f} = \mathbf{p}_k \hat{\mathbf{w}} \tag{3.12}$$

where $\mathbf{p}_k = \mathrm{vec}(\mathbf{P}_k)$ is vectorized matrix $\mathbf{P}_k$ and $\hat{\mathbf{w}} = \mathrm{vec}(\mathbf{f}^H \mathbf{f}) = \mathbf{f} \otimes \mathbf{f}^*$. With this decomposition we get:

$$\left[\begin{array}{c|c} \mathbf{p}_1 & 1 \\ \vdots & \vdots \\ \mathbf{p}_N & 1 \end{array}\right] \left[\begin{array}{c} \hat{\mathbf{w}} \\ -1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{P} & \mathbf{1} \end{array}\right] \left[\begin{array}{c} \hat{\mathbf{w}} \\ -1 \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right] \tag{3.13}$$

Note that there are d independent solutions of (3.13) for $\{\hat{\mathbf{w}}_i\}_{i=1}^d$ and that linear combination $\hat{\mathbf{w}} = \sum \lambda_i \hat{\mathbf{w}}_i$, scaled such that $\sum \lambda_i = 1$, also solve this equation. Then, calculating the QR factorization of the compound matrix

$$\mathbf{Q} \left[\begin{array}{c|c} \mathbf{1} & \mathbf{P} \end{array}\right] = \sqrt{N} \left[\begin{array}{cc} 1 & \hat{\mathbf{p}}^H \\ \mathbf{0} & \mathbf{G} \end{array}\right]$$

we obtain the following system of linear equations:

$$\begin{aligned} \hat{\mathbf{p}}^H \hat{\mathbf{w}} &= 1 \\ \mathbf{G} \hat{\mathbf{w}} &= \mathbf{0} \end{aligned} \tag{3.14}$$

Since the first condition in (3.14) can be satisfied by proper scaling of $\hat{\mathbf{w}}$ it could be dropped out. Then all solutions to the second condition i.e. $\mathbf{G}\hat{\mathbf{w}} = \mathbf{0}$, are formed by basis $\{\hat{\mathbf{w}}_i\}_{i=1}^d$ of the null space of $\mathbf{G}$, which is obtained by the calculation of SVD factorization.

The next step in the algorithm is the decoupling i.e. finding the basis $\{\mathbf{f}_i \otimes \mathbf{f}_i\}_{i=1}^d$ of structured vector which spans the same subspace as $\{\hat{\mathbf{w}}_i\}_{i=1}^d$. This is solved by finding matrix $\mathbf{T}$ which jointly diagonalizes the matrices $\hat{\mathbf{W}}_i = \mathbf{vec}^{-1}(\hat{\mathbf{w}}_i)$. This generalized eigen-value decomposition can be solved by several methods (see [9] for an overview of joint diagonalization methods). In the original work [76] the generalized Gerchberg-Saxton iteration procedure was used.

Finally, in [74] the properties of the proposed algorithm are studied for both the noiseless and noisy channels. It was experimentally shown that ACMA asymptotically converges to the Wiener solution as the input data length N and SNR go to infinity. A modified version of the algorithm, ZF-ACMA, was proposed, which asymptotically converges to the zero forcing solution under the same conditions.

3.3.1 Simulation examples

The typical behavior of the Algebraic CMA algorithm is shown in Figures 3.10 to 3.13. All presented results were obtained using the double precision calculations in Matlab. Since the ACMA is used to simultaneously deconvolve all CM sources at once, we have considered MIMO channels. The algorithm was tested on the BPSK and QAM sources. Since the algorithm uses computations demanding high data precision and sensitive to the input data properties (namely the SVD decomposition), the influence of the AWGN is also presented. We compare the algorithm using the bit-error ratio (BER) and the residual CM cost function.

The BER curves for the system with three and two independent QAM sources are shown as a function of the SNR in Figure 3.10. The "first source" corresponds to the best score i.e. the output sequence with the lowest BER, the second source corresponds to the output sequence with second lowest BER etc. It can be seen that as the noise power grows, the difference in estimation quality between the identified output sequences grows significantly. The mean value of the residual CM cost function J_{CM} (see (2.2) for definition) is shown if Figure 3.12.

The results for the BPSK sources are shown in Figures 3.11 and 3.13. Since the ACMA algorithm does not work well for BPSK signals, modified real ACMA algorithm was used in that case. It can be seen that in the BPSK case, the difference in estimation quality between the simultaneously deconvolved signals is not as high as for the QAM source and that this difference is recognizable only for low SNR.

All results are based on the original algorithm, where the number of CM sources for the received signal is assumed to be known a priori. If the number of CM sources

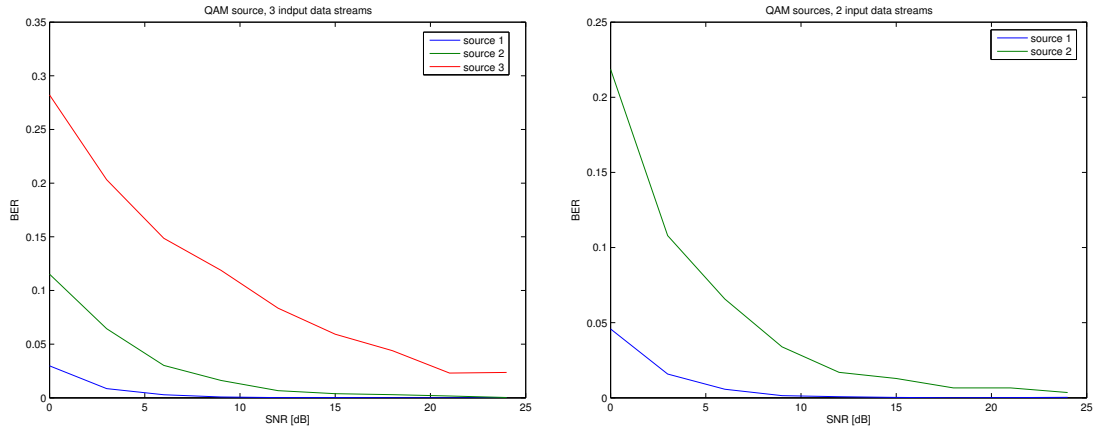


Figure 3.10: Bit error ration as a function of SNR of algebraic CMA algorithm for 2 and 3 QAM sources of the length 500

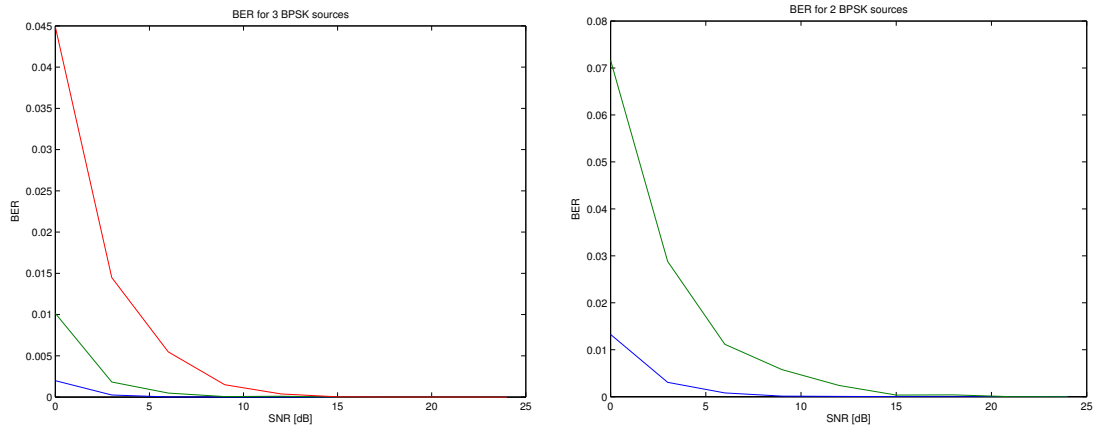


Figure 3.11: Bit error ration as a function of SNR of realACMA algorithm for 2 and 3 BPSK sources of the length 500

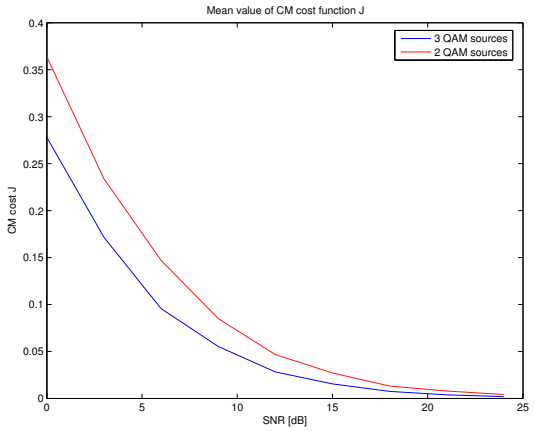


Figure 3.12: Residual CM cost function J_{CM} of algebraic CMA algorithm for two and three QAM sources

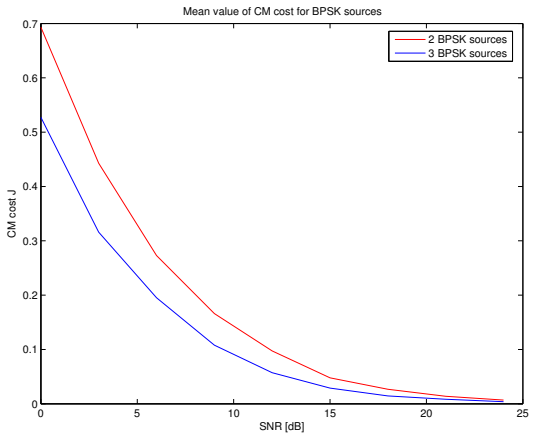


Figure 3.13: Residual CM cost function J_{CM} of real ACMA algorithm for two and three BPSK sources

is estimated by algorithm (see the original work [76]) the results are much worse.

The reason for the difference in quality of estimation of simultaneously deconvolved sources, is caused by the joint diagonalization of the $\mathbf{W}_i$. Namely, it is important, how close are $\mathbf{W}_i = \mathbf{w}_i^T \mathbf{w}_i$ to rank(1) matrices. We illustrate it in the example of three QAM sources for two values of SNR. In the case of no AWGN, the SVD decomposition of matrices $\mathbf{W}_i$ gives its singular values:

$$\begin{matrix} 1.0025 & 0.0006 & 0.0006 \\ 1.0014 & 0.0004 & 0.0004 \\ 1.0020 & 0.0004 & 0.0004 \end{matrix}$$

We can say, that in that particular case, we got the $\mathbf{W}_i$ that are very close to rank(1) and equivalent (with similar eigenvalue spread and BER). When the SNR=25dB we got the singular values:

$$\begin{matrix} 1.0007 & 0.0055 & 0.0055 \\ 1.0012 & 0.0041 & 0.0041 \\ 1.0010 & 0.0034 & 0.0034 \end{matrix}$$

which are acceptably close to rank 1, but with non-equal eigenvalue spread. Although all three matrices $\mathbf{W}_i$ are close to rank 1 and their deviation is small, they produce

very different output in the sense of residual J_{CM} and BER.

It is possible to conclude that the algorithm is computationally intensive and requires high data precision. Lowering the bits of data representation will result in significant degradation of the algorithm performance.

3.4 Finite Interval CMA

The FI-CMA is a batch processing version of the CMA algorithm. The time-window operator is applied to the received data (i.e., the expectation operator in (2.2) is replaced by summation over finite data interval) and several iterations of equalizer update are applied to the current data matrix to obtain the equalizer output. Replacing integration by the summation operator, we get, directly from (2.2), the modified CM cost function of the form:

$$J(\mathbf{w}) = \sum_{n=0}^{N-1} \left(|y_n|^2 - 1 \right)^2 = \sum_{n=0}^{N-1} \left(\left| \mathbf{f}^T \mathbf{r}_n \right|^2 - 1 \right)^2 \tag{3.15}$$

where the constant γ was replaced by 1 without loss of generality because its value has influence to the scaling of $\mathbf{f}$ resp. y_n, but does not change the position of the local extrema points.

Using the decomposition of the equalizer coefficient vector $\mathbf{f}$ into the radial term f_r and the unit-norm direction term $\overline{\mathbf{f}}$ (as in Section 2.6.1) such that :

$$\mathbf{f} = f_r \overline{\mathbf{f}} \qquad \left\| \overline{\mathbf{f}} \right\| = 1, \quad f_r = \|\mathbf{f}\|$$

we may rewrite the finite interval CM criterion (3.15) as:

$$J(\mathbf{f}) = \sum_{n=0}^{N-1} \left(f_r^2 \left| \overline{\mathbf{f}}^T \mathbf{r}_n \right|^2 - 1 \right)^2 = \sum_{n=0}^{N-1} \left(f_r^2 \left| \overline{y}_n \right|^2 - 1 \right)^2 \tag{3.16}$$

where the term $\overline{y}_n = \overline{\mathbf{f}}^T \mathbf{r}_n$ denotes the output of the unit norm filter $\overline{\mathbf{f}}$. Setting the derivative of (3.16) with respect to f_r to zero, we obtain the extrema points: one

$f_r = 0$ which corresponds to the local maximum of J_{CM}, and multiple minima points, which occur when:

$$f_{r_{opt}}^2 = \frac{\sum_n \overline{y}_n^2}{\sum_n \overline{y}_n^4} \tag{3.17}$$

Note: With the choice $f_r = f_{r_{opt}}$, we get the following equivalence :

$$\sum_{n=0}^{N-1} y_n^2 = \sum_{n=0}^{N-1} y_n^4 \tag{3.18}$$

With the choice of the optimal radial term f_r, and with the equality (3.18), the cost function (3.15) has the form:

$$\overline{J}(\overline{\mathbf{f}}) \triangleq J(f_r \overline{\mathbf{f}})\Big|_{f_{r_{opt}}} = N - \sum_{n=0}^{N-1} y_n^2 = N - \frac{\left(\sum_{n=0}^{N-1} \overline{y}_n^2\right)^2}{\sum_{n=0}^{N-1} \overline{y}_n^4} \tag{3.19}$$

It is seen from (3.19), that by setting $f_r = f_{r_{opt}}$, the minimization of (3.15) with respect to the direction term $\overline{\mathbf{f}}$ is equivalent to the minimization of the criterion defined as follows (for details see [43]):

$$\mathbf{J} \triangleq \frac{\sum_{n=0}^{N-1} y_n^4}{\left(\sum_{n=0}^{N-1} y_n^2\right)^2} = \frac{m_4}{m_2^2} \qquad \text{where} \quad m_k = \sum_n y_n^k \tag{3.20}$$

This gives us an equivalent scale invariant cost function, which is dependent only on the fourth and the second order moment of the equalizer output. To develop an iterative coefficient update procedure, we may write the successive equalizer outputs

in the form:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{r}_0^T \\ \mathbf{r}_1^T \\ \vdots \\ \mathbf{r}_{N-1}^T \end{bmatrix}}_{\mathcal{R}} \mathbf{f} = \mathbf{QRf} = \mathbf{Qw} \tag{3.21}$$

where the QR-decomposition of the $\mathcal{R}$ matrix was used. This step can be viewed as a pre-whitening process which speeds up the convergence rate. Also the orthonormality of the columns of $\mathbf{Q}$ will simplify the equalizer update calculation. Using the QR-decomposition, we may calculate the gradient of equivalent objective function (3.20) with respect to the coefficient vector $\mathbf{w}$ as:

$$\frac{\partial}{\partial \mathbf{w}} \frac{\sum_n y_n^4}{\left(\sum_n y_n^2\right)^2} = \frac{4m_2^2 \sum_n y_n^3 \mathbf{q}_n - 4m_4 \sum_n y_n \mathbf{q}_n}{\left(\sum_n y_n^2\right)^4} \tag{3.22}$$

where $\mathbf{q}_n$ is the nth row of matrix $\mathbf{Q}$. Since (3.20) is invariant to the radial term, we may constrain equalizer coefficients $\mathbf{w}$, without loss of generality, to have unit norm. Note that since the matrix $\mathbf{Q}$ has orthonormal columns and $\mathbf{w}$ is unitary, an equality $\sum y_n^2 = \mathbf{f}^T\mathcal{R}^T\mathcal{R}\mathbf{f} = \mathbf{w}^T\mathbf{Q}^T\mathbf{Q}\mathbf{w} = 1$ holds and simplifies the equivalent criterion to the form $\mathbf{J} = m_4$. Applying this property to (3.22), the gradient is reduced to:

$$\nabla_{\mathbf{w}}\mathbf{J} = \mathbf{Q}\mathbf{y}^{\odot 3} - \mathbf{w}\mathbf{J} \tag{3.23}$$

where the operator $(\cdot)^{\odot k}$ means the element-wise kth-power. Finally, using the gradient calculation, the equalizer is updated by block step-descent algorithm of the form:

$$\begin{aligned} \mathbf{v}_{i+1} &= \mathbf{w}_i - \mu(\mathbf{Q}^T\mathbf{y}^{\odot 3} - \mathbf{J}_i\mathbf{w}_i) \\ \mathbf{w}_{i+1} &= \mathbf{v}_{i+1}/\|\mathbf{v}_{i+1}\| \end{aligned} \tag{3.24}$$

where i is the iteration step, μ is a step size parameter and $\mathbf{J}_i$ is the cost function defined as $\mathbf{J} = m_4$. More details on the algorithm derivation can be found in the original work [43].

The bounds of the step size parameter μ ensuring the algorithm convergence were derived in [43], where was also proposed the sub-optimal parameter value, giving reasonable convergence in most cases, to be equal to

$$\mu_i = \frac{\alpha}{\mathbf{J}_i}$$

where $\alpha \in \left(\frac{1}{2}, \frac{2}{3}\right)$. Since the procedure (3.24) can be viewed as a special case of the Fast-ICA algorithm (2.11) with the non-linearity $G = \frac{1}{4}u^4$ (see section 2.4 for comparison), the analysis of the optimal step size for Fast-ICA can be directly applied also for FI-CMA. It was proven in [47], that the optimal step size of the update procedure (2.11) for a sub-Gaussian source is equal to $\mu^{opt} = 1/(\beta_i - E[g'(\mathbf{r}^T\mathbf{f}_i)])$. Substituting proper non-linearity, it directly follows that the optimal step size for the FI-CMA procedure (3.24) is:

$$\mu_i^{opt} = \frac{1}{3 - \mathbf{J}_i}$$

where $\mathbf{J}_i = m_4$ denotes the modified criterion value after the ith iteration. In our experiments we have also used step size $\mu = \frac{\sqrt{N}}{(\sqrt[4]{\mathbf{J}_i - \sqrt{N}\mathbf{J}_i})}$ where N is data block size. This value of step size simplifies the first row of equation (3.24) to the form:

$$\mathbf{v}_{i+1} = \mathbf{w}_i - \frac{\sqrt{N}}{\sqrt[4]{\mathbf{J}_i}}\mathbf{Q}^T\mathbf{y}^{\odot 3} \tag{3.25}$$

3.4.1 Simulation examples

The behavior of the Finite Interval CMA is shown in Figures 3.14 to 3.15. In these examples the equalizer output and criterion flow is shown. All presented results were obtained using double precision calculations in Matlab. We have considered a system with a single input/two output FIR channel of the 10^{th} order with randomly chosen coefficients and a linear equalizer of length 9. The input data stream $\mathbf{s}_n$ is a random (i.i.d.) binary sequence and the block length is $size(\mathbf{Q})$=512.

In Figure 3.14 the noiseless case is presented. It is seen that the algorithm converges after three iterations only and the resulting error is $J_{CM} = 0.0128$. The

influence of the noise to the convergence is shown in the Figure 3.15 where the 15dB AWGN noise is added to the output of the channel. The algorithm converges again after three or four iterations but the residual cost $\mathbf{J}_i$ is higher ($J_{CM} = 0.0804$). In both cases the optimal step-size choice is used.

The results of "simplified" version of FI-CMA (3.25) are shown for the same cases (noise-free and 15dB AWGN) in Figures 3.16 and 3.17. As is seen from the evolution of the cost function, the convergence speed is somewhat slower, but, on the other hand, it gives lower residual error (to be concrete it gives $J_{CM} = 0.0063$ for the noiseless case and $J_{CM} = 0.0613$ for the case of 15dB AWGN). Also this simplified version profits from less computations, with result of lower round-off error.

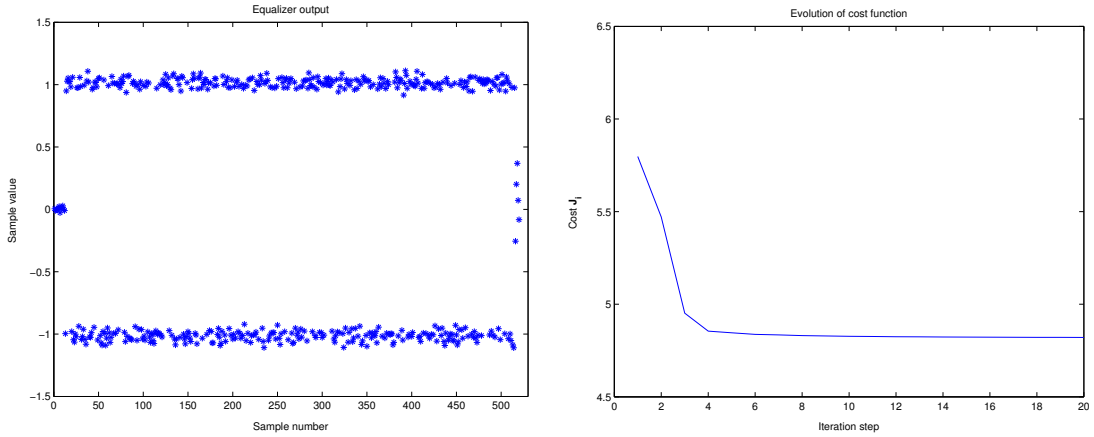


Figure 3.14: Evolution of cost function $\mathbf{J}_i$ and equalizer output for zero noise and static channel

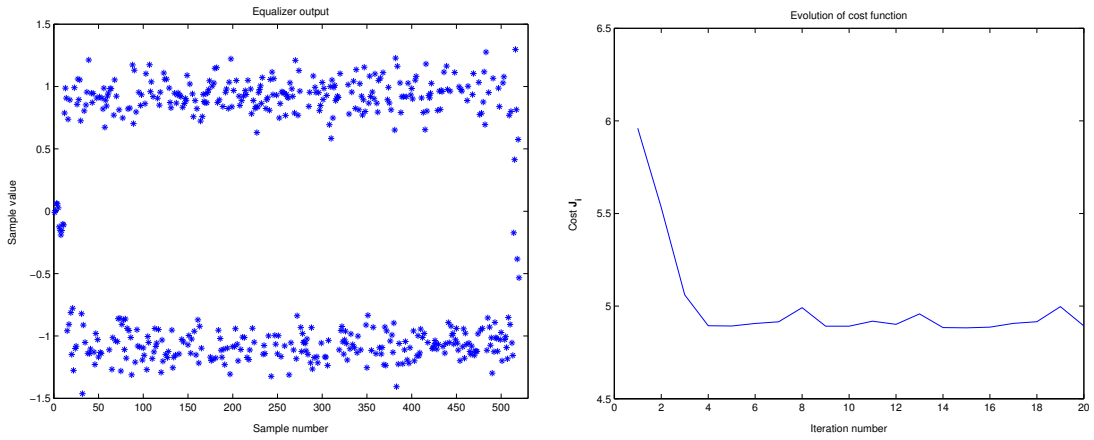


Figure 3.15: Evolution of cost function $\mathbf{J}_i$ and equalizer output for static channel and 15dB AWGN

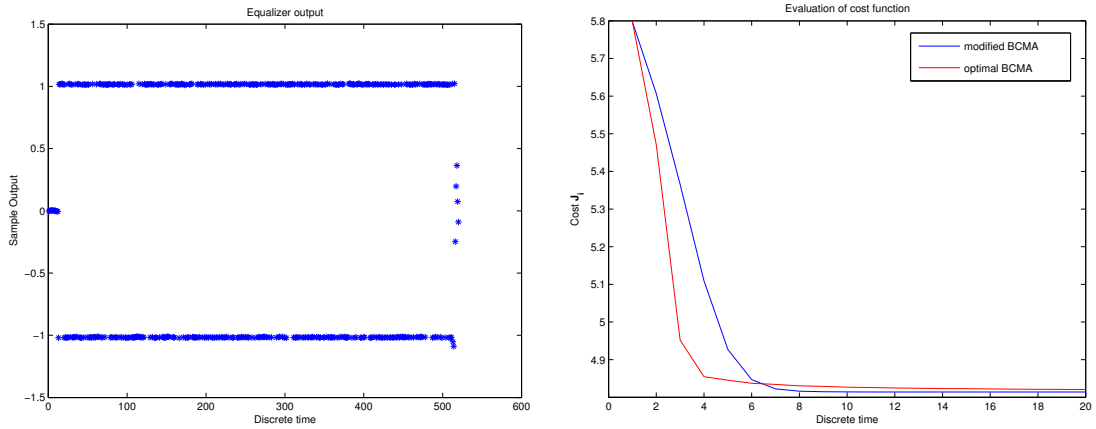


Figure 3.16: Evolution of cost function $\mathbf{J}_i$ and equalizer output of modified FI-CMA for zero noise and static channel

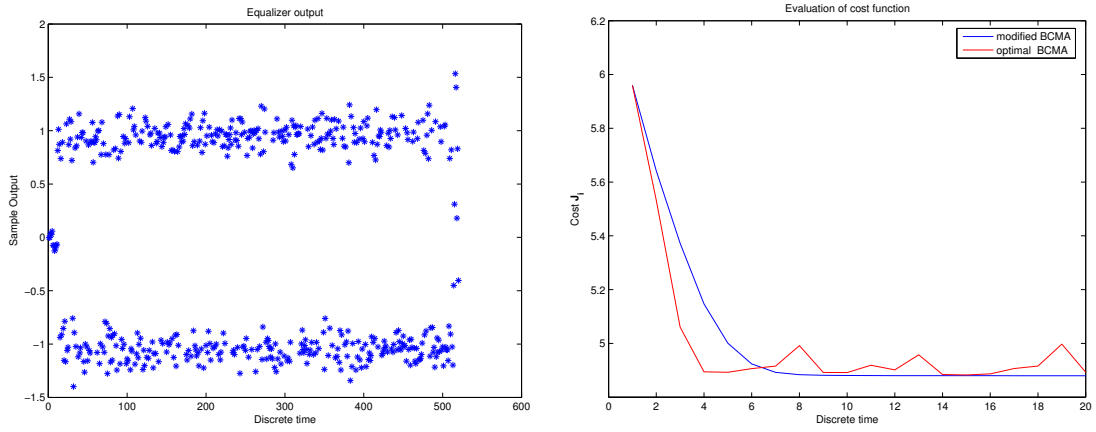


Figure 3.17: Evolution of cost function $\mathbf{J}_i$ and equalizer output of modified FI-CMA for static channel and 15dB AWGN

Chapter 4

Sliding window CMA

The finite interval CMA has the computational complexity significantly lower than the ACMA algorithm while it achieves faster convergence in comparison with the gradient type algorithms described in section 3.2. On the other hand, since it is a batch processing algorithm it can be used only for time-invariant or slowly varying channels (with respect to block length). In addition, since the solutions for two successive blocks do not have to correspond to the same system delay and phase, the blocks must overlap. This gives the need to run a pairing algorithm in the reconstruction of the data stream. This additional post-processing gives the requirement on N to be as large as possible. Unfortunately, bigger block size N results in higher computation complexity of the algorithm and requirement of a time-invariable channel. The growth of computational complexity with N is a critical property because the QR factorization algorithm has $\mathcal{O}(N^2M)$ complexity.

To benefit from the analysis of FI-CMA (namely the fact that there exists an analytical expression for the optimal step size) and its properties such as fast convergence and acceptable computational complexity we have derived its Sliding-window variant. It has the following properties:

- since it uses recursive type of processing, it does not require block overlapping and thus it reduces the additional post-processing

- it works for slowly varying environments

- it achieves the convergence rate of FI-CMA

- it enables the block size N to be as low as possible to reduce computation complexity per decoded symbol

The proposed algorithm uses the modified cost function (3.15) where the iteration index was replaced by the discrete time k of the form:

$$\mathbf{J}_k = \sum_{i=0}^{N-1} \left(|y_{k-i}|^2 - 1 \right)^2 = \sum_{i=0}^{N-1} \left(\left| \mathbf{r}_{k-i}^T \mathbf{f}_k \right|^2 - 1 \right)^2$$

where $\mathbf{f}_k$ is an equalizer coefficient vector at time k. Let us recall that in the FI-CMA algorithm, the $\mathbf{Q}$ matrix (result of the QR decomposition) is used, instead of the input data matrix. Hence, we can update the QR decomposition directly from the input data. Depending on the structure of the input data matrix $\mathcal{R}$, it is possible to use either row update or column update of the QR decomposition. Both variants will be presented in following sections.

4.1 Sliding Window CMA with row update

Following the derivation of the FI-CMA algorithm, two successive input data matrices may be written as:

$$\mathcal{R}_k = \begin{bmatrix} \mathbf{r}_{k-N+1}^T \\ \mathbf{r}_{k-N+2}^T \\ \vdots \\ \mathbf{r}_{k-1}^T \\ \mathbf{r}_k^T \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{k-N+1}^T \\ \mathcal{R}_k^- \end{bmatrix}, \tag{4.1}$$

$$\mathcal{R}_{k+1} = \begin{bmatrix} \mathbf{r}_{k-N+2}^T \\ \vdots \\ \mathbf{r}_k^T \\ \mathbf{r}_{k+1}^T \end{bmatrix} = \begin{bmatrix} \mathcal{R}_k^- \\ \mathbf{r}_{k+1}^T \end{bmatrix} \tag{4.2}$$

where $\mathcal{R}_k^- = [\mathbf{r}_{k-N+2} \ \ldots \ \mathbf{r}_k]^T$ is a matrix shared by both matrices $\mathcal{R}_k$ and $\mathcal{R}_{k+1}$. Thus $\mathcal{R}_{k+1}$ could be viewed as up-down shifted matrix $\mathcal{R}_k$ with one row removed (top row) and one row added (bottom row).

The FI-CMA algorithm (3.24) resp. (3.25) uses the orthonormal matrix $\mathbf{Q}$ obtained by the QR-decomposition of matrix $\mathcal{R}$. Let $\mathbf{Q}_k$ be obtained by QR-decomposition of matrix $\mathcal{R}_k$. Then, matrix $\mathbf{Q}_{k+1}$ can be computed using $\mathbf{Q}_k$ and $\mathbf{r}_{k+1}$ by matrix factorization update, namely by appending and deleting a row of $\mathbf{Q}_k$ directly from the data vector $\mathbf{r}_{k+1}$. In sequel we describe these update algorithms.

4.1.1 Append/delete a row operations

Let us start by deleting a row of $\mathcal{R}_k$ directly from the QR factorization. Suppose we have $\mathbf{Q}_k\mathbf{R}_k = \mathcal{R}_k$, where $\mathbf{R}$ is $m \times n$ and $\mathbf{Q}$ is $m \times m$ with $m > n$, and we want to compute a factorization of matrix $\mathcal{R}_k^-$ as defined in (4.1). Let $\mathbf{q_1^T}$ be the first row of matrix $\mathbf{Q}_k$. Then calculating Givens rotation $\mathbf{G}_1 \ \ldots \ \mathbf{G}_{m-1}$ such that

$$\mathbf{G}_1^T \ \ldots \ \mathbf{G}_{m-1}^T \mathbf{q}_1 = \pm \mathbf{e}_1$$

where $\mathbf{e}_1 = [1\ 0\ \ldots\ 0]$, we obtain a modified matrix of the form:

$$\mathbf{Q}_k \mathbf{G}_{m-1}^T \ \ldots \ \mathbf{G}_1^T = \begin{bmatrix} \pm 1 & 0 \\ 0 & \mathbf{Q}_k^- \end{bmatrix} \tag{4.3}$$

where the matrix $\mathbf{Q}_k^-$ is the desired factorization since:

$$\mathcal{R}_k = \begin{bmatrix} \mathbf{r}_{k-N+1}^T \\ \mathcal{R}_k^- \end{bmatrix} = \begin{bmatrix} \pm 1 & 0 \\ 0 & \mathbf{Q}_k^- \end{bmatrix} \begin{bmatrix} * \\ \mathbf{R}_k^- \end{bmatrix}$$

where $*$ means a residual row vector and $\mathbf{R}_k^- = \mathbf{G}_1 \ \ldots \ \mathbf{G}_{m-1}\mathbf{R}_k$. It is straightforward that $\mathbf{Q}_k^-\mathbf{R}_k^- = \mathcal{R}_k^-$ is the desired "remove a row" factorization. The "append a row" operation updates the QR factorization from $\mathbf{Q}_k^-$ to $\mathbf{Q}_{k+1}$ by $\mathbf{r}_{k+1}$. Because

$$\begin{bmatrix} (\mathbf{Q}_\mathbf{k}^-)^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_\mathbf{k}^- \\ \mathbf{r}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathcal{R}_k^- \\ \mathbf{r}_{k+1} \end{bmatrix} = \overline{\mathbf{R}}_{k+1}$$

where $\mathbf{R}_k^-$ is upper triangular, the Givens rotations $\mathbf{G}_1^T \ldots \ \mathbf{G}_n^T$ can be calculated such that $\mathbf{R}_{k+1} = \mathbf{G}_1 \ \ldots \ \mathbf{G}_n\overline{\mathbf{R}}_{k+1}$ is again upper triangular. It follows that

$$\mathbf{Q}_{k+1} = \begin{bmatrix} (\mathbf{Q}_k^-)^T & 0 \\ 0 & 1 \end{bmatrix} \cdot \mathbf{G}_1 \ \ldots \ \mathbf{G}_n$$

is the desired factorization update, since $\mathcal{R}_{k+1} = \mathbf{Q}_{k+1}\mathbf{R}_{k+1}$. Note that the "row append" procedure has $\mathcal{O}(mn)$ complexity (it updates only the first n columns of $\mathbf{Q}$), while the "row delete" operation has complexity $\mathcal{O}(n^2)$ (it is necessary to update the whole matrix $\mathbf{Q}$).

4.1.2 Algorithm summary

Using the row factorization update algorithms described above, it is possible to design a recursive form of the algorithm (3.24) where, after update of the QR-factorization, one or more iterations of equalizer update (3.24) is performed. The proposed algorithm consists of the steps summarized in Figure 4.1.

The steps 3 and 5 follow from the fact that the equalizer update procedure (3.24) updates the coefficients $\mathbf{w} = \mathbf{R}\mathbf{g}$ since the matrix $\mathbf{R}$ is affected by the QR-factorization update, $\mathbf{w}$ is affected as well. However, we can expect that $\mathbf{R}$ does not change significantly, then $\mathbf{g}$ does not have to be calculated and steps 3 and 5 can be skipped.

1. System initialization:

$$\mathbf{Q}_0 = \begin{bmatrix} \mathbf{I}_{(\mathbf{P+1})\mathbf{M}} \\ \mathbf{0} \end{bmatrix}, \qquad \mathbf{R}_0 = \mathbf{I}_{(P+1)M}, \qquad \mathbf{g}_0 = \mathbf{e}_\nu$$

2. Update QR-factorization by $\mathbf{r}_k$ using the append/delete row routine
3. Calculate transformed coefficient vector $\mathbf{w}_k = \mathbf{R}_k \mathbf{g}_k$
4. Calculate iterations of the equalizer update procedure (3.24) to obtain $\mathbf{w}_{k+1}$
5. Calculate updated equalizer coefficient vector $\mathbf{g}_{k+1} = \mathbf{R}_k^{-1} \mathbf{w}_{k+1}$
6. repeat steps $2. - 5.$ until the end of data stream

Figure 4.1: Sliding window CMA summary - row factorization

4.2 Sliding Window CMA with column update

Thanks to the structure of the matrix $\mathcal{R}_k$, it can be structured column wise. For the purpose of this section, we define the output signal of SIMO channel $\mathbf{H}$ at the time k to be:

$$\mathbf{r}_k = \mathbf{H}\mathbf{s}_n + \mathbf{w}_n$$

where $\mathbf{H}$ is $N_h \times P$ matrix and $\mathbf{r}_k$ is $1 \times P$ vector. Then define the matrix $\hat{\mathbf{r}}_k \in \mathbf{R}^{N \times P}$ collecting N successive inputs as :

$$\hat{\mathbf{r}}_k = \left[\mathbf{r}_{k-N} \ \ldots \ \mathbf{r}_{k-1} \ \mathbf{r}_k\right]^T$$

Then the matrices $\mathcal{R}_k$ and $\mathcal{R}_{k+1}$ can be structured as follows:

$$\begin{aligned} \mathcal{R}_k &= [\hat{\mathbf{r}}_k \ \hat{\mathbf{r}}_{k-1} \ \ldots \ \hat{\mathbf{r}}_{k-N+1} \,] = \left[\mathcal{R}_k^- \ \hat{\mathbf{r}}_{k-N+1}\right] \\ \mathcal{R}_{k+1} &= [\hat{\mathbf{r}}_{k+1} \ \hat{\mathbf{r}}_k \ \ldots \ \hat{\mathbf{r}}_{k-N+2} \,] = \left[\hat{\mathbf{r}}_{k+1} \ \mathcal{R}_k^-\right] \end{aligned}$$

The data matrices $\mathcal{R}_k$ and $\mathcal{R}_{k+1}$ share the same sub-matrix (as in the previews case) and $\mathcal{R}_{k+1}$ can be viewed as left-to-right shifted version of the matrix $\mathcal{R}_k$ with

P columns added to its left side and P columns deleted in its right sight. The QR decomposition can be updated directly from input data matrix $\hat{\mathbf{r}}_k$ using the appending/deleting a column update algorithms. These algorithms are described in the next subsection.

4.2.1 Append/delete a column operations

Let us start by deleting a column from the QR factorization. Suppose we have $\mathbf{Q}_k\mathbf{R}_k = \mathcal{R}_k$ and we want to compute a factorization of matrix $\mathcal{R}_k^- = \hat{\mathbf{Q}}_k\hat{\mathbf{R}}_k$. Since, the column to be removed is the last column of matrix $\mathcal{R}_k$, we can simply use the sub-matrices $\hat{\mathbf{Q}}_k = \mathbf{Q}_k(1:M, 1:N-1)$ and $\hat{\mathbf{R}}_k = \mathbf{R}_k(1:N-1, 1:N-1)$ without any need of matrix rotations.

Now, we want to compute the QR factorization of matrix $\mathcal{R}_{k+1}$ from the matrix $\mathcal{R}_k^-$ by adding vector $\mathbf{x}$ in place of the left most column. Since $\hat{\mathbf{Q}}_k^T\mathbf{R}_{k+1}$ is upper triangular except for the first column, it is possible to determine Givens rotations $G_{m-1} \ \ldots \ G_1$ such that

$$G_{m-1}^T \ \ldots \ G_1^T\mathbf{x} = \begin{bmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $\mathbf{x} = \hat{\mathbf{Q}}_k^T\hat{\mathbf{r}}_{k+1}$. Note that, in our special case, the "remove column" procedure does not require any calculations and the "append column" procedure requires only $\mathcal{O}(nm)$ operations.

4.2.2 Algorithm summary

Using the column factorization update algorithms described above, we obtain similar algorithm as described in previous section. Again, after column update of the QR-factorization, one or more iterations of the equalizer update (3.24) are performed. The algorithm is summarized in Figure 4.2.

1. System initialization:

$$\mathbf{Q}_0 = \begin{bmatrix} \mathbf{I}_{(\mathbf{P+1})\mathbf{M}} \\ \mathbf{0} \end{bmatrix}, \qquad \mathbf{R}_0 = \mathbf{I}_{(P+1)M}, \qquad \mathbf{g}_0 = \mathbf{e}_\nu$$

2. Update QR-factorization by $\hat{\mathbf{r}}_k$ using the append/delete column routine
3. Calculate transformed coefficient vector $\mathbf{w}_k = \mathbf{R}_k\mathbf{g}_k$
4. Calculate iterations of the equalizer update procedure (3.24) to obtain $\mathbf{w}_{k+1}$
5. Calculate updated equalizer coefficient vector $\mathbf{g}_{k+1} = \mathbf{R}_k^{-1}\mathbf{w}_{k+1}$
6. repeat steps $2. - 5.$ until the end of data stream

Figure 4.2: Sliding window CMA summary - column factorization

As in previews case, we can expect that $\mathbf{R}$ does not change significantly, then $\mathbf{g}$ does not have to be calculated and steps 3 and 5 can be skipped. Note, that in the case of row update, only one row is appended and one row is removed. In the case of column update, P columns are appended, where P is the oversampling factor. On the other hand, the delete column procedure does not require any calculations.

4.3 Error of QR factorization calculation

One may observe that calculating $\mathbf{Q}_k$ using the factorization update will produce the round-off error. Let $\Delta\mathcal{R}_k$ be the difference between the QR factorization of $\mathcal{R}_k$ obtained by direct QR calculation and that obtained by the QR factorization update:

$$\Delta\mathcal{R}_k = \ddot{\mathcal{R}}_k - \mathcal{R}_k = \ddot{\mathbf{Q}}_k\ddot{\mathbf{R}}_k - \mathbf{Q}_k\mathbf{R}_k \tag{4.4}$$

where $\ddot{\mathcal{R}}_k = \ddot{\mathbf{Q}}_k\ddot{\mathbf{R}}_k$ is calculated using the standard QR method and $\mathcal{R}_k = \mathbf{Q}_k\mathbf{R}_k$ is calculated using the factorization update procedure. Then, we define the error signal

as

$$e_k = \sum_{i,j} (\Delta \mathcal{R}_k)_{i,j} \tag{4.5}$$

where $(\Delta \mathcal{R}_k)_{i,j}$ means the $\mathrm{i,j}^{th}$ element of the matrix $\Delta \mathcal{R}_k$.

We define also the lost of the orthonormality of the $\mathbf{Q}$ matrix as:

$$\Delta Q_k = \sum_{i,j} (\mathbf{Q}_k^T \mathbf{Q}_k - I)_{i,j} \tag{4.6}$$

where I is a identity matrix of the same size as $\mathbf{Q}$, and the orthonormality loss error signal as:

$$o_k = \sum_{i,j} (\Delta Q_k)_{i,j} \tag{4.7}$$

Numerical experiments shows that the decomposition deviation error (4.5) , shown in Figure 4.3, has nearly zero mean value $E[e_k] = 2.4 \cdot 10^{-16}$ and variance $\sigma_e^2 = 1.1 \cdot 10^{-27}$. The orthonormality loss error is shown in Figure 4.4. The orthonormality loss function has again nearly zero mean value $E[o_k] = 4.4 \cdot 10^{-16}$ and variance $\sigma_o^2 = 1.8 \cdot 10^{-28}$. When compared to the orthonormality of the $\ddot{\mathbf{Q}}$ matrix with mean value $E[o_k] = 3.6 \cdot 10^{-17}$ and variance $\sigma_o^2 = 5.2 \cdot 10^{-30}$, the update version has this kind of error only a little bit higher.

It is possible to conclude that the error caused by the QR factorization update procedure does not accumulate and that it has only little influence on perturbation of matrix $\mathbf{Q}_k$, its orthonormality and the overall performance of the algorithm.

4.4 Simulation examples

The behavior of the sliding window CMA is shown in Figures 4.5 to 4.10. In these illustrative examples the equalizer output and criteria flow is shown for different scenarios. All presented results were obtained using double precision calculation in Matlab. We have considered a system with a single input/two output FIR channel of the 10^{th} order with randomly chosen coefficients and a linear equalizer of length 9. The input data stream $\mathbf{s}_n$ is a random binary i.i.d. sequence of the length 2000.

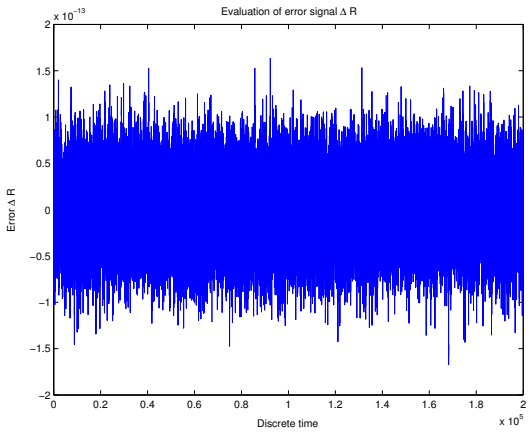


Figure 4.3: Evaluation of error signal of **QR** factorization update procedure

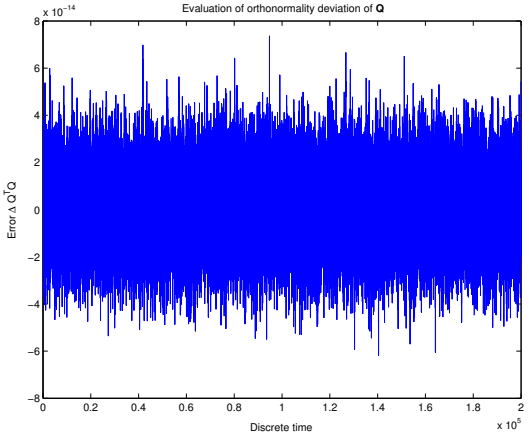


Figure 4.4: Evaluation of the orthonormality loss function

In the situations when the channel noise is present, the white Gaussian noise giving SNR=15dB is added to the output of the channel. In all cases the step size value was chosen to be $\mu_i = \frac{\sqrt{N}}{(\sqrt[4]{\mathbf{J}_i}-\sqrt{N}\mathbf{J}_i)}$ where N_Q is a length of the matrix $\mathbf{Q}$.

In Figures 4.5 and 4.6 the SW-CMA with column and row update are compared for two different block sizes $N_Q = N_f^2$ and $N_Q = 40$. The output of the sliding window FI-CMA equalizer with row update was multiplied by constant 0.9 for the better comparison in plot. It is seen, that for block size $N_Q = N_f^2$, the algorithms provides the same convergence speed and residual error. In the case of $N_Q = 40$, the column update version provides faster convergence than the row update version. This is caused by the fact, that column update version use input data matrix of the size $N_Q \times P$ whereas the row update version use input data vector of the size $N_f \times 1$. Because $N_Q > N_f$, the column update version fills the QR factorization matrices faster with valid data.

Since both variants of SW-CMA (row and column update) are equal, for clarity and better reading of graphs, we will use only the row update variant in sequel. In Figures 4.7 and 4.8 the effect of the additive Gaussian noise is presented. Figures 4.9 to 4.10 show the behavior of the simplified version of SW-CMA where steps 3. and 5. are skipped. The simplified version is presented in both noisy and noiseless cases,

for two different lengths N_Q.

We can conclude from the presented results that higher N_Q gives faster convergence both in the original- and in the simplified version. Lowering SNR (higher noise power) slows down the convergence which is more significant for lower N_Q, but in steady state the output of the equalizer is nearly the same. Skipping the steps 3 and 5 in the simplified SW-CMA has similar affect as an additive noise, and additional channel noise with SNR=15dB does not worsen the performance of the algorithm.

Algorithm	N_Q	SNR	J_{CM}
SW-CMA	N_f^2	∞	$1.5 \cdot 10^{-4}$
	40	∞	$6.5 \cdot 10^{-4}$
	N_f^2	15dB	$4.1 \cdot 10^{-2}$
	40	15dB	$4.5 \cdot 10^{-2}$
simpl. SW-CMA	N_f^2	∞	$1.5 \cdot 10^{-2}$
	40	∞	$3.3 \cdot 10^{-2}$
	N_f^2	15dB	$6.1 \cdot 10^{-2}$
	40	15dB	$7.7 \cdot 10^{-2}$

Table 4.1: Steady state J_{CM} error of SW-CMA and simplified SW-CMA

Comparing the values of cost J_{CM} in steady state (see Table 4.1) with gradient CMA and FI-CMA, we conclude that SW-CMA has similar performance properties as FI-CMA in both noiseless and noisy case, and that it outperforms the G-CMA in convergence speed as well as in the residual error cost. The simplified SW-CMA outperforms the G-CMA as well except for the short size $N_Q = 40$, where the G-CMA with well chosen step size gives lower residual J_{CM}. However, the optimal step size of G-CMA (and its performance) is dependent on the channel properties while the SW-CMA has its step size expressed analytically.

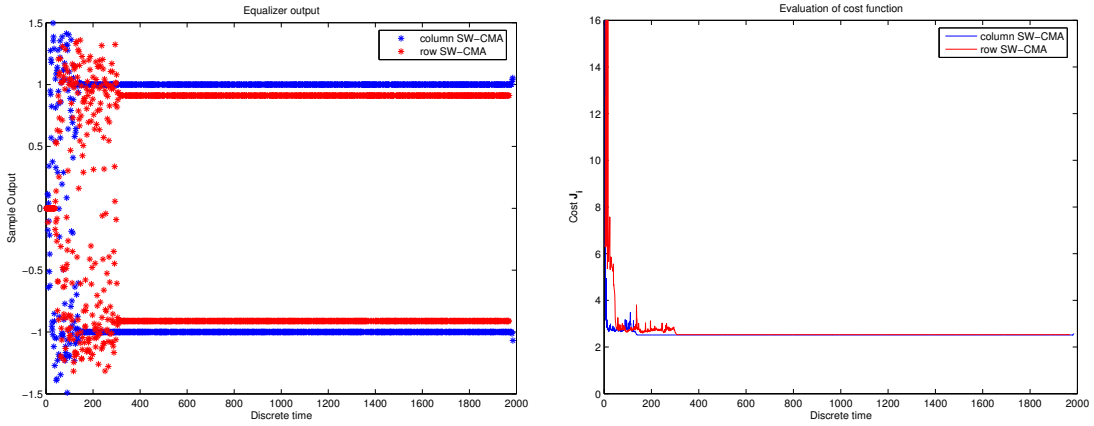


Figure 4.5: Example of criteria evaluation and equalizer output of column and row based update SW-CMA for the following parameters: SNR$\rightarrow \infty$, size($\mathbf{Q}$)=N_f^2

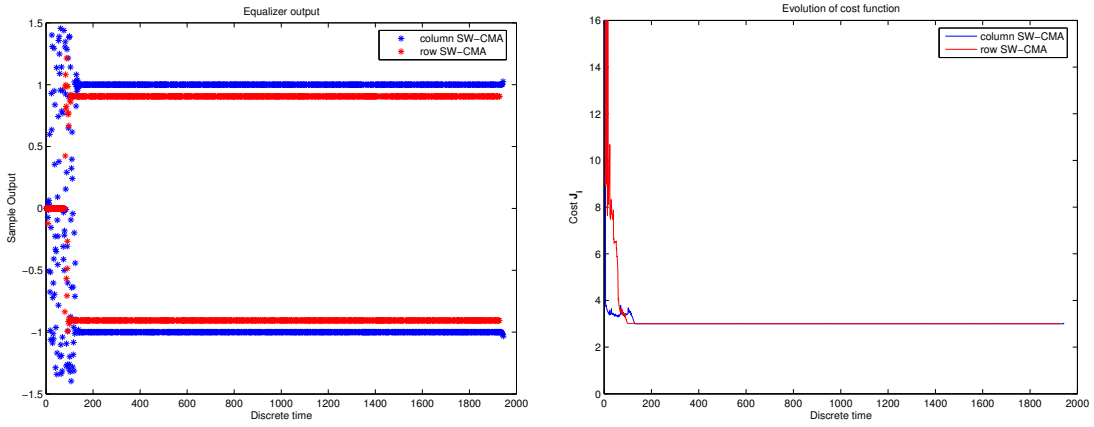


Figure 4.6: Example of criteria evaluation and equalizer output of column and row based update SW-CMA for the following parameters: SNR$\rightarrow \infty$, size($\mathbf{Q}$)=40

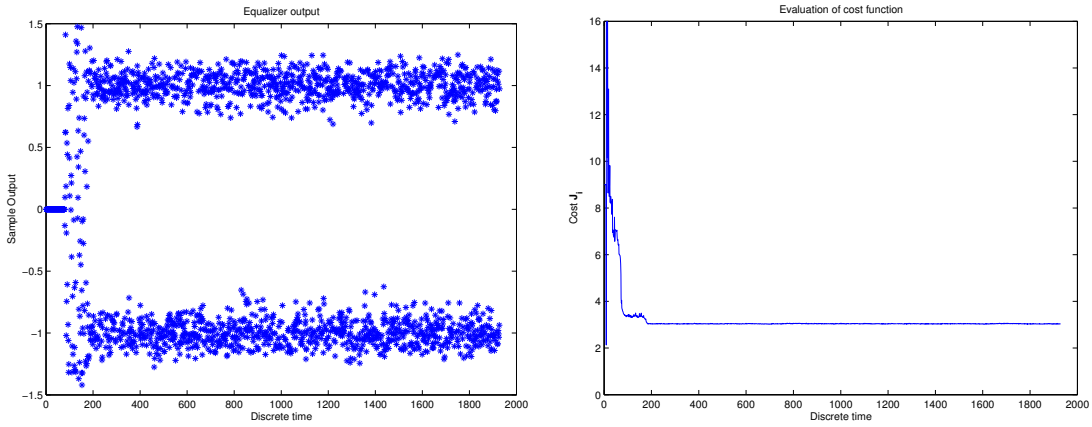


Figure 4.7: Example of criteria evaluation and equalizer output for the following parameters: SNR=15dB, size($\mathbf{Q}$)=N_f^2

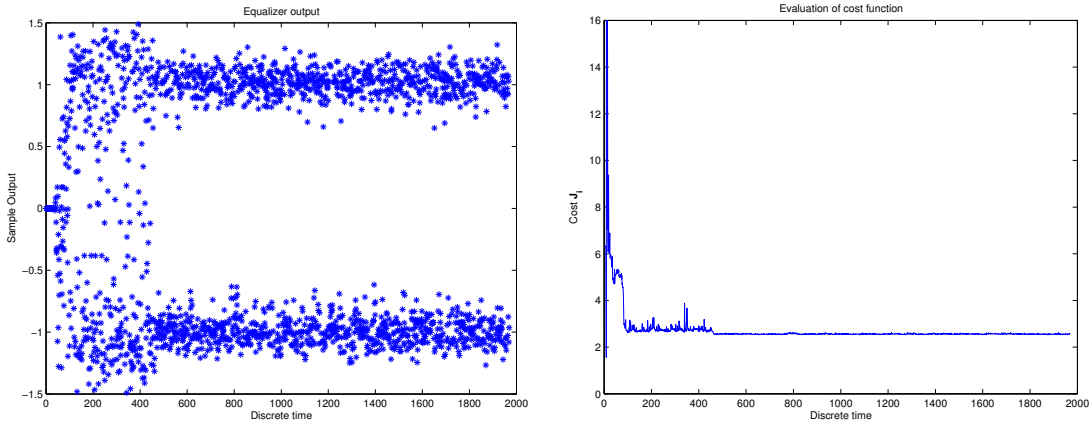


Figure 4.8: Example of criteria evaluation and equalizer output for the following parameters: SNR=15dB, size($\mathbf{Q}$)=40

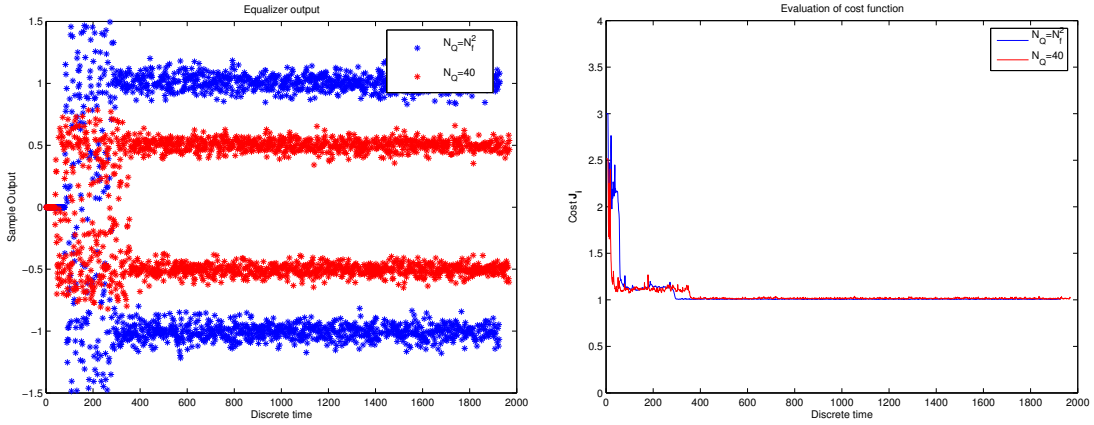


Figure 4.9: Example of criteria evaluation and equalizer output of simplified SW-CMA algorithm for the SNR$\rightarrow \infty$ and two different sizes($\mathbf{Q}$)

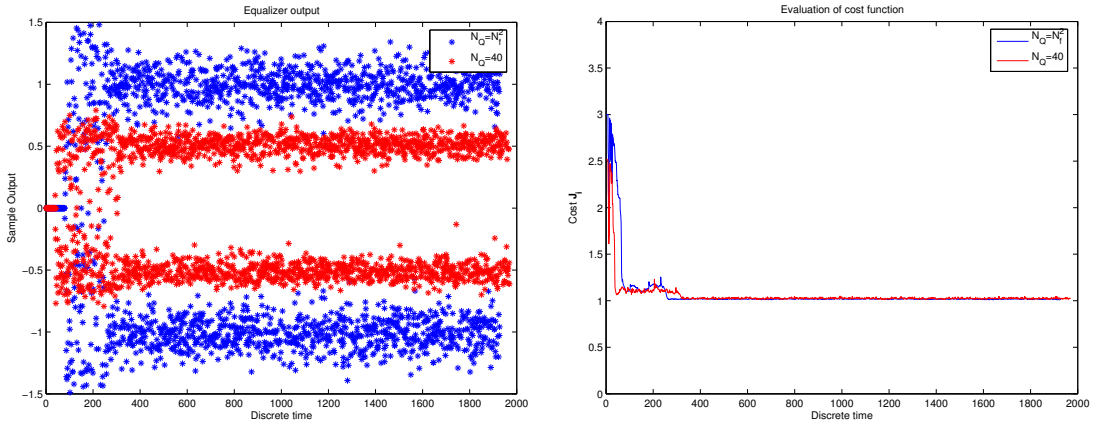


Figure 4.10: Example of criteria evaluation and equalizer output of the simplified SW-CMA algorithm for the SNR$\rightarrow$ 15db and two different sizes($\mathbf{Q}$)

Chapter 5

Performance of algorithms for time varying channels

In the previous sections, we have discussed the gradient CMA, FI-CMA and SW-CMA algorithms. Here, we shall compare their performance for channels with time varying parameters. Based on the results of simulations, five typical scenarios were selected.

5.1 Channel model

To describe the time variation of the system parameters, we have chosen the following channel model:

$$\mathbf{h}(n) = \mathbf{h}(0) + \alpha(n) \tag{5.1}$$

where $\mathbf{h}(n)$ is the channel coefficient vector in time n. The coefficient variation variable $\alpha(n)$ is defined as:

$$\alpha(n) = \lambda\alpha(n-1) + \omega b(n) \tag{5.2}$$

where $b(n)$ is a white random variable and λ and ω are constants. These constants influence the properties of time variations of the channel parameters, namely its variance and autocorrelation function.

	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
FI-CMA[1]	0.2743	0.1433	0.2482	0.8362	0.9247
FI-CMA[2]	0.0992	0.1074	0.2573	0.8700	0.9469
SW-CMA	0.0202	0.0164	0.0994	0.0693	0.1084
Simp. SW-CMA	0.1200	0.0596	0.0596	0.0978	0.1919
G-CMA	0.0630	0.4080	diverge	diverge	diverge

Table 5.1: Comparison of mean value of J_{CM} cost

5.2 Simulation experiments

Figures 5.1 to 5.3 show typical situations for different settings of parameters μ and λ. In all cases we use SIMO channels of the 10^{th} order with the oversampling factor of 2 and with all coefficients changing in time. The equalizers used in receiver are of the 9^{th} order. In Figures 5.1 to 5.3 the following plots are shown:

- output of the FI-CMA equalizer with optimal step size $\mu_1 = \frac{1}{\beta - 3}$
- output of the FI-CMA equalizer with step size $\mu_2 = \frac{\sqrt{N}}{\sqrt[4]{J_i} - \sqrt{N} J_i}$
- output of the SW-CMA equalizer with block size $N_Q = 40$ and step size $\mu = \frac{\sqrt{N}}{\sqrt[4]{J_i} - \sqrt{N} J_i}$
- output of the simplified SW-CMA equalizer with block size $N_Q = 40$ and step size $\mu = \frac{\sqrt{N}}{\sqrt[4]{J_i} - \sqrt{N} J_i}$
- output of gradient CMA with step size $\mu = 0.025$
- channel variations in time of two parameters.

The channels used in the experiments are the following:

Channel 1 is a channel generated with parameters $\lambda = 1$ and $\omega = 0.001$. It has small variance and nearly zero mean value. As is seen from Figure 5.1, all algorithms except the simplified SW-CMA converge acceptably and track the channel parameters

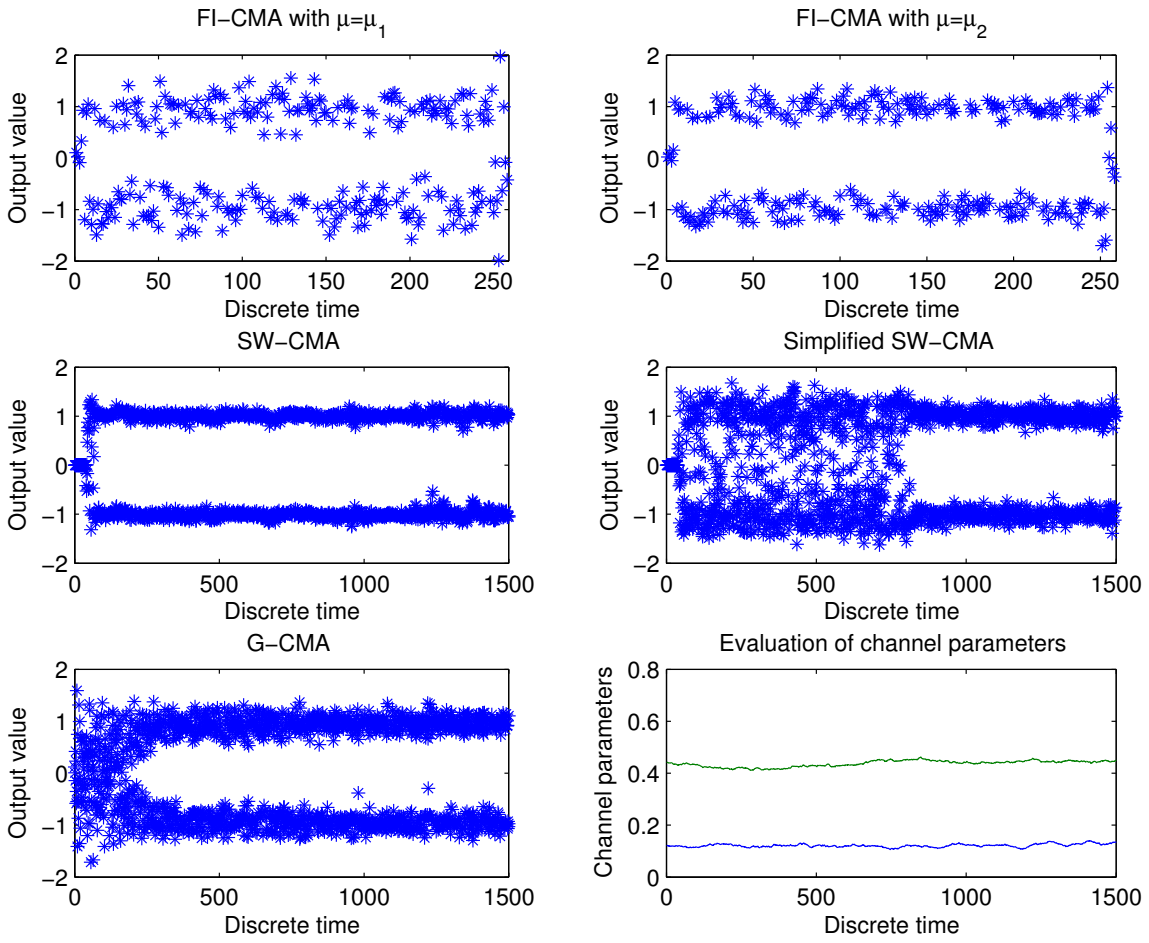


Figure 5.1: Comparison of algorithms for Channel 1 ($\lambda = 1$, $\omega = .001$)

successfully. Although the simplified SW-CMA converges somewhat slowlier than G-CMA and SW-CMA, after convergence (opening the ISI eye) it tracks the channel variations with similar performance. Comparing the mean value of CM cost (see Table 5.1), the best performance is achieved by the SW-CMA followed by G-CMA. Both settings of the block CM algorithm (FI-CMA with step size μ_1 or μ_2) work well and are able to deconvolve the channels of this character.

Channel 2 is a channel generated with parameters $\lambda = 0.999$ and $\omega = 0.005$. It represents channel with medium variations and non-zero mean value in short term.

Between the time steps 1300 and 1500, one of the parameter is decreasing nearly linearly with time. The difference between its values at the beginning and at the end of this decrease is approx. 0.2. Channel variations and the corresponding outputs of equalizers are shown in Figure 5.2.

In this case, all algorithms converge quickly and track well the channel variations till the time 1300. Then, with the start of the fast change of channel parameter, G-CMA collapses. Also, the block algorithms are not able to deconvolve the signal in this part (not shown in the graph). On the other hand, both versions of SW-CMA are able to track the channel variations, with the CM score $J_{CM} = 0.015$ for SW-CMA resp. $J_{CM} = 0.0596$ for the simplified SW-CMA (see Table 5.1).

Channel 3 This example represents the channel with small and slow variations but with non-zero mean value in short time interval. It was generated with parameters $\lambda = 1$ and $\omega = 0.005$. In this case, the G-CMA does not converge at all (probably because of non-optimal step size value) while all other algorithms provide good deconvolution of the input signal. Note that the simplified SW-CMA tracks the variations of channel parameter better than the standard version of algorithm. The FI-CMA performs also reasonably. See Table 4.1 and Figure 5.3 for comparison of algorithm performance.

Channel 4 Figure 5.4 shows the results for the channel with fast time variations with relatively high variance (parameters $\lambda = 0.99$ and $\omega = 0.01$). In this case, FI-CMA and G-CMA are not able to equalize. SW-CMA exhibits somewhat longer convergence than simplified SW-CMA, but once it has converged, it is able to track the channel evolution and gives lower residual cost J_{CM}.

Channel 5 The final example, shown in Figure 5.5, represents channel with very fast variations (parameters $\lambda = 1$ and $\omega = 0.01$). In this case, none of algorithms performs well. G-CMA diverges, and both versions of FI-CMA are not able to find the optimal solution. However, both SW-CMA algorithms open the ISI eye in some time intervals. As is seen from the graphs, the SW-CMA achieves the best performance

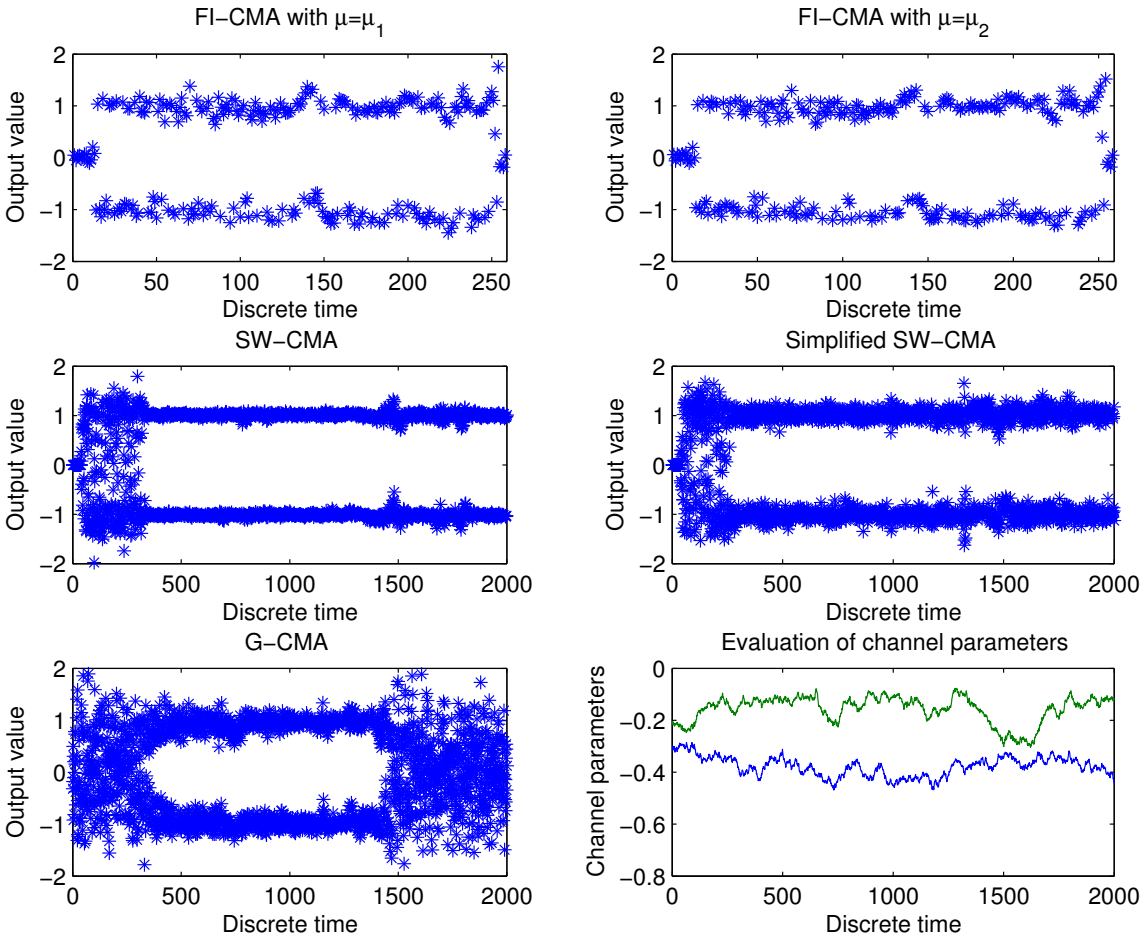


Figure 5.2: Comparison of algorithms for Channel 2 ($\lambda = 0.999\ \omega = 0.005$)

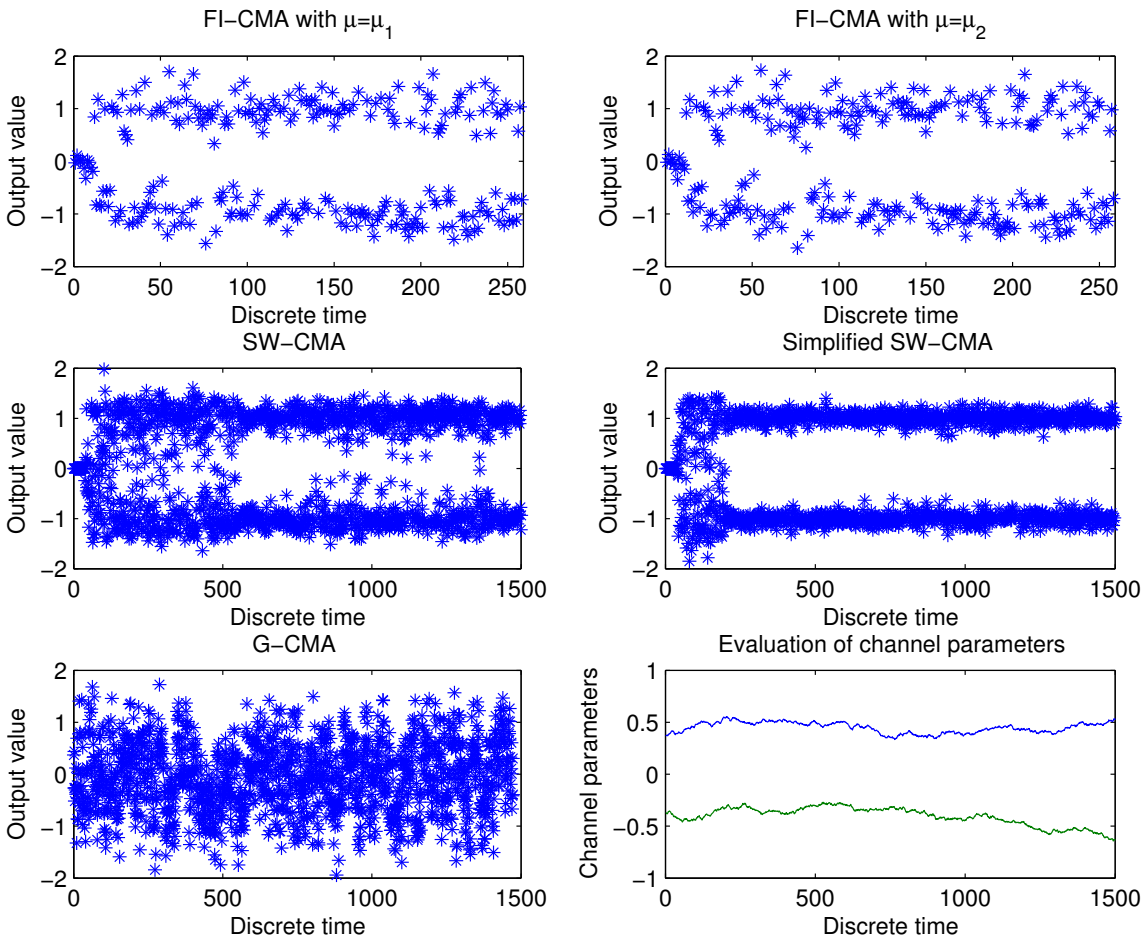


Figure 5.3: Comparison of algorithms for Channel 3 ($\lambda = 1$ $\omega = 0.005$)

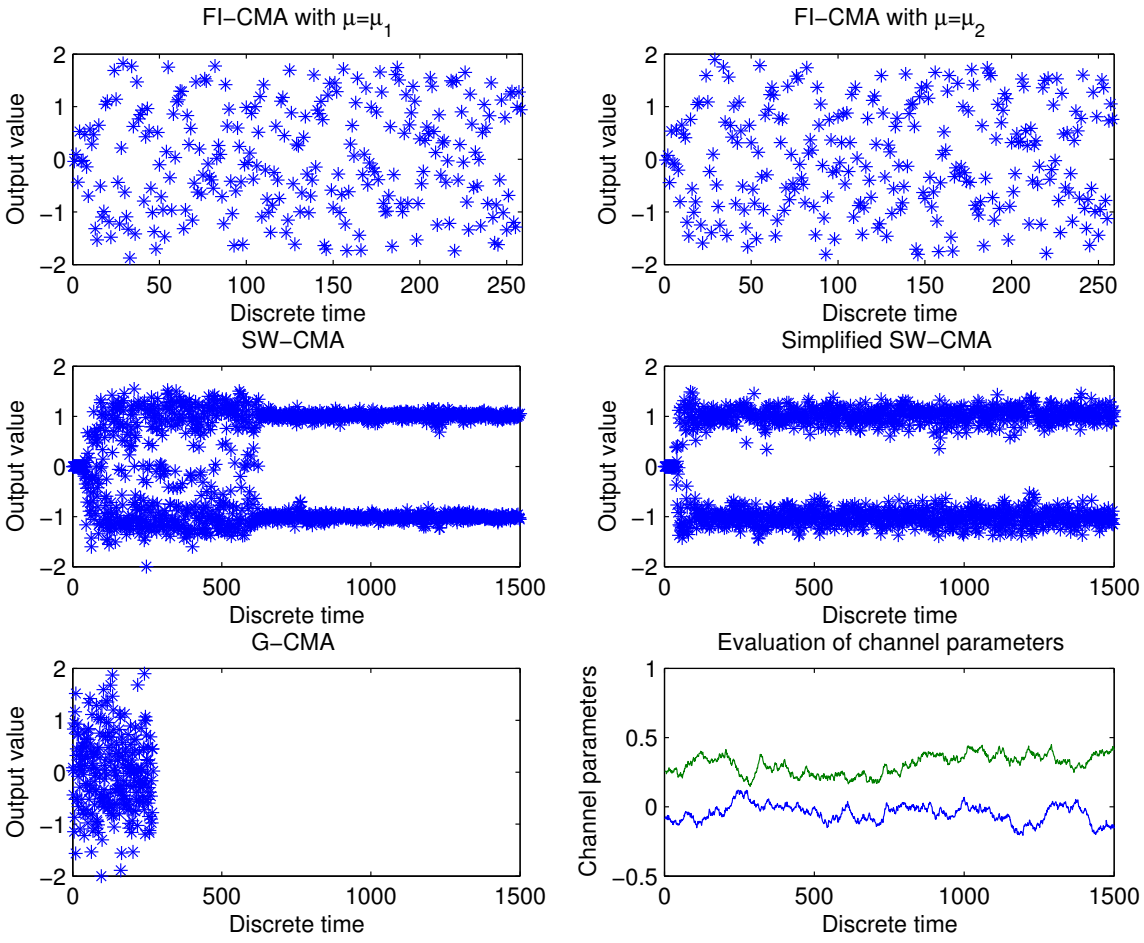


Figure 5.4: Comparison of algorithms for Channel 4 ($\lambda = 0.99\ \omega = 0.01$)

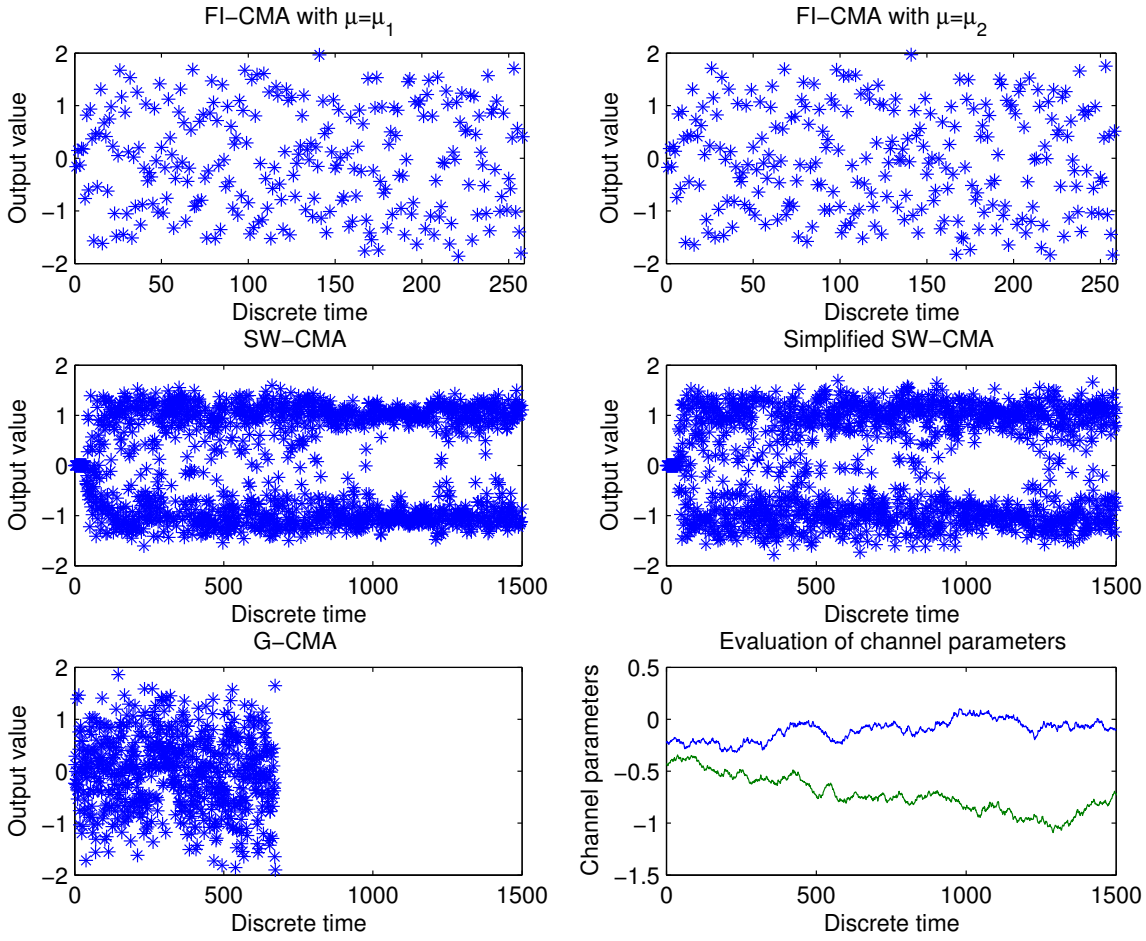


Figure 5.5: Comparison of algorithms for Channel 5 ($\lambda = 1\ \omega = 0.01$)

(see Table 5.1) and after 1500 steps both SW-CMA track the time variations of the channels parameters.

5.3 Conclusion

It is possible to conclude that the SW-CMA performs well in nearly all presented cases and that both the original and the simplified SW-CMA can successfully equalize channels with time varying parameters. FI-CMA can cope with the time varying channels if the changes have small variations and if the short-term mean value is

close to zero. Finally, the G-CMA is most sensitive to the channel variations. Since, for this algorithm, the value of the optimal step size is dependent on the channel parameters, time variations of these parameters can result in unstable behaviour of an algorithm that has already converged.

Chapter 6

Hardware implementation issues

In this chapter, the hardware platforms for implementation of the advanced DSP algorithms will be reviewed and the issues of implementing algorithms using the floating-point arithmetic will be discussed.

The structure of the chapter is the following: first, the hardware platforms for digital signal processing will be reviewed, including brief history of their evolution. Then, selected floating-point arithmetics will be discussed (since the algorithms covered by the thesis require use of floating-point computations). The IEEE implementation, used in the Texas Instruments DSP processors, will be reviewed very briefly, and the logarithmic arithmetic, which has been used for the FPGA implementation, will be discussed in more detail.

6.1 Brief history of the DSP platforms

Many tasks solved in telecommunications, such as equalization, audio and video coding or detection are computationally demanding applications with real-time computational requirements, which are not offered by general-purpose programmable processors. To tackle with these requirements, hardware architectures must offer, in the same time, both sufficient data throughput and computational capacity. The parameters offered by the standard processors are insufficient for these applications. The

implementation platforms which fulfil these demands can be divided into two groups: dedicated DSP processors and hardware solutions.

In 80's the first digital signal processor (DSP) was introduced to the market. In comparison with the general-purpose processor, it was designed to perform several operations in one instruction cycle. The architecture of a typical DSP is optimized for operations such as multiply-and-accumulate , and it contains dedicated units for operations such as barrel shift, address operations, etc. Typically, it has lower power consumption, and slower clock than standard processor.

The options for hardware implementation are represented by application specific circuits (ASIC) and by programmable hardware devices (CPLD, FPGA). The ASIC architectures reach the required performance parameters by using computing structures optimized for the application problem of interest. Nevertheless, even though the ASIC technology offers the highest performance and the lowest power consumption,when considering its application, the (high) design costs have to be weighted against the production volume. Finally, it shows up as a significant disadvantage, that any system update, demanded by an adaptation of new function or protocol or simply because of an exposed function error, becomes very expensive.

The economic factors limiting utilization of a hardware solution can be overcome by exploiting the reconfigurability of SRAM-based FPGAs. The FPGA devices provide unlimited reconfiguration of their logic resources. Through circuit reconfiguration, a wide variety of application-specific circuits can be configured and reconfigured within the same FPGA resource. Unlike the custom VLSI architectures that add general-purpose features to preserve flexibility, FPGA based systems provide flexibility in the form of circuit reconfiguration. The functionality of an FPGA is governed by bits written into its configuration memory. Since the configuration is loaded to FPGA from an external memory at the system start up, system update can be performed by a simple download of a configuration data stream into this memory.

It should be noted for completeness that some devices support modifications of their configuration while an application is running without affecting their unmodified

parts (so called partial dynamic reconfiguration). Depending on a device architecture, the configuration memory can store one or more configuration contexts which can be switched quickly during the application lifetime.

In sequel, we shall describe briefly the properties of the DPS and FPGA devices in general and the architecture of devices, on which the final implementation was tested.

6.2 DSP processor architecture

There are many types of DSP processors in the market and although the architecture differs from piece to piece, they share some common features. All of them are optimized for an efficient calculation of the vector/matrix multiplication, to implement digital filters and/or to calculate the FFT. A typical DSP device has the following properties:

- it implements the Multiply-and-Accumulate (MAC) unit (it can perform multiplication and addition in parallel)
- it supports extended addressing modes such as circular buffer and bit-reversal addressing
- it contains specialized units for address generation
- it provides arithmetic modes with saturation
- it has fast (often dual-access) internal memory
- it has DMA unit(s) for an effective data management

For the implementations described in this thesis, the TMS320C6711 device from Texas Instruments was used. It is a high performance floating-point DSP with a RISC-like VLIW (Very Long Instruction Word) architecture. The device executes up to eight instructions per cycle. Its core consists of 8 functional units which operate on 32 general-purpose registers of 32-bit word length. These 8 functional units contain

two multipliers and 8 arithmetic logic units. The block diagram of the device is shown in Figure 6.1.

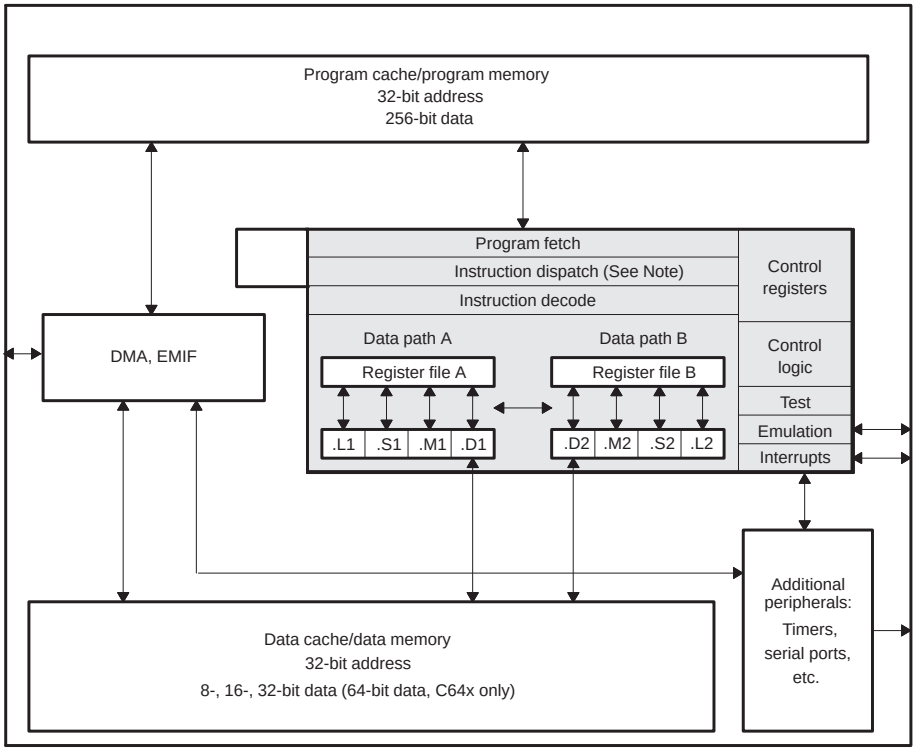


Figure 6.1: TMS320C6711 block diagram

The Program fetch, Instruction dispatch, and Instruction decode units can deliver up to eight 32-bit instructions to the functional units every CPU clock cycle. The instructions are processed in both data paths (A and B), each of which contains four functional units and 16 general purpose registers. The device uses multi-stage pipeline with three main parts: fetch, decode and execute, where each of these parts consists of several stages. The fetch and decode part takes every time the same number of cycles (unless the hold state, used for reading from external memory, occurs), while the execution part takes from one to ten cycles. Most instructions are executed in one cycle (see the documentation [61] for a detailed information about the pipeline structure). The VLIW architecture (called VelocityTI) used by the device has the following advanced features:

- Instruction packing, which reduces the code size
- Fully pipelined branches, which enable the zero overhead branching
- Conditional execution of all instructions

6.3 FPGA architecture

An advantage of the FPGA architecture for the DSP applications is fine-grain internal parallelism that can be used to achieve high data throughput.

The structure of an FPGA device can be divided into two parts. The application is build using a matrix of logic resources and eventually by custom hardware units. The configuration of the logic matrix, of the macros, and of their interconnections is done through the interface of the configuration memory.

The application running on the FPGA is executed using a logic resource matrix. It is a regular structure of basic logic elements, so called logic blocks (LBs), hardware macros (block memories, multipliers etc.), and of their interconnections.

Concrete structure of the logic block varies from one device to another. In general, it contains a function generator, a storage element, and some additional logic. The function generator is usually implemented using an n-input look-up table (LUT). The storage element can be configured to any type of register or to a level-sensitive latch. Several logic blocks are usually grouped together using a fast direct interconnect allowing efficient implementation of more complex logic functions.

The LB matrix is designed as a general-purpose regular structure that can implement any digital circuit. Since some circuits fit well into this structure while others can be implemented only with a significant performance degradation, FPGA vendors have introduced also some dedicated hardware structures into their chips (multipliers, dual-ported block memories, dedicated DPS blocks). Some hardware macros implement functions that cannot be implemented sing FPGA technology such as digital (phase)-locked loop.

XCV2000E	
System Gates	2,541,952
Logic Gates	518,400
CLB Array	80 x 120
Logic Cells	43,200
BlockRAM Bits	655,360
Distributed RAM Bits	614,400
Differential I/O Pairs	344
User I/O	804

Table 6.1: Xilinx Virtex2000E parameters

The routing structure is a very important part of an FPGA, because the configurable interconnect resources introduce additional delays to logic delays. These delays can exceed 60% of complete logic path delays. Modern FPGAs provide a hierarchical interconnect structure combined with some configurable switches to increase routability and decrease delays. Some vendors have introduced special purpose routing resources such as global clock networks, global set/reset networks, carry-chains, and others.

An FPGA configuration is kept in an SRAM-based configuration memory with very special structure. Each element of this memory affects the configuration of the logic matrix and interconnect, while it can be addressed by the configuration interface.

Xilinx Virtex2000-E

In 1998, the Xilinx Corp. introduced to the market the high-density FPGA family named Virtex. Later, a newer version of this family called Virtex-E was introduced to provide even bigger devices with more on-chip block memories. To give an example, the parameters of Virtex2000E are summarized in Table 6.1.

The logic bloc of the Virtex-E architecture is referred as a *slice*. It includes two four-input LUT-based function generators, two configurable storage elements, and a carry logic. Two slices grouped together shape a *configurable logic block* (CLB)

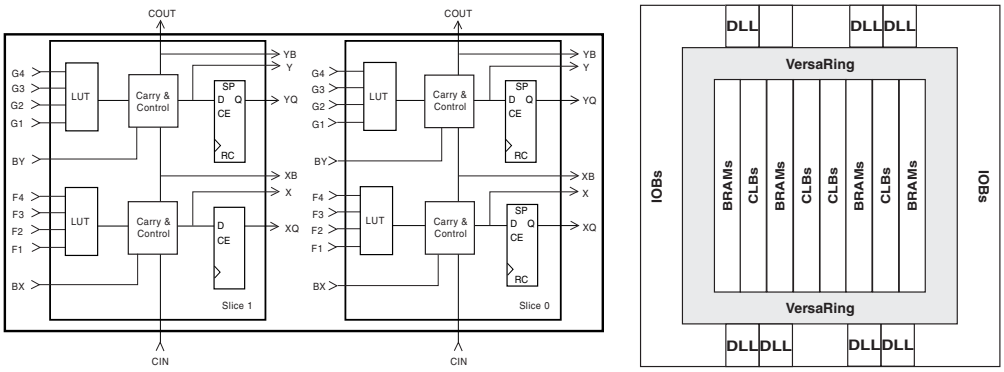


Figure 6.2: Virtex-E configurable logic block structure and hardware resources on the chip

which is the basic building block of the logic matrix. Each CLB contains two tri-state buffers that can drive the on-chip buses. The Virtex-E devices provide dual-port SRAM blocks (so-called BlockRAMs) to implement efficiently the on-chip memory. The I/O pads surrounding the CLB matrix are integrating three storage elements to decrease delay of incoming/outgoing signal paths.

The Virtex-E routing resources are hierarchical. They range from local interconnect resources provide interlan CL connections, direct high-speed connection between horizontally adjacent CLBs, and interconnection between the CLBs and the general routing matrix (GRM) to the general purpose routing, completed finally with horizontal routing resoures to implement three state buffers, dedicated nets to propagate carry signals to vertically adjacent CLBs and some other routing resources.

The structure of the configuration memory of Virtex-E is different from the structure of the logic matrix. The smallest reconfigurable amount is one frame, which refers to a one-bit-wide column of the configuration memory. 48 frames have to be reconfigured to modify one CLB column.

Further details go beyond the scope of this thesis and can be found in [79] data sheets.

6.4 Arithmetic

The choice of the arithmetic used to implement a DSP algorithm is of great importance. An integer- or fixed-point arithmetic is simple to implement and there are many devices using this type of arithmetic. On the other hand, only limited number of signal processing algorithms can be implemented in this arithmetic, since its dynamic data range is relatively small. Furthermore, to design a fixed-point algorithm is relatively expensive.

Most of the DSP algorithms use the floating point numbers. The devices using this arithmetic are usually more expensive than the fixed point devices, because the hardware of the floating-point units requires more logic thus it consumes more silicon. In our implementation we have used the floating point DSP and the Xilinx FPGA. For the DSP implementation the native IEEE floating point was used, for the FPGA design the alternative Logarithmic Numbering System (LNS). Since the IEEE floating point represents the standard choice, it will be covered only briefly in the next subsection, while the LNS arithmetic will be described in more detail, since it is relatively unknown solution.

6.4.1 IEEE Floating Point

The IEEE standard 754, published in 1985, defines a binary floating point arithmetic system. It specifies the floating point number formats, the results of the basic operations (including the square root operation), rounding modes and floating point exceptions. Two main data formats are defined: single and double precision (see Table 6.2). In both formats, one bit is used as a sign bit. Since the standard uses normalized numbers, the most significant bit is always 1 and is not stored (except for the denormalized numbers - see [23] for more details). Figure 6.3 shows the bit structure of the number representation and Table 6.2 shows the parameters for single and double precision.

In the implementation using the DSP processor, the floating-point device was

Precision	Size	Mantisa	Exponent	Unit roundoff	Range
Single	32	23	8	$\approx 5.9 \cdot 10^{-8}$	$10^{\pm 38}$
Double	64	52	11	$\approx 1.1 \cdot 10^{-16}$	$10^{\pm 308}$

Table 6.2: Parameters of IEEE floating point number representation

Figure 6.3: Floating point format

used, which natively uses the IEEE floating point. For the FPGA devices there exist so-called Intellectual Property (IP) cores, implementing the basic operations in floating point. Most of them, however, do not use the IEEE standard. Their parameters (maximal clock speed, latency and chip size) vary significantly between implementations and are platform-dependent. An illustrative example of a pipelined solution, optimized for the Xilinx Virtex devices, is shown in Table 6.3.

6.4.2 Logarithmic Numbering System

For the FPGA implementation of the algorithms treated in this thesis, we have investigated the use of a logarithmic number system (LNS) representation as an alternative to the IEEE floating point arithmetic. In our designs, we have used the technique developed in [5] to provide accuracy comparable with the IEEE single precision floating point. It will be described in sequel.

The LNS data representation consists of the two's complement fixed point value

	32-bit/20-bit			
Operation	Slices	% of chip	Frequency	Latency
add/sub	420/208	2%/1%	130/153	12/11
mul	326/171	2%/1%	122/125	6/6
div	711/348	4%/2%	142/166	27/19
sqrt	780/620	4%/3%	146/173	27/19

Table 6.3: Parameters of selected floating point IP cores optimized for Xilinx Virtex devices

equal to $\log_2 |x|$, where x is the value to be represented and $|\cdot|$ is the absolute value operator and of the sign bit at the MSB position (denoted by s in Figure 6.4). Base-2 logarithms are used, though in principle any base could be used. The signed two's complement fixed point number is divided to an integer part, which is always 8-bits long, and to the fractional part, the size of which depends on the data precision. The format of 32- and 19-bit LNS numbers is depicted in Figure 6.4.

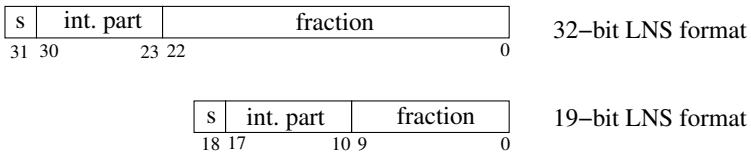


Figure 6.4: The 32- and 19-bit LNS data format

Thanks to the 8-bit integer part, both representations have the same dynamic range $\approx$ 2.9×10^{-39} to 3.4×10^{38}. The 32-bit LNS precision is comparable to the standard 32-bit floating-point data representation while the 19-bit LNS precision is comparable to the 16-bit FLP formats used on some commercial DSP devices. Special values are used to represent the value zero and other exceptional values such as Not-A-Number, Overflow etc.

For two LNS values $i = \log_2 |x|$ and $j = \log_2 |y|$, the LNS arithmetic involves the following computations:

$$\log_2(x+y) = i + \log_2(1 + 2^{j-i}) \tag{6.1}$$

$$\log_2(x-y) = i + \log_2(1 - 2^{j-i}) \tag{6.2}$$

$$\log_2(x * y) = i + j \tag{6.3}$$

$$\log_2(\frac{x}{y}) = i - j \tag{6.4}$$

$$\log_2(\sqrt{x}) = \frac{i}{2} \tag{6.5}$$

where in (6.1) and (6.2) we choose $j \leq i$ without loss of generality. Also, we assume the

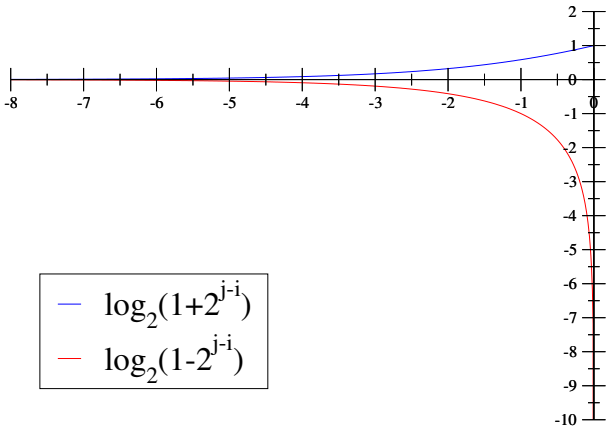


Figure 6.5: LNS sum and difference function

values x and y to be positive. In all these cases the sign bits are handled separately by a logic that is simple. Note that the multiplication, division and square-root operations can be implemented simply as fixed point addition, subtraction, and right shift operations (See equations (6.3), (6.4) and (6.5)). The weak point of the LNS arithmetic lies in implementation of (6.1) and (6.2) which require evaluation of a non-linear function

$$\mathcal{F}(r) = \log_2(1 \pm 2^r), \qquad r = j - i$$

which is shown in Figure 6.5. The solution of this problem was first solved in [59], where a look-up table was used to cover all possible values of equations (6.1) and (6.2). It is apparent that as the word length increases, the table sizes increase exponentially, which limits the practical utility of this approach.

A new approach [80] considered the implementation of the look-up table throughout r only in intervals of Δ, where $r = -n\Delta - \delta$. Intervening values were obtained by interpolation using the Taylor series:

$$F(r) = F(-n\Delta) + D(-n\Delta)\frac{\delta}{1!} + D'(-n\Delta)\frac{\delta}{2!} + \ldots \tag{6.6}$$

The scheme exposed another problem of the LNS arithmetic implementation - the

difficulty of interpolating $F(r)$ in the region $-1 < r < 0$.

In our approach, the first order Taylor series approximation is applied. The look up tables are kept reasonably small by introducing an innovative error correction mechanism and a range shift algorithm. These are described in sequel, together with the conventional solution.

Conventional Implementation of LNS Addition

An interpolation detail of the function $F(r)$ is shown in Figure 6.6. To minimize the storage needs, Δ is progressively increased as the function becomes more linear with decreasing r. An intervening value of r lying in the $n-$th interval is thus correctly expressed as:

$$r = \begin{cases} -\delta & \text{for } n = 0, \\ \sum_{i=0}^{n-1}(-\Delta_i) - \delta & \text{for } n > 0. \end{cases} \tag{6.7}$$

For clarity of explanation, however, the following text will omit further reference to the variation in Δ and shorten the expression (6.7) to $r = -n\Delta - \delta$.

As mentioned above, the Taylor series approximation was chosen. For the first order case, the intervening value of r can be obtained by formula:

$$F(r) \approx F(-n\Delta) - \delta D(-n\Delta) \tag{6.8}$$

where, in the implementation of the LNS add/sub operation, both $F(-n\Delta)$ and $D(-n\Delta)$ are stored in tables.

This interpolation yields an error shown in the Figure 6.6. The error corresponding to an intervening value r can be obtained by the formula:

$$\epsilon(n, \delta) = F(-n\Delta) - \delta D(-n\Delta) - F(r) \tag{6.9}$$

For each n, ϵ increases with δ to a maximum

$$E(n) \approx F(-n\Delta) - \Delta_n D(-n\Delta) - F(r) \tag{6.10}$$

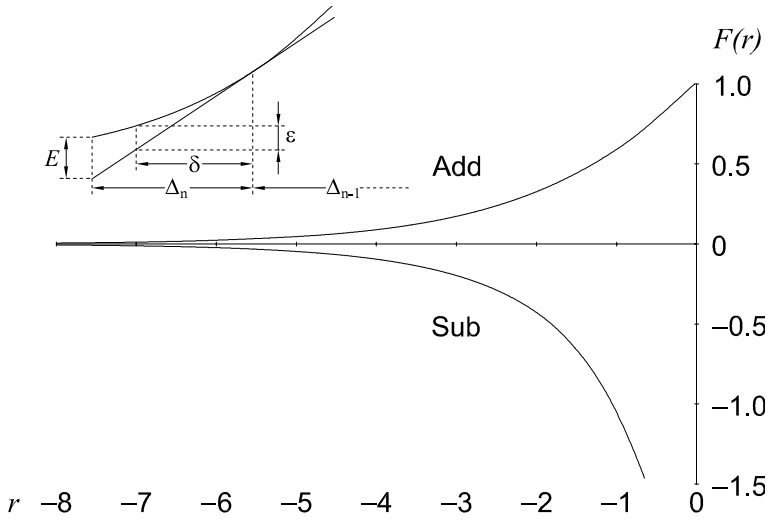


Figure 6.6: The Taylor approximation of $F(r)$ function

Error Correction Algorithm

As the word length grows up, the reasonable small look-up tables F and D produce too large an error in the interpolation. A correction mechanism was used to decrease the error with the given tables. We observe that, for a given δ, the ratio

$$P(n, \delta) = \frac{\epsilon(n, \delta)}{E(n)} \tag{6.11}$$

is roughly constant for all n. It is therefore possible to store, for a given n, a table P of the error at successive points throughout each Δ, expressed as a proportion of the maximum error E attained in that interval. It is also necessary to store, together with the tables F and D for each interval, its value of E. The error ϵ is then obtained for any (n, δ) as

$$\epsilon(n, \delta) \approx E(n)P(c, \delta)$$

where c is a constant. This is accumulated into the result of the interpolation, thereby correcting the error.

This scheme has a practical advantage that the lookups of $E(n)$ and $P(c, \delta)$ can be performed in the same time as those of $F(n)$ and $D(n)$ and that their product can

be evaluated simultaneously with the multiplication in the interpolation.

Range-Shift Algorithm

Since the $F(r)$ curve for subtraction goes quickly to $-\infty$ in the region $r \in (-0.5; 0)$, the approximation contains high error in this region. The range shift simplifies the subtraction operation by obviating the interpolation in this region. The shifter simply replaces the subtraction $2^j - 2^i$ with two successive subtractions:

$$2^i - 2^j = \left(2^i - 2^{j+k_1}\right) - 2^{j+k_2} \tag{6.12}$$

where

$$k2 = \log_2(1 - 2^{k_1})$$

Applying second-based logarithm to the equation (6.12), we obtain a new subtraction formula in the form

$$\log_2(2^j - 2^i) = i_2 + F(r_2) \tag{6.13}$$

where

$$i_2 = i + F(r + k_1) \tag{6.14}$$

$$r_2 = r + k_2 - F(r + k_1) \tag{6.15}$$

To evaluate (6.13) with terms i_2 and r_2, a simple method for $F(r + k_1)$ evaluation is needed. This is achieved by tabulating F in a table referred to as F_1 throughout the range R over which the range shift is required at intervals of Δ, and choosing k_1 such that the $F(r + k_1)$ is a tabulated point. In practice, the address for the table F_1 is obtained easily by the expression:

$$I_1 = r\,\langle \log_2(R) - 1 + f : \log_2(R) + f \rangle$$

where the $\langle \cdot \rangle$ denotes a bit extraction of its argument in the fixed point format (r is always positive) and f is used for correction according to the fraction length. The value of k_1 must be therefore chosen as

$$k_1 = r\,\langle \log_2(R) - 1 + f : 0 \rangle$$

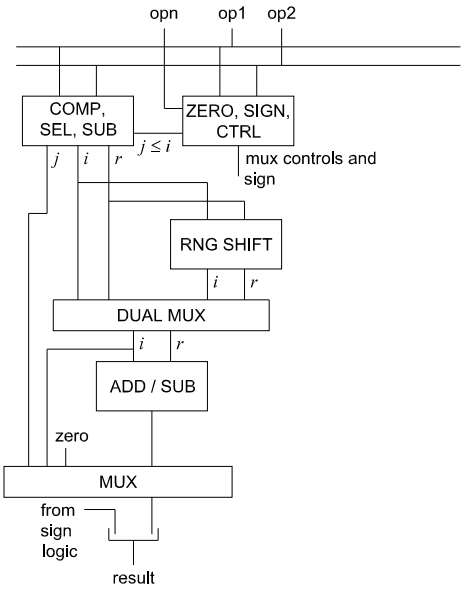


Figure 6.7: Schematic of the LNS adder/subtractor

The value k_2 can be evaluated by tabulating all values $\log_2(1-2^{k_1})$ in the range given by k_1, i.e., $r \in (-\Delta; 0)$. The address can be again obtained easily by :

$$I_2 = r \left\langle \log_2(\Delta) - 1 + f : 0 \right\rangle$$

The proposed method results in simple additional computation given by:

$$i_2 = i + F_1[I_1] \tag{6.16}$$

$$r_2 = r + F_2[I_2] - F_1[I_1] \tag{6.17}$$

where operator $[\cdot]$ denotes table look-up (such as reading from an array in C/C++)

Parameters of LNS arithmetic library for FPGA

The LNS core library implements all arithmetic operations in both 32-bit and 19-bit precision. All macros are optimized for the Xilinx Virtex devices. The structure of the adder/subtractor unit is shown in the Figure 6.7; it uses the theory described above. It is implemented as an 8-stage pipelined macro, so new calculation starts

	19-bit			32-bit			
Operation	Slices	% of chip	Frequency	Slices	% of chip	Frequency	Latency
add/sub twin	1616	8.4	60	2800	14.5%	50	8
mul/div	55	$> 1\%$	100	80	$> 1\%$	75	1
sqrt	16	$> 1\%$	100	28	$> 1\%$	110	1

Table 6.4: LNS core library parameters for Xilinx Virtex XCV2000E-6

	19-bit			32-bit			
Operation	Slices	% of chip	Frequency	Slices	% of chip	Frequency	Latency
add/sub twin	1368	8.4	112	2149	6.3%	100	8
mul/div	58	$> 1\%$	140	80	$> 1\%$	140	1
sqrt	16	$> 1\%$	140	27	$> 1\%$	150	1

Table 6.5: LNS core library parameters for Xilinx Virtex II XC2V6000-6

every clock cycle. For the table storage, used for the approximation, an internal dual-ported RAM was used. Since the tables are relatively big, we have implemented the add/sub unit as a twin, i.e., two add/sub units share the same tables.

Other macros are simple and they complete the operation in exactly one clock cycle. The Tables 6.4 and 6.5 show the important markers of the library, such as the core size, clock rate and pipeline latency, for the Xilinx Virtex and Virtex II devices. Note that add/sub unit consumes 96 block RAM cells in the 32-bit case and just 4 block RAM cells in the 19-bit case.

In the data range, the 32-bit LNS representation is equivalent to the single precision IEEE floating point format while the 19-bit LNS format is comparable to the 16-bit floating-point format used in some commercial DPS devices. More details about the library implemented in the frame of the HSLA project[1], can be found in [28], [20], [6], [29]. Finally, let us note that a library of bit-exact macros for Matlab simulations has been developed.

[1]Supported by the ESPRIT Long-Term Research program: project "High Speed Logarithmic Arithmetic Unit" grant no. 33544.

Chapter 7

Implementation results

In this chapter, concrete hardware implementations of the G-CMA and FICMA algorithms will be described. The aim of this chapter is to provide a proof-of-concept study showing that it is feasible to use the state-of-the-art devices to implement complex computations encountered in modern telecommunications.

Let us note that G-CMA was selected since it is the most simple algorithm, usually used as the reference design. The FICMA algorithm was used as an algorithm well suited for implementation using the above mentioned logarithmic arithmetic – as will be discussed in more detail later in the chapter.

As the target devices, the TMS320C6711 processor from Texas Instruments was used for the DSP implementation and the Xilinx Virtex series devices – either XCV2000E or XC2V6000 – were used for the hardware implementation. These devices offer both enough on-chip dual-port block RAMs, which are necessary to store the intermediate results, and sufficient amount of logic.

7.1 G-CMA implementation

The G-CMA algorithm was implemented on FPGA only. This algorithm is very similar to the LMS type of algorithms and it has very simple architecture. The basic elements of the algorithm are (see equation (3.5)):

- *Multiply-and-Accumulate* (MAC) processor calculating the output of the equalizer $y = \mathbf{f}^T \mathbf{r}_n$
- *error calculation* $e_n = (|y_n| - 1)^2 y_n$ and scalar-vector multiplication $\mu e_n \mathbf{r}_n$
- *vector subtraction* $\mathbf{f}(n+1) = \mathbf{f}(n) - \mu e_n \mathbf{r}_n$

7.1.1 Implementation of the LNS MAC unit

Since the LNS addition unit has the latency of 8 cycles, the main part to be optimized is the MAC unit, which is "addition intensive". For the optimization we suppose the size of vector $\mathbf{f}$ to be $N > 16$ i.e. at least twice the latency of LNS Add unit. For such N we have used an architecture depicted in Figure 7.1.

The input values of the MAC unit are the elements of vector $\mathbf{f}$ and $\mathbf{r}$. Every clock cycle, "new" vector element is fed to the unit inputs. These are multiplied by the LNS multiplier and the result is connected to the input of the LNS adder. In the first 8 cycles, the second input of the adder is connected to constant representing LNS zero value. After the first eight cycles, the output of the addition unit is connected to its second input. In that way, we get eight intermediate sums.

When the result of multiplication of last vectors element is send to the adder, the post-processing starts. In this phase, consecutive addition of the intermediate sums is performed. This overhead takes 32 cycles.

The MAC unit is controlled by the *enable* (en) and *ready* (rdy) signals. If *enable* is set to "1", valid data are on the inputs of the unit, when it is set to "0", the postprocessing phase starts (the signal is changed from "1" to "0" after the last valid data was sent to the unit). When the result of the MAC operation is available, the signal *rdy* is set to "1" for one clock cycle, otherwise it is set to "0".

The block RAMs with the coefficient vector $\mathbf{f}$ and with the input data vector $\mathbf{r}$ are not included in MAC unit, neither is the LNS adder. This way, the adder (which is the most costly element in size) can be reused by other macros.

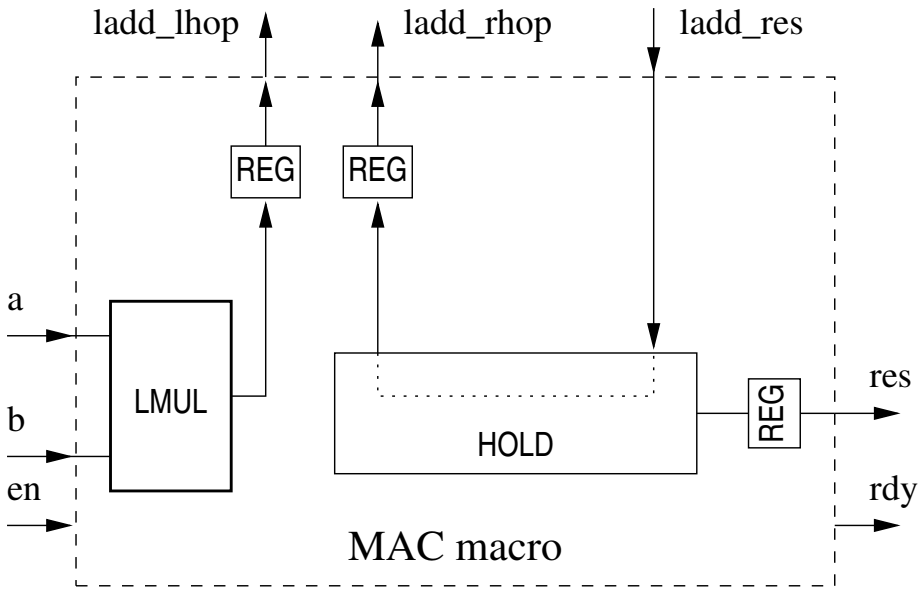


Figure 7.1: MAC unit block diagram

Other parts of the design (error calculation and vector subtraction) are very simple. For that reason, we do not describe them in more details. Note that in the vector subtraction procedure, the adder used in MAC unit can be reused. In our design, one LNS adder (half of twin) and two multipliers were used.

7.1.2 Implementation results

Since the G-CMA algorithm contains nearly as many additions as multiplications, we do not profit from the simplification of multiplication and division in the LNS implementation. On the other hand, LNS arithmetic profits from the lowest power consumption and easiest optimization of the design, since there is only one unit with long computational latency.

The FPGA design was implemented in Celoxica DK2 suit using the Handel-C language. The design was optimized using the Sinplify Pro tool from Synplicity, Inc. and compiled with Xilinx ISE 6.1.03. The results of the 32-bit and 19-bit LNS implementation are shown in Table 7.1

	19-bit precision		32-bit precision	
Block RAMs	16/160	10%	112/160	70%
Slices	1873/19200	9%	2547/19200	13%
TBufs	96/19520	1%	688/19520	3%

Table 7.1: The results of 19- and 32-bit G-CMA implementation for Virtex2000E

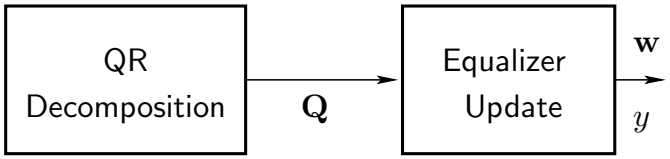


Figure 7.2: Block structure of the FI-CMA and SW-CMA implementation

7.2 FI-CMA and SW-CMA implementation

The FI-CMA algorithm was implemented using both FPGA and DSP. Since SW-CMA has structure very similar to FI-CMA – the only difference from implementation point of view is in the QR calculation – the architecture design presented for the later could be easily extended also for the sliding-window version of the algorithm. Both algorithms use the Givens rotations for the QR calculation but SW-CMA uses the update procedure presented in Chapter 4, while FI-CMA calculates QR decomposition from the input data matrix. In the design, this represents only a little change in the state machine, but the amount of utilized resources and units remains the same. To implement the algorithm, the following computational units are needed (see Figure 7.2):

1. The *QR-processor*, calculating set of Givens rotations

2. The *Equalizer update processor*, calculating one or more iterations of equalizer update (3.24) or (3.25)

Note that in contrast to the usual design of the QR-based RLS filter which uses the QR-update, we have to compute *explicitly* the first N columns of the $\mathbf{Q}$ matrix.

For the overall design of the implementation architecture, we have to keep in mind the following factors: matrix dimensions, restrictions on parallel access to the device

memories (dual ported/dual accessed RAMs) and limited number of the execution units.

For the matrix dimensions, let N be the equalizer order times oversampling factor and let M be the data block size. Then, $N \ll M$. Typical values for N are $10 \ldots 20$ for both algorithms. The size of M differs, for FI-CMA it is typically $M = 250 \ldots 1000$ where for SW-CMA it varies in the range $M = 2N \ldots N^2$. It follows that while $\mathbf{R}$ is a triangular $(N \times N)$ matrix, $\mathbf{Q}$ is an $(M \times N)$ matrix, i.e., it has many more rows than columns.

We assume that the data block is stored in some external FIFO memory. During each update step, only P most recent values are transfered (P is an oversampling factor). The input data vector $\mathbf{r}_n$ is stored in an internal circular buffer which is used to prepare the input for the QR processor.

The FPGA designs were optimized for the Xilinx Virtex and Virtex-II devices. The code was written and debugged in the Celoxica DK Suite using the Handel-C language and optimized by the Synplify Pro tool from Synplicity, Inc. The Xilinx ISE 6.1.03 was used for final compilation to bit-stream. In the FPGA implementation, the LNS arithmetic was used. The LNS implementation will profit from the low-cost multiplication, square and square-root operations, because of extensive use of the third and fourth power of vector elements and the fourth root and division of a scalar in the algorithm. Because of complexity of the design (the number of "processors" to be implemented), we have decided to use the 19-bit LNS arithmetic, to be able to use more adders while achieving acceptable precision.

In the next subsections we describe one DSP- and two different FPGA architectures of the algorithm implementation and its results. The first of the two design architecture for FPGA – the initial architecture – was developed "manually" while the second one was optimized using the methods of Integer Linear Programing.

7.2.1 Initial FPGA design

In this section, the architecture of the *QR* and *Equalizer update* processors is described. The number of the LNS adders was restricted to 4, but we did not put any limits on the number of LNS multipliers and square-root units. For the data storage, the internal dual-ported block RAMs were used, except for the equalizer parameters. Since the size of the equalizer parameter vector is small, it was stored in the distributed RAM.

QR-processor architecture For FI-CMA, the QR-decomposition is calculated using Givens rotations as follows. Let $r_{i,k}$ be the i^{th} element of the k^{th} row of matrix $\mathcal{R}$. Let $\mathbf{R}_k$ be a triangular matrix obtained by triangularization of the sub-matrix $\mathcal{R}_k$ (first k rows of matrix $\mathcal{R}$) and let $\mathbf{Q}_k$ be the corresponding orthonormal matrix. Then, matrices $\mathbf{Q}_{k+1}$ and $\mathbf{R}_{k+1}$ can be calculated as follows:

$$\begin{bmatrix} \mathbf{R_{k+1}} \\ \mathbf{0} \end{bmatrix} = G_N G_{N-1} \dots G_1 \begin{bmatrix} \mathbf{R}_k \\ \mathbf{r}_{k+1} \end{bmatrix} \tag{7.1}$$

$$\left[\begin{array}{c|c} \mathbf{Q}_{k+1} & \mathbf{q}_k \end{array} \right] = \left[\begin{array}{c|c} \mathbf{Q}_k & 0 \\ \hline 0 & 1 \end{array} \right] G_1^T \dots G_{N-1}^T G_N^T \tag{7.2}$$

where the matrix G_i eliminates $r_{i,k}$ and is defined as follows:

$$G_i = \begin{bmatrix} I & & & & \\ & c_i & & s_i & \\ & & I & & \\ & -s_i & & c_i & \\ & & & & I \end{bmatrix}$$

where I is the identity matrix. The sine and cosine parameters are computed using the following formulae:

$$c_i = \frac{\mathbf{R}_{(i,i)}}{\sqrt{\mathbf{R}_{(i,i)}^2 + r_{i,k}^2}} \qquad s_i = \frac{r_{i,k}}{\sqrt{\mathbf{R}_{(i,i)}^2 + r_{i,k}^2}} \tag{7.3}$$

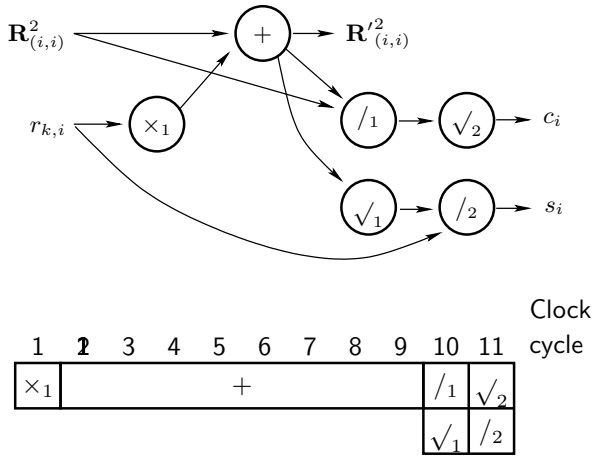


Figure 7.3: Data flow and timing of the diagonal cell

where $\mathbf{R}_{i,j}$ is the $(i,j)^{th}$ element of the triangular matrix $\mathbf{R}$.

Note that the SW-CMA use QR update procedures described in 2.6.1 with the same meaning of G_i, s_i and c_i.

From the equations (7.1) to (7.3) we can see that the QR processor consists of the following parts:

- the *diagonal processor* of the matrix $\mathbf{R}$, computing the rotation sine and cosine values;
- the *off-diagonal processor*, which rotates the rows of the matrix $\mathbf{R}$;
- the *column processor*, which rotates the columns of the matrix $\mathbf{Q}$.

The $\mathbf{R}$ matrix update procedure is implemented using one diagonal and one off-diagonal processor. While the off-diagonal cell depends on the output of the diagonal cell for the same input data, the rotation of the i-th row of the matrix $\mathbf{R}$ can be fully pipelined and can be computed in parallel with the update of $\mathbf{Q}$. Because of this data dependency, the diagonal and off-diagonal processor can share a single twin-adder. Data flows in both cells are shown in Figure 7.3 and 7.4.

Since the number of rows in $\mathbf{Q}$ increases with every new data vector in a single

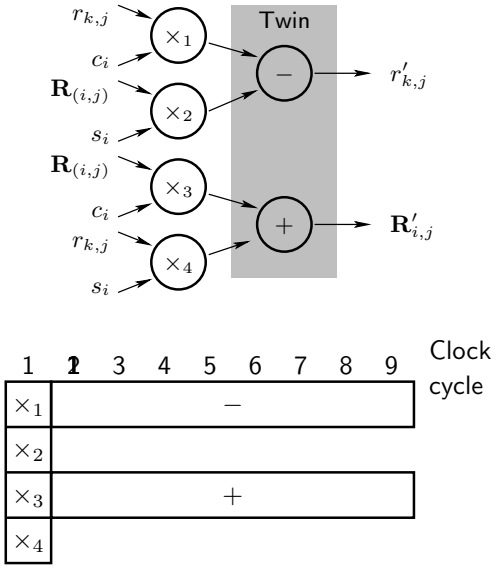


Figure 7.4: Data flow and timing of $\mathbf{R}$ off-diagonal and $\mathbf{Q}$ column cell

block, the computational complexity of this matrix grows linearly from the beginning to the end of the block. (Note the complexity of update of matrix $\mathbf{R}$ remains constant.) The intermediate results are stored in two dual-port RAMs: one for the elements of the matrix $\mathbf{Q}$, the other for the right-most column of the composed matrix in equation (7.2). The data flow in the column processor is the same as in the off-diagonal processor of $\mathbf{R}$ (Figure 7.4).

Equalizer update procedure The update of an equalizer coefficient is defined by the equation (3.24) or (3.25). In our implementation, we use the the simplified version with $\mu = \frac{1}{\mathbf{J}_i}$, with lower computational requirements. The *Equalizer update processor* consists of the following parts (see Figure 7.5):

- the *matrix-vector multiplier* for $\mathbf{y} = \mathbf{Q}\mathbf{w}$;
- the *parallel processor* for calculation of $\hat{\mathbf{w}} = \mathbf{Q}^{\mathbf{T}}\mathbf{y}^{\mathbf{3}}$ and $\mathbf{J}_i = \sum \mathbf{y}^4$;
- the *parameter update processor* for $\mathbf{v} = \mathbf{w} - \mu\hat{\mathbf{w}}/\mathbf{J}_i$ and $\mathbf{w} = \mathbf{v}/\left\|\mathbf{v}\right\|$.

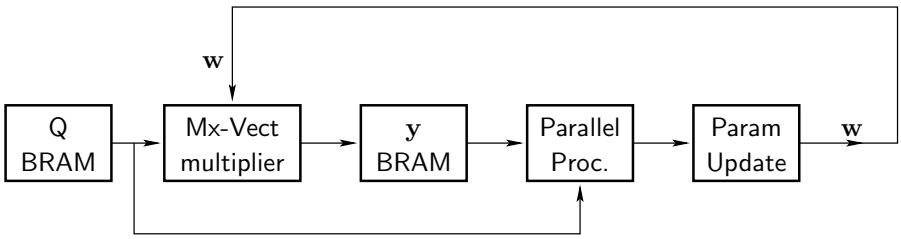


Figure 7.5: Equalizer update processor

The operations included in the equalizer update would allow for massively parallel computation in each step of update, however, the device limitations will again limit the degree of parallelism that can be achieved in practice.

The equalizer output $\mathbf{y}$ is computed using the vector-matrix multiplier. When the multiplication is evaluated in the usual way, i.e., row-wise with respect to matrix $\mathbf{Q}$, the efficiency of the resource utilization will be rather small. This is due to the relatively high latency of the LNS adder (and, consequently, the latency of the LNS Multiply-and-Accumulate unit) with respect to the row length. To improve the efficiency, we propose to compute the multiplication in a column-wise form: all elements of the i^{th} column of matrix $\mathbf{Q}$ will be multiplied by the i-th element of the coefficient vector $\mathbf{w}$. The result will be added to the previous results and stored in temporary memory. Using the dual-port Block RAMs, we are able to address two columns of $\mathbf{Q}$ at the same time. Since we do not accumulate (only multiply-and-add operation is used) and since the column size is much longer than the latency of the adder, there will be no dummy cycles. When partitioning the $\mathbf{Q}$ matrix to two Block-RAMs, for example to $\mathbf{Q}_e$ and $\mathbf{Q}_d$ with even and odd columns of $\mathbf{Q}$, we may employ up to four multiply-add units in parallel. Note that in the case of four multiply-add units, we can reuse all adders that were used for the QR-decomposition. The structure of the proposed unit is shown in Figure 7.6

The *parallel processor* computes the value of the criterion function which is simply $\sum y_i^4$ and the matrix-vector multiplication $\mathbf{Q}^T\mathbf{y}^3$. Since M is much higher than the latency of the MAC unit, it can be used efficiently here. With the $\mathbf{Q}$ matrix

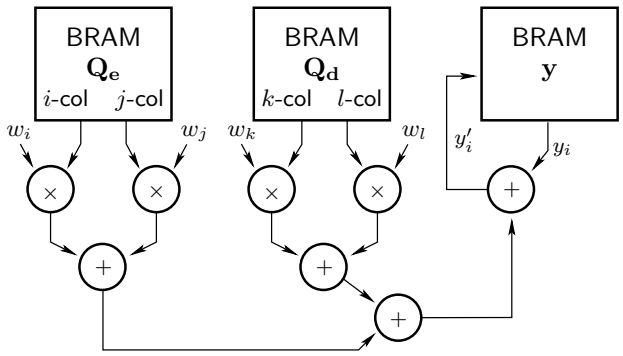


Figure 7.6: Matrix-vector multiplier

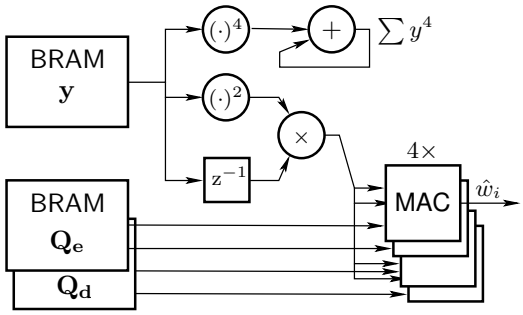


Figure 7.7: Parallel processor architecture

partitioned into two separate block RAMs, we may use up to four MAC units and reuse all the available hardware again. The structure of the processor is shown in Figure 7.7. It should be stressed again that with the LNS arithmetic, computation of the y^3 and y^4 powers is cheap on the amount of logic and can be calculated in a single cycle.

The structure of the *parameter update* processor is simple: it performs mainly computation of the step size $\mu_i = \frac{1}{\mathbf{J}_i}$, division and multiplication of a vector by constant and a second norm calculation. Again, with the LNS arithmetic, the computation of multiplication/division as well as the square- and fourth root are inexpensive. The architecture of the unit is straightforward.

7.2.2 Optimized FPGA design

The design was later optimized using the Basic Cyclic Scheduling algorithm (see [40] [19] [58] for the description). The aim of this algorithm is to find a periodic schedule while minimizing the schedule length.

Scheduling of the QR decomposition

QR decomposition algorithm, shown on Figure 7.8a, is a set of equations solved in an inner and an outer loop. The outer loop is repeated for each input data sample. The inner loop is a subject of our cyclic scheduling. Figure7.8b shows data flow graph of the M-th iteration of the outer loop, consisting of upper triangular matrix **R** cells redrawn in lower triangular form (i.e. flipped over main diagonal) and matrix **Q** cells. In i-th iteration of the inner loop (see one column in Figure 7.8b) computes one diagonal element of **R** (A cell), off diagonal elements of **R** (B cells) and one to k elements of **Q** (C cells). Therefore the number of lines of **Q** grows up to k, the iteration index of the outer loop.

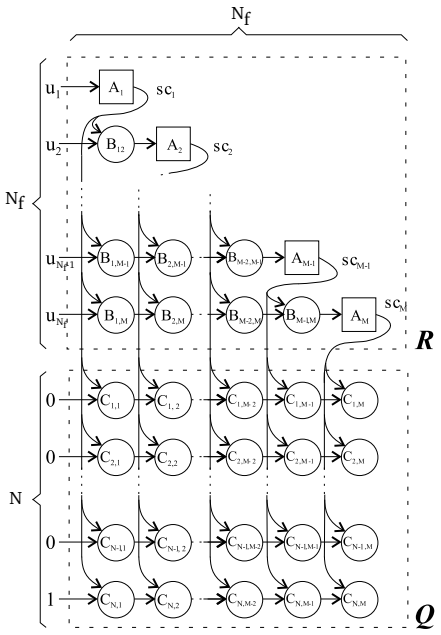
The most restrictive factor for parallelisation of the algorithm is presence of only one twin-adder (two LNS add units, which share the same tables). Since the algorithm contains a lot of parallelism, the aim is to find cyclic schedule where the additional unit is fully used. From the optimization point of view, the algorithm of the QR decomposition can be formulated as monoprocessor cyclic scheduling with precedence delays on single twin adder. The main problem of the algorithm is that the number of rows in **Q** increases, moreover the number of the off-diagonal operations in a row of matrix **R** is not constant since matrix **R** is triangular matrix. These facts cause that the number of operations in single iteration is not the same and that the cyclic scheduling cannot be directly used.

The proposed abstract model shown in Figure 7.9 of one iteration consists of:

- task T_A corresponding to one update of the diagonal cell of matrix **R** – cell A
- task T_{B1} corresponding to update of frist off-diagonal cell of matrix **R** – cell B;

```
for k = 1 to N do                    //outer loop
    u = data_{k:(k+N_f)};            //input data
    t = zeros(k, 1);
    t(k) = 1;
    for i=1 to N_f do                //inner loop
        [r_ii^2, sc] = A(r_ii^2, u_i);
        //computation of diagonal element of R
        for j = i + 1 to N_f do
            [r_ij, u_j] = B(r_ij, u_j, sc);
                //computation of off-diagonal element of R
        end
        for j = 1 to k do
            [q_ji, t_j] = C(q_ji, t_j, sc);
                //computation of elements of Q
        end
    end
end
```

a)



b)

Figure 7.8: a) The algorithm of QR decomposition. b) The data flow graph of the M-th iteration of the outer loop.

it takes data from one cell A and generates data for cell A in the next iteration

- task T_{BC} corresponding to the other cells B and cells C – update of off-diagonal cells of corresopnding row in **R** except the first one and corresponding column of the matrix **Q**

Task T_{BC} represents the cells B (excluding the first one) and the cells C sequentially processing iterations of the nested loop I and nested loop II. No expansion of the nested loop is needed at this point since the sequential processing fully utilizes the twin-adder, therefore it is optimal. The processing time of T_{BC} is given as a sum of the processing times of corresponding cells.

Figure 7.9 shows a graph $\mathcal{G}'$ corresponding to the abstract model assuming matrix **R** of size 20×20 and **Q** of size 24×20 (i.e. $M = 20$ and $N = 24$). The processing time of tasks T_A resp. T'_{B1} is equal to1 since they correspond to one cell. The processing

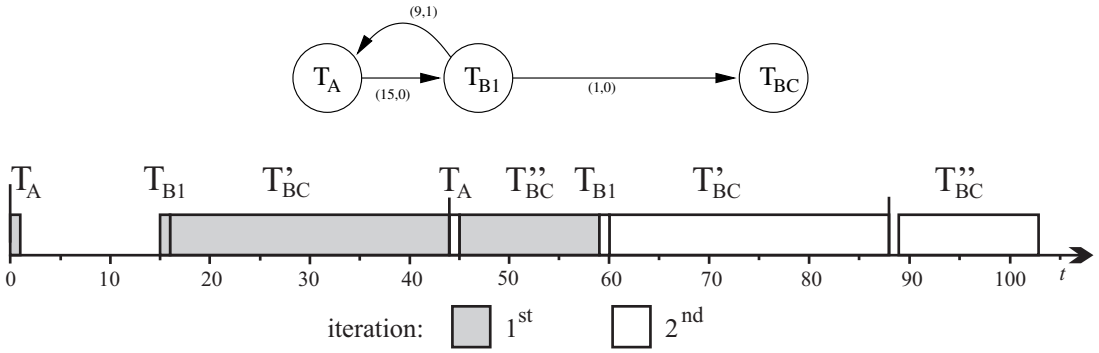


Figure 7.9: The abstract modelG' of QR decomposition (M-th iteration of the outer loop) and resultant schedule for first two iterations.

time p_{BC} of T_{BC} depends on the number of iterations of inner and outer loop therefore it si in the rage $\langle 0, M+N-2\rangle$.

It is visible that the optimal schedule of one iteration must contain a time gap of 14 clock cycles between tasks T_A and T_{B1} (since T_{B1} has to wait for result of T_A, processed in the pipelined unit). In order to be able to use the time gap the task T_{BC} is divided into tasks T'_{BC} and T''_{BC}, where processing time of T''_C equals to the length the time gap i.e. 14 cycles. The division of T_{BC} in the abstract model between T'_{BC} and T''_{BC} can be considered as one preemption of T_{BC}. Moreover the processing time of T''_{BC} need not to be fixed and can be included directly into ILP model.

Using this method, we have optimized the QR decomposition as a compact problem with of more efficient use of the LNS add/sub unit as the result. We have removed the dummy cycles between the diagonal and the off-diagonal rotation calculation, which were present in the initial design. Also the ineffectiveness in update of the rows of $\mathbf{R}$ has been removed.

Scheduling of the equalizer update iterative procedure

The equalizer update is performed by the iterative procedure (3.24) or (3.25). It's representation as a computational loop with data dependencies is in Figure 7.10.

for k=1 **to** N **do**
T_1: $\mathbf{y}(k) = \mathbf{Q} \cdot \mathbf{w}(k-1)$
T_2: $F(k) = \sqrt[4]{\sum \mathbf{y}(k)^4}$,
$F_\mu(k) = \frac{\mu}{F(k)}$
T_3: $\mathbf{v}'(k) = \mathbf{Q}^T \cdot \mathbf{y}(k)^3$
T_4: $\mathbf{v}(k) = \mathbf{w}(k-1) - F_\mu(k) \cdot \mathbf{v}'(k)$
T_5: $\alpha(k) = \frac{1}{\sqrt{\sum \mathbf{v}(k)^2}}$
T_6: $\mathbf{w}'(k) = \mathbf{v}(k) \cdot \alpha(k)$
end

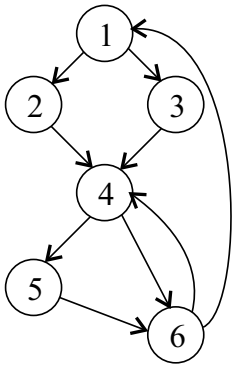


Figure 7.10: Equalizer update algorithm and its data dependencies.

From the graph of data dependencies, it is obvious that only operations T_2 and T_3 can be processed in parallel. Other operations are strictly sequential. Moreover, overlapping of iterations is not possible since the next iteration cannot start until the last operation in the previous iteration (operation T_6 in Figure 7.10) is finished.

The number of parallel computations in the process can be increased by the following modifications: Let us define $\mathbf{y}'$ as $\mathbf{y}' = \frac{\mathbf{y}}{\alpha}$. Using this substitution, operation T_1 in Figure 7.10 reads

$$\mathbf{y}' = \frac{\mathbf{y}}{\alpha} = \frac{\mathbf{Q} \cdot \mathbf{w}}{\alpha} = \mathbf{Q} \cdot \mathbf{v} \tag{7.4}$$

and the operations T_2 and T_3 are modified by a simple substitution of $\mathbf{y}$ by $\mathbf{y}'\alpha$. In result, we obtain modified set of equations and dependency graph $\mathcal{G}^c$ (see Figure 7.11)

In this model, processes T_1 and T_5 and processes T_2, T_3 and T_6 can be executed in parallel. The modification also uses overlapping between iterations, since the normalization of the new coefficient vector $\mathbf{w}$ is performed in the next iteration (processes T_5 and T_6). The schedule for utilization of the LNS adder, obtained by the BCS algorithm, is shown in Figure 7.12. In the graph, the timing is shown only for one adder (half of the twin). The notation used in the time slots, corresponding to the

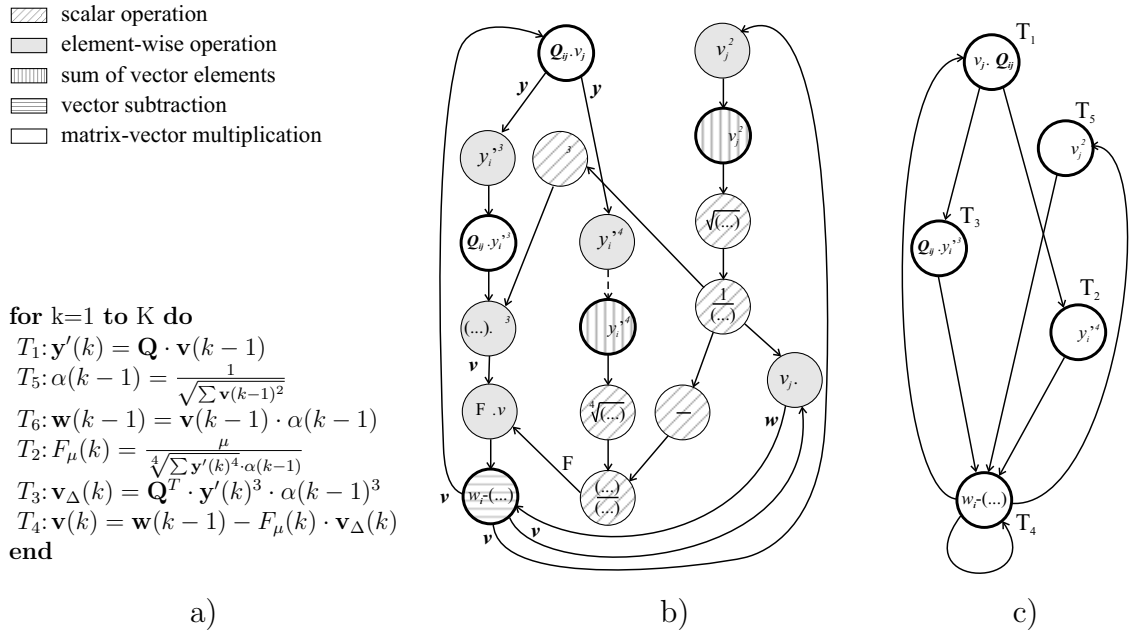


Figure 7.11: a) Iterative algorithm of equalizer. b) Data dependencies of the algorithm represented by condensed graph $\mathcal{G}^c$. c) Corresponding reduced condensed graph $\mathcal{G}'^c$

processes T_1 to T_5, has the following meaning:

- for process T_1, index i means that all rows of the i^{th} column of matrix $\mathbf{Q_o}$ or $\mathbf{Q_e}$ are treated
- for process T_3, index j means that all columns of the j^{th} row of matrix $\mathbf{Q_o}$ or $\mathbf{Q_e}$ are treated
- in processes T_2, T_4 and T_5, the number represents the element index of vector $\mathbf{y}$, $\mathbf{w}$ and $\mathbf{v}$ respectively.

When two LNS adders are used (one twin), the same process, delayed by 8 clock cycles, runs in parallel. In such a way, we obtain the modified architecture of the matrix-vector multiplier shown in Figure 7.13. In this figure, the index i represents the i^{th} column of the original matrix $\mathbf{Q}$, not of the submatrix $\mathbf{Q_o}$ or $\mathbf{Q_e}$. The vector $\mathbf{y}$ is used for the storage of the intermediate values as in the initial design.

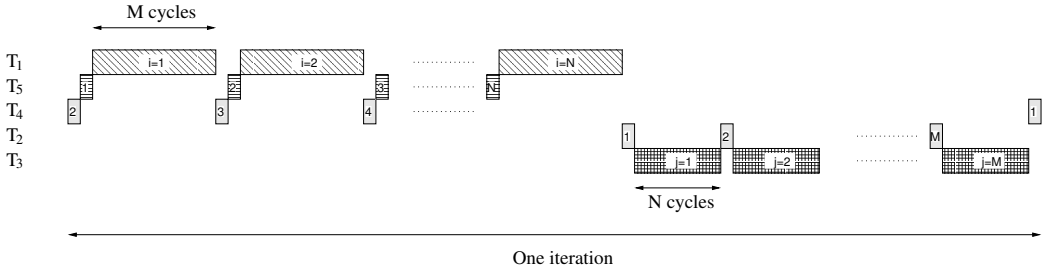


Figure 7.12: Time schedule of LNS adder utilization for Equalizer update processor

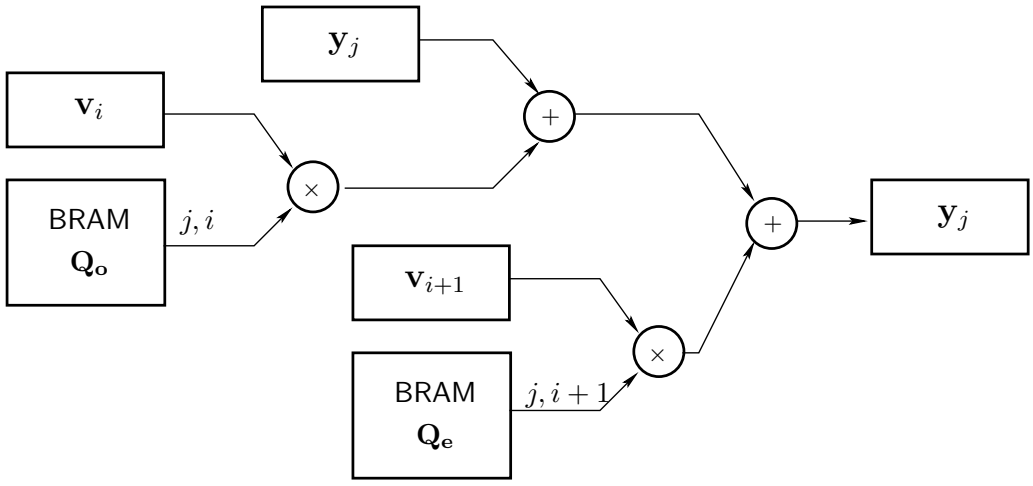


Figure 7.13: Modified architecture of matrix-vector multiply processor

	Virtex2000E		VirtexII-1000	
BlockRAMs	16/160	10%	16/40	40%
Slices	3972/19200	20%	3779/5120	73%
TBufs	192/19500	1%	192/2560	7%
MULT18×18	—	—	10/40	25%
Max Clock	35MHz	–	50MHz	–

Table 7.2: Implementation results of FI-CMA for Xilinx Virtex-E and Virtex-II devices

The *parallel processor* is now composed of the processes T_2 and T_3 and, contrary to the initial design, it uses the same calculation principle as the matrix-vector multiply process, i.e., all elements of the j^{th} row of matrix $\mathbf{Q}$ are multiplied by the third power of the j^{th} element of vector $\mathbf{y}$. Coefficient vector $\mathbf{v}$ is used for the intermediate values. Again, second process, delayed by 8 cycles, is run in parallel, as in the case of the matrix-vector multiply processor.

This rearrangement results in better utilization of the LNS adder, but it is effective only for a particular number of the LNS add/sub units and for a particular size of the matrix $\mathbf{Q}$. To be concrete, the efficiency is achieved whenever the parameter N satisfies the condition $N > p \times l$, where p is number of the Add units and l is their latency.

7.2.3 Hardware implementation results

The results of the implementation are shown in Table 7.2. The design was again implemented in Celoxica DK2 suit using the Handel-C language. The Sinplify Pro from Sinplicity, Inc. was used for the EDIF optimization and it was compiled by Xilinx ISE 6.1.03 tool. Because FI-CMA and SW-CMA use relatively huge amount of memory, only the 19-bit precision was implemented.

The algorithm implementation was tested on the data matrix of the size 20×24. The QR decomposition unit uses two LNS adders (i.e., one twin unit) and four multipliers, which results in 210 MFlops performance for Virtex-E and 300 MFlops for Virtex-II. The QR decomposition procedure execution time is 464μs and 325μs. One

iteration of equalizer update procedure takes 558 cycles which represents execution time 15.9μs for Virtex-E resp. 11μs for Virtex-II.

7.2.4 DSP implementation results

The FI-CMA algorithm has been implemented also in the TMS320C6711 DSP processor. The algorithm was written in ANSI C and optimized by the proprietary C compiler delivered with the Code Composer Studio environment. For the best performance, the following strategies were used:

- All the variables and data blocks were stored in the internal memory
- Both data and program memory space, use cache of equal size
- The storage of matrices in memory was optimized to simplify indexing
- When indexing a matrix, all indexes were calculated using the addition/subtraction operation, no multiplication was used
- Short functions were unrolled in the code
- Maximal optimization switch with respect to code speed was used
- The circular addressing mode was used for the circular buffer implementation

The code consists of the following functions: QR decomposition, equalizer update and matrix multiplication. Matrix multiplication is written as a separate function, since it is called from several places of the code. It is written in a form, which enables the use of parallel floating point operations. The results of the DSP implementation are shown in Table 7.3.

The cycles count corresponds to the data matrix size 92×10. Since the Matrix multiplication function is used several times during the computation, the minimum and maximum cycle count is shown. The execution takes 360,000 cycles in total. For the TMSC67xx device on 300 MHz, which is actually the fastest device of the C6000

Function	Code size	Cycles
matrix multiplication	496	4396 - 10009
QR decomposition	3048	9856
Equalizer update 1.iter.	1056	120 000

Table 7.3: DSP implementation results of FI-CMA

family, it represents execution time of 1.2 ms, which is not sufficient to deconvolve the GSM signal in real time. On the other hand, it should be noted that the analysis of the final code indicates that the code does not utilize well the parallel architecture of the device.

7.3 Conclusion

This chapter, with the design considerations and performance data for actual implementations, represents on of the core parts of the thesis. The implementation issues for the G-CMA (namely, an efficient implementation of the floating-point MAC unit) and for the FI-CMA/SW-CMA algorithms (implementation of the QR processor and of the equalizer initialization) using the LNS floating-point library are covered in detail. Also, a novel approach to optimization of the resource utilisation using the Basic Cyclic Scheduling algorithm is presented.

Considering the results presented in the chapter, it is possible to conclude that the use of the LNS arithmetic brings significant benefits for both implementing the QR-computations and for the FI-CMA-based equalizer initialization. Namely, since, in the LNS arithmetic, there is only one unit with long data latency, the code optimization problem is much simpler and it is possible to achieve nearly 100%. Also, it should be stressed that this latency is the same for both accuracies (19/32 bit) that were implemented so far (unlike many other floating point implementations, were the latency varies with the required accuracy).

Considering the Basic Cyclic Scheduling algorithm, it has been demonstrated that it can help to optimize the resource utilization in the FPGA design and that it can

be used for optimization of advanced DSP applications.

Finally, the performance data show that our FPGA implementation of the FI-CMA algorithm outperforms the implementation on a floating-point DSP. To be fair, it should be noted, that the for the DSP implementation, no scheduling optimization was used. Also, the DSP code does not fully use the parallel resources and it should be rewritten in some lower-level language (linear assembler, for example).

Chapter 8

Conclusion

This thesis deals with advanced digital signal processing algorithms for communications and the implementation issues associated with this topic. We have focused on the algorithms for blind equalization. Several different methods of blind equalization have been studied and their relations have been indicated. Finally, we have concentrated on the methods based on the constant modulus (CM) criterion.

Three CMA algorithms, probably the most cited ones, have been reviewed – Gradient CMA, Algebraic CMA and Finite Interval CMA – and their properties have been demonstrated in simulations. Based on the results of these simulations, we conclude that even though the G-CMA algorithm is indisputably very popular, namely thanks to its implementation simplicity, its performance is relatively low (when compared with the other two algorithms). While, in this algorithm, the step size parameter has significant influence to the algorithm, its optimal value is dependent on the channel parameters and its improper initialization can cause the algorithm to diverge.

The second algorithm – the Algebraic CMA – simultaneously deconvolves all CM signals contained in the received signal and, using the joint diagonalization procedure, it searches global optimum of the CM cost function. This method does not require any step size parameter and it is not sensitive to proper initilization. On the other hand, since it uses several SVD decompositions of large matrices, high data precision is required and the method is very computationally expensive. Probably due to the

sensitivity to precison of computations, it does not deconvolve all sources equally well with respect to the residual CM cost. Additionally, the difference between the residual CM costs of individual sources is influenced by the value of SNR.

Finaly, we have presented the properties of the Finite Interval CMA algorithm. It represents a compromis in computational requirements between the G-CMA and the ACMA algorithms. In the initial phase, it uses QR decompositon of the input data matrix; this is followed by iterative computatation of the equalizer. This computation is based on the gradient iteration method, but the orthornormal matrix $\mathbf{Q}$ is used instead, with result of very fast convergence speed of the algorithm (units of iterations). The value of the optimal step size (optimal from the point of view of the convergence speed) is known in analytical form. We have tested the algorithm for two step sizes; first results in fastest convergence, second gives lower residual CM cost and slightly slower convergence.

Finally, we have proposed a Sliding Window CMA as a modificitaion of the FI-CMA algorithm. It applies the sliding window on the input data and for every window, it calculates one or more iterations of the coefficient update as in FI-CMA. For the QR decomposition, it uses QR update instead of direct calculation of the decomposition from the input data matrix. Two methods of update were presented: row and column update. It was shown that the SW-CMA achieves the convergence properties of the FI-CMA and its computation requirements are similar. Moreover, it can operate with smaller sizes of data matrices.

Analysis of the error caused by the QR update procedure was presented. The error in QR decomposition and the orthonormality loss of $\mathbf{Q}$ matrix, was tested. The results shows, that both these errors have zero mean value and are acceptably small.

The behavior of the G-CMA, FI-CMA and SW-CMA for time varying channels was tested. Results for five different channels, representing some typical situations, have been presented. The results show, that even though the FI-CMA algorithm is a batch processing one, it is capable to deconvolve the channel with small time variations. The G-CMA, when properly initialized, can deconvolve the channel and

track the channel variations. But due to these variations, it can also diverge after it has already successfully converged. Finally, the SW-CMA performs well in nearly all presented cases. In contrast to the G-CMA, it has fast convergence rate and the algorithm can track successfully the time variations. Since the SW-CMA uses smaller data matrix (which corresponds to shorter time window), it is able to equalize the channels with relatively fast and significant changes.

In the last part of the thesis, we focus on the implementation architectures for digital signal processing. Implemetation on two different platforms – DSP and FPGA – are presented. The architecture of two modern devices has been reviewed: Texas Instruments TMS320C6711 DSP and the Xilinx Virtex FPGA. In this part, also the floating-point arithmetics for different data representation are presented, namely the Logarithmic Numbering System arithmetic has been explained in detail.

Finally, the implementaton results of the G-CMA and FI-CMA algorithms were presented. Both FPGA designs were implemented using the LNS core library. The design architecuter of the Multiply-and-Accumulate unit, which represents the most expensive part of G-CMA algorithms, is proposed. Finally, the results of the proposed FPGA design for 32- and 19-bit LNS arithmetic for Xilinx Virtex2000E are summarized. Since the G-CMA algorithm contains as many additions as multiplications and no division nor square root, the use of the LNS implementation does not bring much advantage.

In the FI-CMA design, we have first proposed the initial design of architecture, which was made "by hand". This design was then optimized using the Basic Cyclic Scheduling. The design was optimized with respect to the utilization of the LNS add unit, since it is the most expensive unit in the design, for particular size of the matrices $\mathbf{Q}$ and $\mathbf{w}$ and for particular number of the add units (one unit). The optimization resulted in full, 100%, utilization of the adder in both algorithm parts: the QR decomposition and coefficient update. It has to be pointed out that the optimized architecture is efficient only for the number of units the optimization was performed for and it sets limits on the matrix sizes. To be concrete, lowering the

matrix size of $\mathbf{Q}$ or using more units could result in less efficient design with dummy cycles.

In the case of the FI-CMA algorithm, the LNS arithmetic brings significant advantage, since there are much more multiplications than additions and there is a lot of power operations on vector elements in the algorithm. It has been shown that the LNS implementation on FPGA outperforms the modern floating point DSPs. Moreover, since the LNS arithmetic has only one unit with latency longer than 1, it is much easier to optimize the algorithm.

Appendix A

Formulation and solution of the scheduling problem

The iterative procedure, as that one mentioned in previous section, can be implemented as a computation loop performing an identical set of operations repeatedly. Therefore our work, dealing with optimized implementation of such procedures, is based on cyclic scheduling.

A.1 Basic Cyclic Scheduling

Operations in a computation loop can be considered as a set of n_T *generic tasks* $\mathcal{T} = \{T_1, T_2, ..., T_{n_T}\}$ to be performed K times where K is usually very large. One execution of $\mathcal{T}$ labeled with integer index $k \geq 1$ is called an *iteration*. Let us denote by T_i^k the k^{th} occurrence of the generic task T_i, which corresponds to the execution of statement i in iteration k. The scheduling problem is to find a start time $s_i(k)$ of every occurrence T_i^k [19]. Figure A.1 shows an example of a simple computation loop with corresponding processing times of oparations executed on HSLA.

Data dependencies of this problem can be modeled by a directed graph $\mathcal{G}$. Edge e_{ij} from the node i to j is weighted by a couple of integer constants l_{ij} and h_{ij}. Length l_{ij} is equal to p_i, the processing time of task T_i. In fact, l_{ij} represents minimal

for k=1 **to** K **do**
$Y(k) = (X(k-3)+1)^2 + 3$
$X(k) = Y(k) + 5$
$Z(k) = (Z(k-2)-2)^3 + 2$
end

operation of HSLA	proc. time p_i	input-output latency l_{ij}
$+, -$	1	9
$*, /, ^2, \sqrt{\ }$	1	2

Figure A.1: An example of a recurrent loop and corresponding processing times.

distance in clock cycles from a start time of task T_i to a start time of T_j and it is always greater than zero. On the other hand, the height h_{ij} specifies a shift of the iteration index related to the data produced by T_i and consumed by T_j. Therefore, each edge e_{ij} represents the set of K relation constraints of the type

$$s_i(k) + l_{ij} \leq s_j(k + h_{ij}), \quad \forall k \in \langle 1, K \rangle \,. \tag{A.1}$$

Figure A.2 shows the data dependence graph of a computation loop shown in Figure A.1.

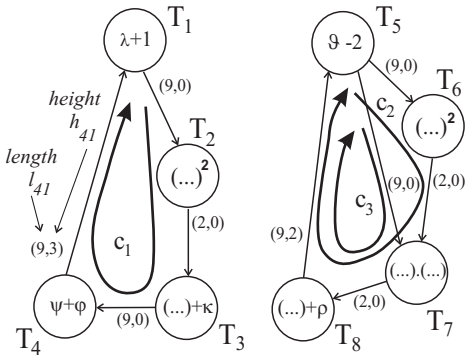


Figure A.2: Data dependency graph $\mathcal{G}$, representing operations and data of the computation loop in Figure A.1. Operation power three is realized as power two and multiplication. The $\mathcal{G}$ contains three cycles c_1, c_2 and c_3 with average cycles times $\{29/3, 22/2, 20/2\}$. The critical circuit is c_2 with $\tau = 11$.

The aim of *Basic Cyclic Scheduling* algorithm (BCS) [19] is to find a periodic schedule while minimizing the schedule length (C_{max}) by minimizing period length τ. The scheduling problem is simply solved when the number of processors is not limited, i.e. it is sufficiently large. Thereafter the minimal feasible period τ is given by the *critical circuit* c in graph $\mathcal{G}$. This is a circuit $c \in C(G)$ maximizing the ratio

$$\tau = \max_{c \in C(G)} \frac{\sum_{e_{ij} \in c} l_{ij}}{\sum_{e_{ij} \in c} h_{ij}}. \tag{A.2}$$

Any schedule with a shorter period can't be feasible assuming that the schedule of iterations is identical. The start time of task T_i^k is

$$s_i^k = s_i + \tau \cdot (k-1), \tag{A.3}$$

where s_i denotes the start time of task T_i^1. Using Equation (A.3), the set of K precedence constraints (A.1) can be reformulated as one inequality

$$s_j - s_i \geq l_{ij} - \tau \cdot h_{ij}. \tag{A.4}$$

Since the tasks are repeated every τ time units, the periodic schedule is entirely given by scalar τ and vector of start times in the first iteration $s = (s_1, s_2, ..., s_{n_T})$. An optimal periodic schedule can be provided in polynomial time, since the time complexity to find a critical circuits is $O(n_T^3 . \log(n_T))$.

A.2 Formulation of Monoprocessor Cyclic Scheduling with Precedence Delays

The basic cyclic scheduling problem, solved in polynomial time, assumes that the number of processors is not limited. When the number of processors is restricted, the problem becomes NP–hard [40]. The scheduling problem related to our motivation

example is even different, since some tasks run on one pipelined dedicated processor (the twin-adder of HSLA) and the remaining tasks run on arbitrary number of processors. Therefore, we introduce the model based on so called *precedence delays*.

In this new model, the length of edge e_{ij} is greater or equal to processing time p_i assigned to node T_i. Therefore, the processor is occupied by the task T_i during processing time p_i, but the task T_j may start at least l_{ij} time units after the start time of T_i. Therefore, related length l_{ij} specifies the *precedence delay* from task T_i to task T_j.

The precedence delays are useful when we consider pipelined processors. The processing time p_i represents the time to feed the processor and length l_{ij} represents the time of computation i.e. input-output latency. Therefore, the result of a computation is available after l_{ij} time units.

In the case of unlimited number of processors, the problem with precedence delays is still solvable using polynomial BCS. But assumption of unlimited number of processors is not satisfied for our application where some tasks are running on one dedicated processor (the twin-adder of HSLA). This problem can be formulated as *monoprocessor cyclic scheduling with precedence delays*.

Suggested formulation is based on the following reduction of graph $\mathcal{G}$ to $\mathcal{G}'$ while using calculation of the longest paths (solved e.g. by the Floyd's algorithm). All nodes (tasks) except the ones running on the dedicated processor are eliminated. Therefore, tasks running on the dedicated processor constitute nodes of $\mathcal{G}'$. There is e'_{ij} (the edge from T_i to T_j in $\mathcal{G}'$) of height h'_{ij} if and only if there is a path from T_i to T_j in $\mathcal{G}$ of height h'_{ij} such that this path goes only through eliminated nodes (task scheduled on an arbitrary number of processors). The value of length l'_{ij} is the longest path from T_i to T_j in $\mathcal{G}$ of height h'_{ij}.

Such reduction allows to find the schedule for our application by solving the problem of monoprocessor cyclic scheduling with precedence delays. The operations addition and subtraction from example on Figure A.2 are all processed on the dedicated processor.

A.3 Solution of Monoprocessor Cyclic Scheduling with Precedence Delays by ILP

Let $\hat{s}_i$ be the remainder after division of s_i by τ and let $\hat{q}_i$ be the integer part of this division. Then s_i can be expressed as follows

$$s_i = \hat{s}_i + \hat{q}_i \cdot \tau, \quad \hat{s}_i \in \langle 0, \tau - 1 \rangle, \quad \hat{q}_i \geq 0. \tag{A.5}$$

This notation divides s_i into $\hat{q}_i$, the index of *execution period*, and $\hat{s}_i$, the number of clock cycles within the execution period. The schedule has to obey to two kinds of restrictions. The first kind of restrictions is given by *precedence constraint* corresponding to Inequality (A.4). It can be formulated using $\hat{s}$ and $\hat{q}$ as

$$(\hat{s}_j + \hat{q}_j \cdot \tau) - (\hat{s}_i + \hat{q}_i \cdot \tau) \geq l'_{ij} - \tau \cdot h'_{ij}. \tag{A.6}$$

Each edge represents one precedence constraint. Hence, we have n'_e inequalities (n'_e is the number of edges in reduced graph $\mathcal{G}'$).

The second kind of restrictions are *processor constraints* [58]. They are related to the monoprocessor restriction, i.e. at maximum one task is executed at a given time.

$$p_j \leq \hat{s}_i - \hat{s}_j + \tau \cdot \hat{x}_{ij} \leq \tau - p_i. \tag{A.7}$$

To derive feasible monoprocessor schedule, Double–Inequality (A.7) must hold for each unordered couple of two distinct tasks. Therefore, there are $(n'^2_T - n'_T)/2$ Double–Inequalities (where n'_T is the number of tasks in reduced graph $\mathcal{G}'$), i.e. there are ${n'_T}^2 - n'_T$ inequalities specifying the processor constraints.

Using ILP formulation we are able to test the schedule feasibility for given τ. In addition we can minimize the iteration overlap by formulation the objective function as $min \sum_{i=1}^{n_T} \hat{q}_i$.

The summarized ILP model, using variables $\hat{s}_i, \hat{q}_i, \hat{x}_{ij}$, is shown in Figure A.3. It contains $2n'_T + (n'^2_T - n'_T)/2$ variables and $n'_e + n'^2_T - n'_T$ constraints.

$$\min \sum_{i=1}^{n_T} \hat{q}_i$$

subject to

$$\hat{s}_j + \hat{q}_j \cdot \tau - \hat{s}_i - \hat{q}_i \cdot \tau \geq l'_{ij} - \tau.h'_{ij} \ , \quad \forall e'_{ij} \in G'$$

$$p_j \leq \hat{s}_i - \hat{s}_j + \tau \cdot \hat{x}_{ij} \leq \tau - p_i \ , \quad \forall i \neq j \text{ and } T_i, T_j \in \mathcal{T}$$

where

$\hat{s}_i \in \langle 0, \tau - 1 \rangle$, $\hat{q}_i \geq 0$, $\hat{x}_i \in \langle 0, 1 \rangle$,
$\hat{s}_i, \hat{q}_i, \hat{x}_{ij}$ are integers.

Figure A.3: ILP model.

Period τ^*, the shortest period resulting in feasible schedule, can be found iteratively by formulating one ILP model for each iteration. These ILP iterations need not to be performed for all τ between the lower and upper bound, but the interval bisection method can be used, since τ^* is not preceded by any feasible solution (i.e. no $\tau \leq \tau^* - 1$ results in a feasible solution). Therefore, there are at maximum $\log_2 (upperbound - lowerbound)$ iterative calls of ILP.

A.4 Generalization to the Set of Distinct Dedicated Processors

The above mentioned formulation of processor constraints allows generalization to the problem with more than one dedicated processor. The role of Double–Inequality (A.7) is to avoid conflicts of tasks T_i and T_j when the same processor is dedicated to both tasks (e.g. both T_i and T_j perform addition on an exclusive addition unit). Assuming a problem with the set of distinct dedicated processors, the tasks dedicated to different processors are not conflicting, i.e. they do not impose any restriction with

respect to the feasibility of the schedule. Therefore Double–Inequality (A.7) is only considered for each couple of tasks dedicated to the same dedicated processor.

A.5 Elimination of Redundant Processor Constraints

The time requirements to solve the generated ILP model roughly corresponds to the number of integer variables, therefore it is meaningful to reduce this number as much as possible. Not all processor constraints are necessary considering precedence constraints (A.6) and keeping in mind that the tasks from distinct iterations can be found in one period related to $\hat{q}_{max} = max\{\hat{q}_i\}$ given *a priory* as an additional constraint to the ILP model A.3. The necessity of Double–Inequality (A.7) for the couple of tasks T_i and T_j dedicated to the same processor could be decided e.g. while using linear programming (LP) composed of constraints (A.8), (A.9) and (A.10). If any feasible solution of this LP problem exists, then T_i and T_j can be in conflict.

Inequality (A.8) represents all precedence constraints given by graph G'. Inequality (A.9) represents the case when the end of task T_j overlaps with the start of the task T_i by at least one clock cycle and inequality (A.10) holds when the end of task T_i overlaps with the start of the task T_j by at least one clock cycle. This is considered iteratively for all possible δ, the difference of the iteration index between tasks T_i and T_j within one period.

If the LP finds a feasible solution for any integer $\delta \in \langle -\hat{q}_{max}, \hat{q}_{max} \rangle$, the tasks T_i and T_j can eventually cause conflict on their dedicated processor and then the corresponding Double–Inequality (A.7) is necessary.

$$s_k - s_l \geq l'_{kl} - \tau \cdot h'_{kl}, \qquad \forall e'_{kl} \in G', \tag{A.8}$$

$$s_i - s_j + \delta \cdot \tau \leq p_j - 1, \tag{A.9}$$

$$s_j - s_i - \delta \cdot \tau \leq p_i - 1, \tag{A.10}$$

where

$$s_i \in \langle 0, (\hat{q}_{max} + 1) \cdot \tau \rangle \tag{A.11}$$

This elimination leads to the decrease in the number of constraints and to the decrease in the number of binary decision variables $\hat{x}_{ij}$.

Bibliography

[1] A. Benveniste and M. Goursat. Blind equalizers. *IEEE Trans. on Communication*, 32:871–883, August 1984.

[2] Celoxica Ltd. *Handel-C Language Reference Manual*, 2004.

[3] W. Chung and J. P. LeBlanc. The Local Minima of Fractionally-Spaced CMA Blind Equalizer Cost Function in the Presence of Channel Noise. In *IEEE Int. Conf on Acoustics, Speech and Signal Processing*, May 1998.

[4] Chung. W. *Geometrical Understanding of the Constant Modulus Algorithm: Adaptive Blind Equalization and Cross-Polarized Source Separation*. PhD thesis, Cornell University, May 1999.

[5] J. N. Coleman, E. I. Chester, C. I. Softley, and J. Kadlec. Arithmetic on the European Logarithmic Microprocessor. *IEEE Transactions on Computers*, 49(7):702–715, 2000.

[6] Coleman J. N. and Kadlec J. Extended Precision Logarithmic Arithmetic. In Monterey IEEE Signal Processing Society, editor, *Signal Systems and Computers 2000, 34th Asilomar Conference on Signal Systems and Computers*, pages 124–129, 2001.

[7] P. Comon. Independent Component Analysis - A new concept? *Signal Processing*, 36:287–314, 1994.

[8] A. de Baynast and I. Fijalkow. Prefiltered blind equalization: how to fully benefit from spatio-temporal diversity? In *Proceedings of Acoustics, Speech, and Signal Processing*, volume 4, pages 2069 – 2072, May 2001.

[9] L. De Lathauwer. *Signal Processing based on Multilinear Algebra*. PhD thesis, KU Leuven, Belgium, 1997.

[10] D. Donoho. On Minimum Entropy Deconvolution. In D.F. Findley, editor, *Applied time series analysis II*, pages 565–608. New York: Academic, 1981.

[11] C. C. Feng and C. Y. Chi. Performance of Cumulant Based Inverse Filters for Blind Deconvoluion. *IEEE Trans. on Signal Processing*, 47(7):1922–1935, 1999.

[12] C. C. Feng and C. Y. Chi. Performance of Shalvi and Weinstein's Deconvolution Criteria for Channels with/without Zeros on the Unit Circle. *IEEE Trans. on Signal Processing*, 48(2):571–575, 2000.

[13] I. Fijalkow, C. E. Manlove, and C. R. Johnson Jr. Adaptive Fractionally Spaced Blind CMA Equalization: Excess MSE. *IEEE Trans. on Signal Processing*, 46(1):227–231, 1998.

[14] I. Fijalkow, A. Touzni, and J. R. Treichler. Fractionaly Spaced Equalization Using CMA: Robustness to Channel Noise and Lack of Disparity. *IEEE Trans. on Signal Processing*, 45(1):56–66, January 1997.

[15] I. Fijalkow, J. R. Treichler, and C. R. Johnson, Jr. Fractionally Spaced Blind Equalization: Loss of Channel Disparity. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ASSP-95)*, pages 1988–1991, Detroit, MI, May 1995.

[16] D. N. Godard. Self-Recovering Equalization and Carrier Tracking in Two-Dimensional Data Communication Systems. *IEEE Trans. Communications*, 28:1867–1875, November 1980.

[17] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[18] W.C. Gray. *Variable Norm Deconvolution*. PhD thesis, Stanford University, 1979.

[19] C. Hanen and A. Munier. A Study of the Cyclic Scheduling Problem on Parallel Processors. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 57, 1995.

[20] Hermanek A., Matousek R., Licko M., and Kadlec J. FPGA Implementation of Logarithmic Unit. In *Sbornik prispevku 8. rocniku konference MATLAB 2000*, 2000.

[21] A. Hyvärinen. A Family of Fixed-Point Algorithms for Independent Component Analysis. In *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing (ICASSP'97)*, pages 3917–3920, Munich, Germany, 1997.

[22] A. Hyvärinen. Fast and Robust Fixed Point Algorithms for Independent Component Analysis. *IEEE Trans. on neural networks*, 10(3):626–634, May 1999.

[23] Institute of Electrical and Electronics Engineers, New York. *IEEE Standard for Binary Floating Point Arithmitic, ANSI/IEEE Standard 754-1987*, 1987.

[24] C. R. et al Johnsoh Jr. The Core of FSE-CMA Behaivor Theory. In Haykin S., editor, *Blind deconvolution II*. Wiley, New York, 2000.

[25] R.C. Johnson et al. Blind Equalization Using the Constant Modulus Criterion: A review. *Proc. IEEE*, 86:1927–1950, October 1998.

[26] R. C. Johnson Jr. and B. D. O. Anderson. Godard Blind Equalizer Error Surface Characteristics: White, Zero-mean, Binary Case. *Int. Journal of Adaptive Control and Signal Processing*, 9:301–324, July 1995.

[27] C. Jutten and J. Herault. Blind Separation of Sources, Part I: An Adaptie Algorithm Based on Neuromimetic Architecture. *Signal Processing*, 24:1–10, 1991.

[28] Kadlec J., Matousek R., Hermanek A., Licko M., and Softley C. Logarithmic ALU 32-bit for Handel C 2.1 and Celoxica DK1. In Abington Celoxica, editor, *Celoxica User Conference. Proceedings*, 2001.

[29] Kadlec J., Matousek R., and Licko M. FPGA Implementation of Logarithmic Unit Core. In Nürnberg Design & Elektronik, editor, *Embedded Intelligence 2001*, pages 547–554, 2001.

[30] G. Kawas Kaleh and R. Vallet. Joint Parameter Estimation and Symbol Detection for Linear or Nonlinear Unknown Channels. *IEEE Trans. on Communication*, 42:2406 – 2413, July 1994.

[31] J. P. LeBlanc and P. L. DeLeon. Speech Separation by Kurtosis Maximization. In *IEEE Int. Conf on Acoustics, Speech and Signal Processing*, Seattle, WA, May 1998.

[32] J. P. LeBlanc, I. Fijalkow, and C.R. Johnson Jr. CMA Fractionally Spaced Equalizers: Stationary Points and Stability Under i.i.d. and Temporally Correlated Sources. *International Journal of Adaptive control and Signal Processing, special issue on adaptive channel equalization*, 12(2):135–156, March 1998.

[33] J. P. LeBlanc and Fijalkow I. Blind Adapted, Pre-Whitened Constant Modulus Algorithm. In *International conference on Communications, ICC 2001*, volume 8, pages 2438 – 2442, June 2001.

[34] J. P. LeBlanc and Jr. C. R. Johnson. Global CMA Error Surface Characteristics, Source Statistic Effects: Polytopes and Manifolds. In *13th Int'l. Conf. on Digital Signal Processing*, Santorini, Greece, July 1997.

[35] A. Leshem, N. Petrochilos, and A.J. van der Veen. Finite Sample Identifiability of Multiple Constant Modulus Sources. *IEEE Tr. Information Theory*, 49(9):2314–2319, September 2003.

[36] Y. Li and Z. Ding. Convergence Analysis of Finite Length Blind Adaptive Equalizers. *IEEE Trans. on Signal Processing*, 43(9):2120–2129, Sep. 1995.

[37] Y. Li and Z. Ding. Global Convergence of Fractionally Spaced Godard (CMA) Aadaptive Equalizers. *IEEE Trans. on Signal Processing*, 44(4):818–826, November 1996.

[38] R. W. Lucky. Techniques for Adaptive Equalization of Digital Communication Systems. *Bell Systems Technical Journal*, 45(2):255–286, February 1966.

[39] E. Moulines, P. Duhamel, J. Cardoso, and S. Mayrargue. Subspace Methods for Blind Identification of Multichannel FIR Filters. *IEEE Trans. on Signal Processing*, 43(2):516–525, February 1995.

[40] A. Munier. The Complexity of a Cyclic Scheduling Problem with Identical Machines and Precedence Constraints. *European Journal of Operational Research*, pages 471–480, 1996.

[41] John G. Proakis. *Digital Communication*. McGraw Hill, 1995.

[42] P. A. Regalia. On the Equivalence Between the Godard and Shalvi-Weinstein Schemes of Blind Equalization. *Signal Processing*, 73(1-2):185–190, Feb. 1999.

[43] P. A. Regalia. A Finite-Interval Constant Modulus Algorithm. In *Proc. International Conference on Acoustics Speech, and Signal Processing (ICASSP-2002)*, volume III, pages 2285–2288, Orlando, FL, May 13-17 2002.

[44] P. A. Regalia and Mamadou Mboup. On the Equivalence Between the Super-Exponential Algorithm and a Gradient Search Method. In *Proc. International Conference Acoustics, Speech, and Signal Processing (ICASSP'99)*, volume 5, pages 2643–2646, March 1999.

[45] P. A. Regalia and Mamadou Mboup. Undermodeled Equalization: A Characterization of Stationary Points for a Family of Blind Criteria. *IEEE Trans. on signal proc.*, 47(3):760–770, March 1999.

[46] P. A. Regalia and Mamadou Mboup. Properties Of Some Blind Equalization Criteria in Noisy Multiuser Environments. *IEEE Trans. on signal proc.*, 49(12):3112–3122, Dec. 2001.

[47] P.A. Regalia and E. Kofidis. Monotonic Convergence of Fixed-Point Algorithms for ICA. *IEEE Trans. Neural Networks*, 14(4):943–949, 07 2003.

[48] Y. Sato. A Method for Self-recovering Equalization for Multilevel Amplitude-Modulation Systems. *IEEE Trans. on Communications*, 23(6):679–682, 06 1975.

[49] P. Schniter, R. A. Casas, A. Touzni, and Jr. C.R. Johnson. Performance Analysis of Godard-Based Channel Identification. *IEEE Trans. on Signal Processing*, 49(8):1757–1767, August 2001.

[50] P. Schniter and C. R. Johnson, Jr. The Dithered Signed-Error Constant Modulus Algorithm. *IEEE Trans. on Signal Processing*, 47(6):1592–1603, June 1999.

[51] P. Schniter and Jr. C. R. Johnson. Dithered Signed-Error CMA: The Complex-Valued Case. In *32nd Asilomar Conference on Signals, Systems, and Computers*, November 1998.

[52] P. Schniter and Jr. R. C Johnson. Bounds for the MSE performance of constant modulus estimators. *IEEE Transactions on Information Theory*, 46(7):2544–2560, July 2000.

[53] P. Schniter and C. R. Johnson Jr. Sufficient Conditions for the Local Convergence of Constant Modulus Algorithms. *IEEE Trans. on Signal Processing*, 48(10):2785–2796, October 2000.

[54] P. Schniter and L. Tong. Existence and performance of Shalvi-Weinstein estimators. *IEEE Trans. on Signal Processing*, 49(9):2031–2041, 2001.

[55] N. Seshadri. Joing data and channel estimation using blind trellis search techniques. *IEEE Trans. on Communication*, 42:1000 – 1011, April 1994.

[56] O. Shalvi and E. Weinstein. New criteria for blind deconvolution of nonminimum phase systems. *IEEE Transaction of Informaition Theory*, 36:312–321, Mar 1990.

[57] O. Shalvi and E. Weinstein. Super-exponential methods for blind deconfolution. *IEEE Trans. of Informaition Theory*, 39:504–519, 05 1993.

[58] P. Sucha, Z. Pohl, and Z. Hanzálek. Scheduling of iterative algorithms on FPGA with pipelined arithmetic unit. In *10th IEEE real-time and embedded technology and applications symposium*, 2004.

[59] E. E. Swartzlander and A. G. Alexopoulos. The Sign / Logarithm Number System. In *IEEE Transactions on Computers*, volume C-24, pages 1238–1242, 1975.

[60] Texas Instruments. *TMS320C6000 Assembly Language Tools User's Guide*, March 2004.

[61] Texas Instruments. *TMS320C6000 CPU and Instruction Set Ref Guide*, June 2004.

[62] Texas Instruments. *TMS320C6000 DSP Peripherals Overview Reference Guide*, October 2004.

[63] Texas Instruments. *TMS320C6000 Optimizing Compiler User's Guide*, March 2004.

[64] L. Tong, G. Xu, B. Hassibi, and T. Kailath. Blind Channel Identification Based on Second-Order Statistic: A Frequenci Domain Approach. *IEEE Trans. on Information Theory*, 41:329–334, January 1995.

[65] L. Tong, G. Xu, and T. Kailath. Fast Blind Equalization Via Antenna Arrays. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 272–275, Mineapolis, MN, April 1993.

[66] L. Tong, G. Xu, and T. Kailath. Blind Channel Identification Based on Second-Order Statistics: A Time Domain Approach. *IEEE Trans. on Information Theory*, 40(2):340–349, March 1994.

[67] A. Touzni and I. Fijalkow. Does Fractionally-Spaced CMA Converge Faster Than LMS? In *Proc. European Signal and Image Processing Conference*, pages 1227–1230, Trieste, Italy, September 1996.

[68] A. Touzni, I. Fijalkow, M. G. Larimore, and J. R. Treichler. Blind Adaptive Deconvolution from Multiple Mixtures. *IEEE Trans. on Signal Processing*, 49(6):1166–1178, June 2001.

[69] A. Touzni, I. Fijalkow, and J. R. Treichler. Robustness of Fractionally-Spaced Equalization by CMA to Lack of Channel Disparity and Noise. In *Proc. signal Processing Workshop on Statistical Signal and Array Processing*, pages 144–147, Corfu, Greece, June 1996.

[70] J. R. Treichler, I. Fijalkow, and C. R. Johnson, Jr. Fractionally-Spaced Equalizers: How Logn Should They Really Be? *Signal Processing Magazine*, 13(3):65–81, May 1996.

[71] J.R. Treichler and B.G. Agee. A new approach to multipath correction of constant modulus signals. *IEEE Trans. Acoustics, Speech, Signal Processing*, 31:459–471, April 1983.

[72] A.-J. van der Veen. Analytical method for blind binary signal separation. *IEEE Trans. Signal Processing*, 45(4):1078–1082, April 1997.

[73] A.-J. van der Veen. Blind separation of BPSK sources with residual carriers. *Signal Processing*, 76(1):67–79, January 1999.

[74] A.-J. van der Veen. Asymptotic Properties of the Algebraic Constant Modulus Algorithm. *IEEE Trans. Signal Processing*, 49(8):1796–1807, August 2001.

[75] A.-J. van der Veen. Statistical performance analysis of the Algebraic Constant Modulus Algorithm. *IEEE Tr. Signal Processing*, 50(12):3083–3097, December 2003.

[76] A.-J. van der Veen and A. Paulraj. An analytic constant modulus algorithm. *IEEE Trans. Signal Processing*, 44:1136–1155, May 1996.

[77] R.A. Wiggins. Minimum entropy deconvolution. In *Geoexploration*, pages 21–35, 1978.

[78] C. Wonzoo and Jr. R. C. Johnson. Characterization of the Regions of Convergence of CMA Adapted Blind Fractionally Spaced Equalizer. In *32nd Asilomar Conference on Signals, Systems, and Computers*, November 1998.

[79] Xilinx. *Virtex Data Book*, 2003.

[80] L. K. Yu and D. M. Lewis. A 30-b Integrated Logarithmic Number System Processor. In *IEEE Journal on Solid-State Circuits*, volume 26, pages 1433–1440, 1991.

[81] H.H. Zeng, L. Tong, and C. R. Johnson Jr. Relationships between the constant modulus and Wiener receivers. *IEEE Trans. Information Theory*, 44(4):1523–1538, July 1998.

List of publications

Papers in proceedings

[A1] Heřmánek A., Matoušek R., Líčko M., Kadlec J.: FPGA implementation of logarithmic unit. In: *Sborník příspěvků 8. ročníku konference MATLAB 2000.* VŠCHT, Praha 2000, pp. 84-90.

[A2] Heřmánek A., Matoušek R., Líčko: Alpha accelerator for RTW - Windows Target. In: *Sborník příspěvků 8. ročníku konference MATLAB 2000.* VŠCHT, Praha 2000, pp. 197-201.

[A3] Albu F., Kadlec J., Softley C., Heřmánek A., Matoušek R., Coleman J. N., Fagan A.: Implementation of (Normalised) RLS Lattice on Virtex. In: *Field-Programmable Logic and Applications. Proceedings.* (Brebner G., Woods R. eds.). (Lecture Notes in Computer Science. 2147). Springer, Berlin 2001, pp. 91-100.

[A4] Heřmánek A., Matoušek R., Líčko M., Kadlec J., Pohl Z.: Pipelined logarithmic 32bit ALU for Celoxica DK1. In: *Sborník příspěvků 9. ročníku konference MATLAB 2001.* (Procházka A., Uhlíř J. eds.). VŠCHT, Praha 2001, pp. 72-80.

[A5] Kadlec J., Heřmánek A., Matoušek R., Líčko M., Softley C.: Logarithmic ALU 32-bit for Handel C 2.1 and Celoxica DK1. In: *Celoxica User Conference.*

Proceedings. Celoxica, Abington 2001, 202 kB.

[A6] Líčko M., Pohl Z., Matoušek R., Heřmánek A.: Tuning and implementation of DSP algorithms on FPGA. In: *Sborník příspěvků 9. ročníku konference MATLAB 2001*. (Procházka A., Uhlíř J. eds.). VŠCHT, Praha 2001, pp. 226-230.

[A7] Albu F., Kadlec J., Heřmánek A., Fagan A., Coleman N.: Analysis of the LNS implementation of the fast affline projection algorithms. In:*Proceedings of the Irish Signals and Systems Conference 2002*. ISSC 2002. (Marnane W., Lightbody G., Pesch D. eds.). Institute of Technology, Cork 2002, pp. 251-255.

[A8] Heřmánek A., Regalia P.: Recursive Finite Interval Constant Modulus Algorithm for blind equalization. In: *Sborník příspěvků 10. ročníku konference MATLAB 2002* VŠCHT, Praha 2002, pp. 133-141.

[A9] Kadlec J., Tichý M., Heřmánek A., Pohl Z., Líčko M.: Matlab Toolbox for high-level bit-exact emulation of HandelC VHDL FPGA designs. In: *Design, Automation and Test in Europe DATE'02*. (Sciuto D., Kloos C. D. eds.). IEEE, Los Alamitos 2002, pp. 264.

[A10] Matoušek R., Pohl Z., Kadlec J., Tichý M., Heřmánek A.: Logarithmic arithmetic core based RLS LATTICE implementation. In: *Design, Automation and Test in Europe DATE'02*. (Sciuto D., Kloos C. D. eds.). IEEE, Los Alamitos 2002, pp. 271.

[A11] Heřmánek A., Pohl Z., Kadlec J.: FPGA implementation of the adaptive lattice filter. In: *Field-Programmable Logic and Applications. Proceedings of the 13th International Conference*. (Cheung P. Y. K., Constantinides G. A., de Sousa J. D. eds.). (Lecture Notes in Computer Science. 2778). Springer, Berlin 2003, pp. 1095-1098.

[A12] Heřmánek A., Regalia P.: Comparison of two recursive constant modulus algorithms. In:*Proceedings of the 4th Electronic Circuits and Systems Conference*.

(Butad J., Stopjakov V. eds.). Slovak University of Technology, Bratislava 2003, pp. 159-162.

[A13] Pohl Z., Schier J., Líčko M., Heřmánek A.,Tichý M.: Logarithmic arithmetic for real data types and support for Matlab/Simulink based rapid-FPGA-prototyping. In: *Proceedings of the International Parallel and Distributed Processing Symposium.* IPDPS 2003. (Werner B. ed.). IEEE Computer Society Press, Los Alamitos 2003, pp. 1-6, 149 kB

[A14] Heřmánek A., Schier J., Regalia P.: Architecture design for FPGA implementation of finite interval CMA. In: *Proceedings of the 12th European Signal Processing Conference.* (Hlawatsch F., Matz G., Rupp M. eds.). University of Technology, Vienna 2004, pp. 1-4, 250 kB

[A15] Schier J., Heřmánek A.: FPGA implementation of recursive QR update using LNS arithmetic. In: *Proceedings of the 4th IEEE Benelux Signal Processing Symposium.* Technical University, Delft 2004, pp. 1-4, 144 kB

[A16] Schier J., Heřmánek A.: Using logarithmic arithmetic to implement the Recursive Least Squares (QR) algorithm in FPGA. In: *Field-Programmable Logic and Applications. 14th International Conference FPL 2004.* Proceedings. (Becker J., Platzner M., Vernalde S. eds.). (Lecture Notes in Computer Science. 3203). Springer, Berlin 2004, pp. 1149-1151.

Research reports

[B1] Albu F., Kadlec J., Softley C., Matoušek R., Heřmánek A.: *Implementation of Normalized RLS Lattice on Virtex. (Research Report No. 2040).* ÚTIA AV ČR, Praha 2001, 10 pp.

[B2] Coleman J. N., Kadlec J., Matoušek R., Pohl Z., Heřmánek A.: *The European Logarithmic Microprocessor - a QRD RLS Applications. (Research Report No. 2038).* ÚTIA AV ČR, Praha 2001, 9 pp.

[B3] Heřmánek., Kadlec J., Matoušek R., Líčko M., Softley C.: *Pipelined Logarithmic 32bit ALU for Celoxica DK1. (Research Report No. 2034).* ÚTIA AV ČR, Praha 2001, 11 pp.

[B4] Kadlec J., Albu F., Softley C., Matoušek R., Heřmánek A.: *RLS Lattice for Virtex FPGA using 32-bit and 20-bit Logarithmic Arithmetic. (Research Report No. 2036).* ÚTIA AV ČR, Praha 2001, 11 pp.

[B5] Kadlec J., Heřmánek A., Softley C., Matoušek R., Líčko M.: *32-bit Logarithmic ALU for Handel-C 2.1 and Celoxica DK1. (Research Report No. 2037).*ÚTIA AV ČR, Praha 2001, 12 pp.

[B6] Matoušek R., Líčko M., Heřmánek A., Softley C.: *Floating-Point-Like Arithmetic for FPGA. (Research Report No. 2039).* ÚTIA AV ČR, Praha 2001, 7 pp.

[B7] Líčko, Schier J., Pohl Z., Kadlec J., Tichý M., Matoušek R., Heřmánek A.: *Logarithmic Arithmetic for Real Data Types and Support for MATLAB/SIMULINK Based Rapid-FPGA-Prototyping. (Research Report No. 2069).* ÚTIA AV ČR, Praha 2002, 7 pp.

Unreferred workshops and posters

[C1] Líčko M., Tichý M., Heřmánek A., Matoušek R., Pohl Z.: Prototyping of DSP algorithms on FPGA. In: *POSTER 2002.* FEL ČVUT, Praha 2002, pp. 2.

[C2] Matoušek R., Líčko M., Heřmánek A., Softley C.: Floating-Point-Like Arithmetic for FPGA. In: *POSTER 2002.* FEL ČVUT, Praha 2002, pp. 2.