Computationally Efficient Multiuser and MIMO Detection based on Dichotomous Coordinate Descent Iterations

This thesis is submitted in partial fulfilment of the requirements for the degree of

PhD of Science

May 2009

Zhi Quan

Communications Research Group

Department of Electronics

University of York

Supervisor: Yuriy Zakharov

Thesis Advisor: Rodrigo de Lamare

Abstract

The detection in multiuser (MUD) and multiple-input multiple-output (MIMO) systems can increase the spectral efficiency, and therefore is of great interest. Although multiuser and MIMO detection is mature in theory, the real-time implementation is still an open issue. Many suboptimal detection schemes have been proposed, possessing low computational load, but also having poorer detection performance compared to the optimal detector. Multiuser detection can be described as a solution of an optimisation problem; in most cases it is the quadratic optimisation problem. Unconstrained quadratic optimisation is known to result in decorrelating and MMSE multiuser detection, which cannot provide high detection performance. The optimal detection is equivalent to the solution of a constrained problem. However, such detection is too complex for practical systems. In this work, we propose several detectors which possess low complexity and high detection performance. These detectors are based on Dichotomous Coordinate Descent (DCD) iterations, which are multiplication and division free, and therefore are attractive for real-time implementation. We propose a box-constrained DCD algorithm, and apply it to multiuser detection. We also design an FPGA architecture of the box-constrained DCD detector and implement it in an FPGA. This design requires a very small area usage. The fixed-point implementation offers a constant throughput over the signal-to-noise ratio (SNR) and provides almost same detection performance as that of a floating-point implementation. We further exploit the box-constrained DCD algorithm and propose a complex-valued box-constrained DCD algorithm. A box-constrained MIMO detector based on the DCD algorithm shows a better detection performance than the MMSE detector. The proposed FPGA design requires a small area usage, which is significantly less than that required by known designs of the MMSE MIMO detector. Since the box-constrained DCD algorithm could not offer the optimal detection performance, while the sphere decoder encounters high complexity at low SNRs, we suggest a combination of the box-constrained DCD algorithm with the sphere decoder (fast branch and bound algorithm). The combined detection results in reduced complexity at low SNRs while retaining outstanding detection performance at all SNRs. As the box-constrained DCD algorithm is efficient for hardware implementation, we apply it to the nonstationary iterative Tikhonov regularization and propose a DCD-BTN detector. The DCD-BTN detector shows the detection performance very close to the optimal performance. It also shows the lowest complexity among the most advanced detectors. An architecture of the detector has been developed. This detector has been implemented on an FPGA platform. The design requires a small number of FPGA slices. Numerical results have shown that the fixed-point FPGA implementation and a floating-point implementation have similar detection performance. The DCD-BTN detector can only be applied in systems with BPSK modulation. Therefore, we also propose a multiple phase decoder (MPD), which is based on a phase descent search (PDS) algorithm. The PDS algorithm uses coordinate descent iterations, where coordinates are unknown symbol phases, while constraining the symbols to have a unit magnitude. The MPD is investigated in application to detection of M-PSK symbols in multiuser and MIMO systems. In the multiuser detection, the MPD is applied to highly loaded scenarios and numerical results show that it provides the near-optimal performance at low complexity. The MPD significantly outperforms such advanced detector as the semidefinite relaxation detector in both the detection performance and complexity. In MIMO systems, the MPD exhibits more favorable performance/complexity characteristics and can be considered as a promising alternative

to the sphere decoder. The matrix inversion is required in many applications. The complexity of matrix inversion is too high and makes its implementation difficult. To overcome the problem, we propose an approach based on the DCD algorithm to simplify the matrix inversion. This approach obtains separately the individual columns of the inverse matrix and costs a very small number of slices, which is suitable for application, *e.g.* in MIMO-OFDM systems.

Acknowledgements

The work described in this thesis could not have been accomplished without the help and support of others. Firstmost, I would like to thank my supervisor Yuriy Zakharov for his guidance and support during the past four years. I have had the pleasure of working with many graduate students. Roy Zhang has helped me to better understand many aspects of wireless communications. I felt so lucky to do my research with someone who can clearly explain the concepts. Thanks to Ben Weaver for all the help and advice, especially during my first year at York. Thanks Dr. J. Rockey Luo from Colorado State University, for providing the decision feedback, probability data association and fast branch and bound MATLAB programs. I had a great time with Rami, Eric and Terry in my office, thanks for your guys' accompanying and encouragement. Thanks to Dan Shanchen for helping my research work. I would like to thank the fellow members of the Communications Research Group, for their help and support throughout my PhD research. Most of all thanks to Jie Liu, teaching me how to use FPGA tools, being so patient and helping whenever I had a problem with design and debugging. Finally, I would like to express my deepest appreciation for my family and friends. Thanks to Gaosu Xia for being such a great friend ever since college. Thanks to Grandfather for always supporting and having faith in me. Thanks to my parents unconditional love and continual encouragement, which have been a source of strength. I dedicate this work to you with all my love.

Declaration

Some of the research presented in this thesis has resulted in some publications. These publications are listed at the end of the thesis.

All work presented in this thesis as original is so, to the best knowledge of the author. References and acknowledgements to other researchers have been given as appropriate.

Glossary

AWGN	Additive White Gaussian Noise
ASIC	Application Specific Integrated Circuit
BB	Branch and Bound
BER	Bit Error Ratio
BLAST	Bell Labs Layered Space Time
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
DCD	Dichotomous Coordinate Descent
DF	Decision Feedback
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
GDE	Group Detection Error
i.i.d	independent identically distributed
IC	Interference Cancelation
ISI	Inter-Symbol Interference
MAI	Multiple Access Interference
ML	Maximum Likelihood
MF	Matched Filter
MIMO	Multiple-Input and Multiple-Output
MMSE	Minimum Mean Squared Error
MUD	Multiuser Detection
MPD	Multiple Phase Decoder
OFDM	Orthogonal Frequency-Division Multiplexing
PDA	Probabilistic Data Association
PDF	Probability Density Function
PED	Partial Euclidean Distances
PDS	Phase Descent Search
PSK	Phase Shift Keying
QP	Quadratic Programming
QAM	Quadrature Amplitude Modulation
RAM	Random-access Memory
SISO	Single-Input Single-Output
SIMO	Single-Input Multiple-Output
SDR	Semi-definite Relaxation

SER	Symbol Error Rate
SM	Spatial Multiplexing
SNR	Signal to Noise Ratio
UMTS	Universal Mobile Telecommunications System
V-BLAST	Vertical BLAST
WCDMA	Wideband Code Division Multiple Access
ZF	Zero-Forcing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Research Area

In CDMA systems, all users operate in the same frequency band, so multiple-access interference (MAI) occurs in code-division channels. Multiuser detection (MUD) is used to demodulate one user's data stream from a non-orthogonal multiplex, which can achieve a significant increase in the spectral efficiency of the systems [2] [3]. The conventional approach is the matched filter (MF), however, it is vulnerable to near-far problem and the performance is far from the ideal [3]. Maximum likelihood (ML) multiuser detector provides the optimal detection performance, however it is highly complex [3]. The decorrelating and the minimum mean squared error (MMSE) linear detectors take into account MAI and enforce near-far resistance, which is ignored in the matched filter [4]. However, they require matrix inversion which costs intensive complexity.

Recently, wireless communication research has focused on multiple-input multiple-output (MIMO) communication systems. The previous work showed that by using multiple antennas at the transmitting and the receiving ends allowed to increase the capacity and reliability [5]. It is believed that MIMO and multiuser detection techniques will play an important role in the development of wireless mobile systems. MIMO communication system is mathematically formulated similarly to CDMA MUD system but normally deals with small size systems. The sphere decoding algorithm approaches the optimal ML performance while reducing the complexity significantly [6, 7], however the complexity of the sphere decoder depends on SNR and the modulation/ constellation, *i.e.* , its worst-case computational complexity is exponentially proportional to the number of users (or transmit antennas), especially at low signal-to-noise ratios (SNRs) [3].

Overall, when these algorithms are not computationally efficient, and a great deal of hardware resources will be consumed, that limit their practical real-time applications. This is especially true when systems with many users are of interest and over loaded systems are considered. This work is focused on the development of efficient multiuser and MIMO detectors. The attention is concentrated on algorithms derived from the Dichotomous Coordinate Descent (DCD) algorithm [8]. The DCD algorithm is based on the coordinate descent technique with power of two variable step-size, which is easily realized by using the bit-shift operation in hardware implementation. Therefore this algorithm can be considered as a promising technique for multiuser and MIMO detection. The goal of this work is to explore state-of-the-art algorithms for the multiuser detection and MIMO systems. For this purpose, the FPGA architectures for various detectors have been developed in this thesis.

1.2 Overview of Advanced Multiuser Detection and MIMO Technologies

1.2.1 System Models for Multiuser Detection

We first describe the channel model corresponding to an additive white Gaussian noise (AWGN). Then, we show that this system model also describes flat and frequency-selective channels, channels with multipath propagation, fast and slow fading channels.

Various channel models can be converted into a synchronous CDMA system as a generic system model for multiuser detection. The spreading sequence in DS-CDMA multiuser detection is similar to the spatial signature in MIMO systems. Solving the MIMO communication problems is mathematically similar to that of the multiuser detection, therefore the model for multiuser detection can be extended for the MIMO systems.

Synchronous CDMA Systems

In a synchronous CDMA system, the received signal is given by

$$y(t) = \sum_{k=1}^{K} A_k h_k s_k(t) + n(t)\,, \tag{1.1}$$

where

- $s_k(t)$ is a signature waveform for the kth user. In CDMA systems, the signature waveform $s_k(t)$ is represented by

$$s_k(t) = [p(t), p(t - T_c), \cdots, p(t - (m-1)T_c)]\mathbf{s}_k\,, \tag{1.2}$$

  where $p(t)$ is the chip waveform and T_c is the chip period, $\mathbf{s}_k$ is a $m \times 1$ vector and known as the spreading sequence for user k; m is the spreading factor [9].

- h_k is the input symbol of the kth user.

- A_k is the received amplitude of the kth user symbol.

- $n(t)$ is the zero-mean additive white Gaussian noise with power spectral density σ^2.

The cross-correlation of the signature waveform is defined as

$$\rho_{ij} = \int_0^{T_s} s_i(t) s_j(t) dt\,, \tag{1.3}$$

where T_s is the symbol duration. We define the cross-correlation matrix as: $\mathbf{R} = \{\rho_{ij}\}$; $\mathbf{R}$ is a $K \times K$ symmetric, non-negative definite matrix.

Asynchronous CDMA Systems

In the case of asynchronous transmission, the received signal is given by [9]

$$y(t) = \sum_{k=1}^{K} \sum_{i=0}^{N} A_k h_k[i] s_k(t - iT_s - \tau_k) + n(t)\,, \tag{1.4}$$

where t is the transmission time, τ_k is the delay in transmission of the kth user, $h_k[i]$ is the ith transmitted symbol of the kth user, $(N+1)$ is the data block length. We assume that users send a stream of symbols: $h_k[0], \cdots, h_k[N]$. If the channel is known, *i.e.* the receiver knows all the delays τ_k and channel gains A_k, we can consider an asynchronous CDMA system as a special case of the synchronous CDMA when the number of users is $K(N+1)$. The modified spreading sequence is given by

$$\tilde{s}_{k,i}(t) = s_k(t - iT_s - \tau_k). \tag{1.5}$$

The modified channel gain is $\tilde{A}_{k,i} = A_k$ and the modified data symbols are $\tilde{h}_{k,i} = h_k[i]$, where $k = 1, \cdots, K$ and $i = 0, \cdots, N$. The modified signal model can be represented as

$$y(t) = \sum_{k=1}^{K(N+1)} \tilde{A}_k \tilde{h}_k \tilde{s}_k(t) + n(t). \tag{1.6}$$

Therefore, the asynchronous CDMA system can be referred to as a special case of the synchronous CDMA system model in (1.1) when the number of users increases from K to $K(N+1)$ [9].

1.2.2 Channel Models

When the broadband wireless connection is concerned, the symbol rate must be increased further which to some extend will lead to a frequency selective channel. The recent development of the communication technology has built on great interest in multi-antenna systems as an effective technique to combat fading and reduce the effect of channel interference. For help understanding these techniques, this section gives a brief introduction to fading channels.

Flat Fading

Flat fading is caused when the radio channel coherence bandwidth is greater than the bandwidth of the transmitted signal and the channel has a constant gain and linear phase response. Slow frequency-flat fading affects the received amplitude without introducing signature waveform distortion. Therefore, the mathematical formula for representing the synchronous CDMA system model (1.1) is also suitable for characterizing the slow frequency flat fading channels. Fast frequency-flat fading affects the received amplitudes and introduces signature waveform distortion. The modified signature waveform is introduced as

$$\tilde{s}_k(t) = A_k(t)s_k(t)\,, \tag{1.7}$$

hence the synchronous CDMA model (1.1) is still applicable in fast frequency flat fading channels:

$$y(t) = \sum_{k=1}^{K} h_k \tilde{s}_k(t) + n(t)\,, \qquad t \in [0, T_s]. \tag{1.8}$$

Frequency Selective Fading

The flat fading channel model assumes that the symbols of one signal propagating through different paths reach the receiver simultaneously. However, in practical systems, different paths normally have distinct path delays. It is known as the frequency-selective fading channel. In frequency-selective fast fading channel, the signature waveform of the kth user undergoes a linear time-variant transformation characterized by an impulse response: $r_k(\tau, t)$. The transformed signature waveform in the frequency selective fading channel $\tilde{s}_k(t)$ can be represented by

$$\tilde{s}_k(t) = \int_{-\infty}^{\infty} r_k(\tau, t) s_k(t-\tau) d\tau. \tag{1.9}$$

Then the synchronous CDMA model can still be used as

$$y(t) = \sum_{k=1}^{K} h_k \tilde{s}_k(t) + n(t). \tag{1.10}$$

Equation (1.10) indicates that the frequency selective fading channel model can be transformed into a synchronous CDMA system model.

1.2.3 Multiuser Detectors

Multiple access interference significantly limits the performance and capacity of transmission systems. Multiuser detection reduces the interference and combats the near-far problems [10]. The optimal, *i.e.* Maximum Likelihood (ML) multiuser detector [9] provides detection performance close to that of the single user detection but the complexity is exponentially proportional to the number of users. The decorrelating detector eliminates the MAI but enhances the noise power [10]. Minimum Mean Square Error (MMSE) detector [9] has better performance than the decorrelating detector but it requires the estimation of amplitudes and the matrix inversion. The Decision-Feedback (DF) detector [9] is one of the most popular methods because of the simplicity and good performance. Sphere constrained and box-constrained algorithms allow the solutions to lie within a closed convex set [11]; this significantly improves the performance. Semi-definite relaxation (SDR) [12] relaxes the ML problem into a semi-definite problem, and provides a BER performance very close to the ML detector. The Probabilistic Data Association (PDA) detector treats MAI as Gaussian noise with matched mean and covariance, and offers a detection performance close to the ML detector [13, 14]. The Sphere Decoding (SD) and branch and bound (BB) detectors achieve optimal performance, however, its worst-case computational complexity is too high. A comparison of these advanced multiuser detection techniques in [15], in terms of complexity and detection performance, has shown that an "efficient frontier" of multiuser detectors is primarily composed of the DF detector, PDA detector, and BB detector.

Conventional Matched Filter

The conventional detector known as the matched filter detector, correlates the received signal with the desired user's spreading waveform as presented in Fig.1.1. The output θ_k of the kth matched filter is given by:

$$\theta_k = \int_0^{T_s} y(t)s_k(t)dt\,, \tag{1.11}$$

where $y(t)$ is the received signal. Therefore we can write

$$\theta_k = \int_0^{T_s} \left\{ \sum_{j=1}^{K} A_j h_j s_j(t) + n(t) \right\} s_k(t)dt\,. \tag{1.12}$$

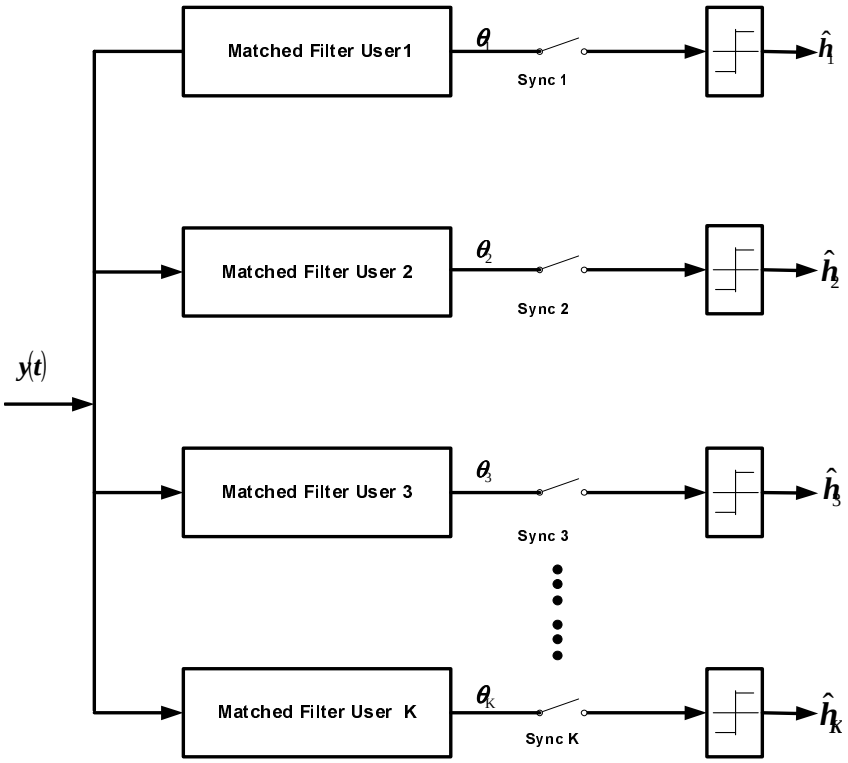


Figure 1.1: Conventional Matched Filter Detector

Applying equation (1.3) to (1.12), we obtain

$$\theta_k = \sum_{j=1}^{K} A_j h_j \rho_{jk} + n_k \,, \tag{1.13}$$

where

$$n_k = \int_0^{T_s} n(t) s_k(t) dt. \tag{1.14}$$

Thus, we have

$$\theta_k = A_k h_k + \sum_{\substack{j=1 \\ j \neq k}}^{K} A_j h_j \rho_{jk} + n_k. \tag{1.15}$$

The second term in (1.15) is the multiple access interference. The matched filter treats the MAI as additive white Gaussian noise. Nevertheless, the existence of MAI has significant impact on the capacity and performance of the conventional matched filter detectors. When the number of interfering users increases, the effect of MAI is significantly enhanced. In addition, users with large amplitudes result in more MAI effects to the users with low amplitudes. The signals of closer transmitting users have less amplitude attenuation than signals of transmitting users which are further away. That is known as the near-far problem [9] [16]. The conventional matched filter detector requires no knowledge beyond the spreading sequences. However, when the number of users increases, the matched filter will result in poor detection performance [9].

Maximum Likelihood Detector

Consider a K-user symbol synchronous CDMA system in AWGN channel. The output of the matched-filter is given by

$$\boldsymbol{\theta} = \mathbf{RAh} + \mathbf{n}\,, \tag{1.16}$$

where (for BPSK modulation) the vector $\mathbf{h} \in \{-1, +1\}^K$ contains the information symbols transmitted by K users, $\mathbf{R}$ is a positive definite spreading sequence correlation matrix, $\mathbf{A}$ is a diagonal matrix whose kth diagonal element, A_{kk}, is the square root of the received signal energy per bit of the kth user, and $\mathbf{n}$ is a real-valued zero-mean Gaussian random vector with covariance matrix $\sigma^2\mathbf{R}$.

The optimal ML multiuser detector estimates the vector $\mathbf{h}$ by minimizing the following quadratic function [9]

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h}\in\{-1,+1\}^K} J(\mathbf{h})\,, \tag{1.17}$$

where the quadratic function $J(\mathbf{h})$ is represented by

$$J(\mathbf{h}) = \|\boldsymbol{\theta} - \mathbf{Rh}\|^2 \Longrightarrow \mathbf{h}^T\mathbf{ARAh} - 2\mathbf{h}^T\mathbf{A}\boldsymbol{\theta}. \tag{1.18}$$

The computational complexity of the ML detection is $\mathcal{O}(2^K)$ arithmetic operations. Although the ML detector provides the best detection performance, its complexity is exponential in the number of users and makes it difficult for hardware implementation.

Decorrelating Detector

In the model (1.16), the transmitted data can be recovered as

$$\begin{aligned} \hat{\mathbf{h}} &= \text{sign}(\mathbf{R}^{-1}(\mathbf{RAh} + \mathbf{n})) \\ &= \text{sign}(\mathbf{Ah} + \mathbf{R}^{-1}\mathbf{n}). \end{aligned} \tag{1.19}$$

If $\sigma = 0$, then $\hat{\mathbf{h}} = \text{sign}(\mathbf{h})$. This detector is called decorrelating detector. The advantage of the decorrelating detector is that it does not require knowledge of the received signal amplitudes. However, when the matrix $\mathbf{R}$ is ill-conditioned, the term of $\mathbf{R}^{-1}\mathbf{n}$ in (1.19) results in noise power enhancement and as a result the error probability increases.

MMSE Detector

The MMSE detector takes into account the background noise and uses the knowledge of the received signal powers, which leads to a better detection performance compared to the decorrelating detector for low values of SNR. The MMSE detector minimizes the mean squared error between the actual symbol and the decision output of the detector [17]. The inverse of matrix $\mathbf{R}$ in the decorrelating detection is now replaced by:

$$[\mathbf{R} + \sigma^2 \mathbf{A}^{-2}]^{-1}. \tag{1.20}$$

The MMSE detector has a trade-off between MAI elimination and noise enhancement. This detector obtains the performance similar to the conventional MF when the noise variance tends to infinity. When the SNR goes to infinity, the MMSE detector will converge to the decorrelating detector [9, 18]. The MMSE detector is resistant to the near-far problem.

Decision Feedback Detector

The Decision Feedback (DF) detectors have been examined in successive interference cancellation [19–21], parallel interference cancellation [22–24] and multistage or iterative DF detectors [23], [24]. The DF detector with successive interference cancellation (S-DF) is optimal, in the sense that it achieves the sum capacity of the the synchronous AWGN channel [20]. The S-DF scheme is capable of alleviating the effects of error propagation despite it generally leads to non uniform performance over the users. In particular, the user ordering plays an important role in the performance of S-DF detectors. Studies on decorrelator DF detectors with optimal user ordering have been reported in [21] for imperfect feedback and in [25] for perfect feedback. The problem with the optimal ordering algorithms in [21], [25] is that they represent a very high computational burden for practical receiver design. On the contrary, the DF receiver with parallel interference cancellation (P-DF) [22–24] satisfies the uplink requirements, *i.e.* , cancellation of intra-cell interference and suppression of the remaining other-cell interference, and provides, in general, uniform performance over the user population even though it is more sensitive to error propagation. The multistage or iterative DF schemes presented in [23, 24] are based on the combination of S-DF and P-DF schemes in multiple stages in order to refine the symbol estimates, resulting in improved performance over conventional S-DF, P-DF and mitigation of error propagation [26].

Semi-definite Relaxation Detector

Relaxation is an effective approximation technique for certain difficult optimization problems. Relaxing some constraints will simplify the problem solving. The semi-definite relaxation algorithm reduces computational complexity without losing performance. For solving the Boolean QP problem [27–29]

$$\arg\min_{\mathbf{x}} \mathbf{x}^T\mathbf{B}\mathbf{x} \tag{1.21}$$

where $\mathbf{B}$ is any symmetric matrix. Since $\mathbf{x}^T\mathbf{B}\mathbf{x} = \mathrm{Trace}(\mathbf{x}\mathbf{x}^T\mathbf{B})$, problem (1.21) can be restated as

$$\begin{aligned} &\arg\min_{\mathbf{X}} \mathrm{Trace}(\mathbf{B}\mathbf{X}) \\ &\text{s.t. } \mathrm{diag}(\mathbf{X}) = \mathbf{e} \\ &\mathbf{X} = \mathbf{x}\mathbf{x}^T. \end{aligned} \tag{1.22}$$

where $\mathbf{e}$ is the vector all ones and where

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}^T\mathbf{R} & -\mathbf{R}^T\boldsymbol{\theta} \\ -\boldsymbol{\theta}^T\mathbf{R} & \boldsymbol{\theta}^T\boldsymbol{\theta}, \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{h} \\ 1 \end{bmatrix}. \tag{1.23}$$

The constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^T$ implies that $\mathbf{X}$ is symmetric, positive semi-definite and of rank-1. Because of the constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^T$, problem (1.22) is a non-convex optimization problem. When the rank-1 constraint is removed from (1.22), we can get the following problem

$$\begin{aligned} &\arg\min_{\mathbf{X}} \ \mathrm{Trace}(\mathbf{B}\mathbf{X}) \\ &\text{s.t. } \mathbf{X} \succeq 0 \\ &X_{jj} = 1,\ j = 1, \cdots, K. \end{aligned} \tag{1.24}$$

where $\mathbf{X} \succeq 0$ means that $\mathbf{X}$ is symmetric and positive semi-definite. Problem (1.24) is the relaxation of problem (1.22) because the feasible set in (1.22) is a subset of that in (1.24). The problem in (1.24) is considered as the semidefinite relaxation of (1.22). The problem of (1.24) can be solved in the order of $\mathcal{O}(K^{3.5})$ operations. The semi-definite relaxation detector provides a BER performance close to that of the ML detector, even when the cross-correlations between users are strong or the near-far effect is significant. However, the semi-definite relaxation detector is very complex for large systems [15, 28].

Constrained Multiuser Detectors

The ML detector for BPSK modulation finds a solution constrained to $\mathbf{h} \in \{-1, +1\}^K$, where $\{-1, +1\}^K$ denotes the set of all binary symbols and each symbol is either $+1$ or -1. However the ML detector has been proved to be too complex for practical use [10]. A simple limitation by constraining the symbol estimate vector to be within a closed convex set often reduces the complexity. Constraining the symbol estimate to lie within a hypercube leads to a box-constrained quadratic problem (*e.g.* $K = 2$) and is shown in Fig.1.2a [30]. When applying the box-constraint to solve the ML problem in a CDMA

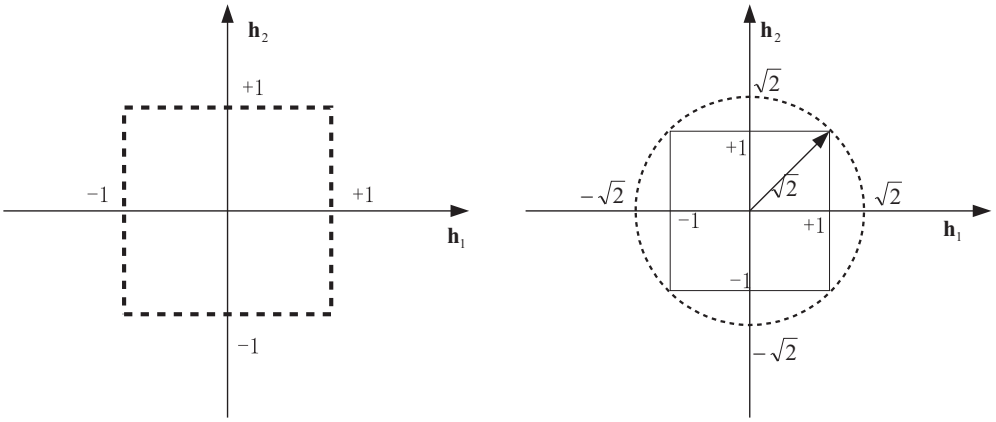


Figure 1.2: (a) Projection onto a box region; (b) Projection onto a sphere region

system with BPSK, the problem is reformulated as

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in [-1,+1]^K} J(\mathbf{h}). \tag{1.25}$$

We define an orthogonal projection operation P_B on a hypercube closed convex set $\mathcal{B}$ ($\mathcal{B} = [-1, +1]$), which is represented as

$$P_B(\hat{\mathbf{h}}) = \arg \min_{\mathbf{h} \in \mathcal{B}} \|\mathbf{h} - \hat{\mathbf{h}}\|, \tag{1.26}$$

where $P_B(h_k)$ is represented as

$$\begin{cases} h_k & \text{if} \quad -1 < h_k < 1 \\ -1 & \text{if} \quad h_k \leq -1 \\ +1 & \text{if} \quad h_k \geq +1. \end{cases} \tag{1.27}$$

The sphere-constraint ML problem [31] is described in Fig.1.2b:

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}\in\mathcal{S}} J(\mathbf{h})\,, \tag{1.28}$$

where $\mathcal{S} = \{\mathbf{h} \in \mathcal{R}^K : \|\mathbf{h}\|^2 \leq K\}$. We assume $P_S(h_k)$ is the kth element of the orthogonal projection onto a sphere, which is represented as

$$\begin{cases} \alpha h_k & \text{if} \quad \|\mathbf{h}\|^2 > K \\ h_k & \text{if} \quad \|\mathbf{h}\|^2 \leq K\,, \end{cases} \tag{1.29}$$

where $\alpha = \sqrt{K}/\|\mathbf{h}\|$, and $0 < \alpha \leq 1$. A multiuser detector with sphere constraint provides a detection performance similar to the MMSE detector [32]. Furthermore, Fig.1.2b also shows that the sphere region contains the box-region $[-1, +1]$, which means the detector with the box constrained having a more restricted condition, thus resulting in a better performance than the sphere constrained detector [15, 32, 33].

Probabilistic Data Association Detector

When multiplying $\mathbf{A}^{-1}\mathbf{R}^{-1}$ both sides of the matched filter output vector $\boldsymbol{\theta}$ in (1.16), the result can be represented as [14]

$$\bar{\boldsymbol{\theta}} = \mathbf{h} + \bar{\mathbf{n}} = h_i\mathbf{e}_i + \sum_{j\neq i} h_j\mathbf{e}_j + \bar{\mathbf{n}}\,, \tag{1.30}$$

where $\bar{\boldsymbol{\theta}} = \mathbf{A}^{-1}\mathbf{R}^{-1}\boldsymbol{\theta}$, $\bar{\mathbf{n}} = \mathbf{A}^{-1}\mathbf{R}^{-1}\mathbf{n}$ and $\mathbf{e}_i$ is a column vector with a one in the kth position and zeros elsewhere. The equation (1.30) is in fact a normalized version of the decorrelator output before the hard decision. The decision on user i can be considered as binary variables $+1$ or -1, with the currently estimated probabilities $P_h(i)$ and $1 - P_h(i)$, respectively. *i.e.* $P_h(i)$ is the current estimate of the probability that $h_i = 1$ and $1 - P_h(i)$ is the corresponding estimate for $h_i = -1$. For an arbitrary user signal h_i, we treat the other user signals $h_j(j \neq i)$ as binary random variables and treat $\sum_{j\neq i} h_j e_j + \bar{\mathbf{n}}$ as the effective noise. Based on the decorrelated model, the basic multistage PDA algorithm is described as follows [14].

1. Sort users according to the user ordering principle for the decision feedback detector in [21].

2. For all users, initializing $P_h(i) = 0.5$ and stage counter $k = 1$.

3. Initialize the user counter $i = 1$.

4. According to the current $P_h(j)(j \neq i)$ for user i, update $P_h(i)$ by

$$P_h(i) = P\{h_i = 1|\bar{\boldsymbol{\theta}}, \{P_h(j)\}_{j\neq i}\} \tag{1.31}$$

5. If $i < K$, let $i = i + 1$ and go to step 4.

6. If $\forall i$, $P_h(i)$ has converged, go to step 7. Otherwise, let $k = k + 1$ and return to step 3.

7. $\forall i$, make a decision on user signal i, *i.e.* h_i via

$$\begin{cases} 1 & P_h(i) \geq 0.5 \\ -1 & P_h(i) < 0.5. \end{cases} \tag{1.32}$$

Since the computational cost of obtaining (1.31) is exponential in the number of users. We define

$$\breve{\mathbf{n}}_i = \sum_{j\neq i} h_j \mathbf{e}_j + \bar{\mathbf{n}}\,. \tag{1.33}$$

We denote the mean and covariance matrix of $\breve{\mathbf{n}}_i$ as

$$\begin{aligned} E(\breve{\mathbf{n}}_i) &= \sum_{j\neq i} \mathbf{e}_j(2P_h(j) - 1) \\ Cov(\breve{\mathbf{n}}_i) &= \sum_{j\neq i} [4P_h(j)(1 - P_h(j))\mathbf{e}_j\mathbf{e}_j^T] + \sigma^2\mathbf{R}^{-1}\,. \end{aligned} \tag{1.34}$$

Correspondingly we define $\boldsymbol{\Phi}_i = E(\breve{\mathbf{n}}_i)$ and $\boldsymbol{\Psi}_i = Cov(\breve{\mathbf{n}}_i)$. The updated probability $P_h(i)$ is given by

$$\frac{P_h(i)}{1 - P_h(i)} = \exp\{-2\boldsymbol{\Phi}_i^T\boldsymbol{\Psi}_i^{-1}\mathbf{e}_i\} \tag{1.35}$$

The algorithm continues till all the probabilities $\{P_{h_i}\}$ have converged. The detection performance approaches the single user performance over high SNRs. More details are shown in [14]. The overall complexity of the PDA detector is $\mathcal{O}(K^3)$ [34, 35].

Dichotomous Coordinate Descent Algorithm

In many communications systems, the detection is based on the solution of a system $\mathbf{Rh} = \boldsymbol{\theta}$, where $\mathbf{R}$ is a $K \times K$ symmetric positive definite matrix and both $\mathbf{h}$ and $\boldsymbol{\theta}$ are $K \times 1$ vectors. The matrix $\mathbf{R}$ and vector $\boldsymbol{\theta}$ are known, solution $\mathbf{h}$ is unknown. The DCD

Table 1.1: Dichotomous Coordinate Descent Algorithm

Initialization: $\mathbf{h} = \bar{\mathbf{h}}$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, $d = H$, $p = 0$.
for $m = 1 : M_b$
1. $d = d/2$
2. Flag $= 0$ $\cdots pass$
3. for $j = 1 : K$ $\cdots iteration$
4. if $|r(j)| > (d/2)R(j,j)$ then
5. $h(j) = h(j) + \text{sign}(r(j))d$
6. $\mathbf{r} = \mathbf{r} - \text{sign}(r(j)) \cdot d \cdot \mathbf{R}(:, j)$
7. Flag $= 1, p = p + 1$
8. if $p > N_u$ end algorithm
9. end j-loop
10. if Flag=1, go to 2

end m-loop

algorithm [8] is designed to offer a simple solution for the vector $\mathbf{h}$, without explicit multiplications and divisions. The accuracy of the solution vector $\mathbf{h}$ depends on the number of bits (M_b), which is the number of the bits used for the representation of elements of the vector $\mathbf{h}$ within an amplitude range $[-H, H]$. The first set of iterations in the algorithm determines the most significant bit ($m = 1$) for all elements of $\mathbf{h}$ using a step size parameter d. The subsequent sets of iterations determine the less significant bits up to a suitable number of bits (maximally M_b). The initial residual vector is given by $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, where $\bar{\mathbf{h}}$ is the initialization of $\mathbf{h}$. Table 1.1 describes the DCD algorithm. We denote $h(j)$ and $r(j)$ the elements of the vectors $\mathbf{h}$ and $\mathbf{r}$ respectively. In case the vector $\bar{\mathbf{h}}$ is set to zero, $\mathbf{r}$ is equal to $\boldsymbol{\theta}$. The step-size is set to H and successful iteration counter p is set to 0. The step-size d is reduced by power of two at step 1, and so, no explicit multiplication or division is carried out as all the multiplications and divisions can be replaced by simple bit shifts. If an element of the solution vector is updated at an iteration, such an iteration is labeled "successful". For every step size update, the algorithm repeats successful iterations until all elements of the residual vector $\mathbf{r}$ become so small that the condition at step 4 is not met for all j. The computational load of the algorithm mainly depends on these successful update iterations p and the number of bits M_b. A limit for the number of successful iterations N_u can be predefined and used as a stopping condition. If there is no such limit, or the limit is high enough, the accuracy of the solution is 2^{-M_b+1}. In our work, wherever the DCD algorithm is used, we denote it as DCD($\bar{\mathbf{h}}, \mathbf{R}, \boldsymbol{\theta}, N_u, M_b$). The complexity of the DCD algorithm for a particular system of equations depends on many

factors. However, for given N_u and M_b, the worst-case scenario complexity is presented as $K(2N_u + M_b)$ shift-accumulation (SAC) operations [36].

Lattice Detection: Branch and Bound Detector

The universal lattice decoding problem dates back to at least the early 1990s [6]. The principle of universal lattice decoding can trace its roots back to the theory and algorithms developed for solving the shortest/closest lattice vector problem for integer programming problems. The closest (lattice) vector problem (CVP) (also called the nearest lattice point problem) is a class of nearest neighbor searches or closest-point queries, in which the solution set to be searched consists of all the points in a lattice. Very efficient algorithms for solving the CVP [37] have been derived for the root lattices, which are generated by the root system of certain Lie algebras. These algorithms are important for implementing low-complexity lattice quantizers and coding schemes for Gaussian channels. From a lattice point of view, ML decoding corresponds to solving the closest vector problem in a lattice. However, the optimal multiuser detection is NP-hard (*i.e.* there is hardly to be a polynomial time algorithm), however, there exist some sub-optimal algorithms that can be solvable with polynomial complexity. Branch and bound (BB) is a divide and conquer structure for the hard combinatorial optimization problem [38]. The main idea is to separate the solution set of a discrete optimisation problem into successive smaller subsets (branch), bound the cost function value over each subset and use the bounds to remove some subsets. The process stops when the entire solution set has been totally searched. The best solution of the BB algorithm is a global optimum since it effectively searches the whole solution space. This method is most efficient when it is possible to remove many subsets as early as possible during the branching procedure without really calculating them. Luo *et al.* [39] proposed a detector based on depth-first BB [40] and showed that the sphere decoder is a type of depth-first BB [41]. The BB can be seen as a tree search algorithm where each subset is indicated by a node in a tree, and the root of the tree indicates the whole solution space. Each node is related with a cost that is a lower bound to the global optimum. Accordingly, if a node cost exceeds the current best solution, the node can be pruned, *i.e.* the children of the node can be removed without a loss. The algorithm maintains a list (queue) of nodes to be processed. When a node is remained, its children branches are made and their costs are evaluated. Those children nodes whose cost is less than the current best solution are added to the list. Unfortunately, this algorithm may have to save the whole tree in a worst-case scenario. Its memory

requirements are exponential in terms of the number of levels in the search tree [42].

As a summary, Table 1.2 compares some multiuser detection algorithms. The benefits, complexity and shortcomings of these detectors are emphasized in this table. The optimal maximum likelihood detector unfortunately has an exponential complexity $\mathcal{O}(2^K)$. The traditional matched filter detector comes straight from single user design with a low complexity $\mathcal{O}(K)$. However since it does not take into account any other users in the system, it cannot provide good performance. The decorrelating detector essentially applies the inverse of the correlation matrix of user spreading sequences to the output of the conventional detector. The complexity of decorrelating detector is $\mathcal{O}(K^3)$. This detector does not require the power of the each user estimation or controlling, however the noise is enhanced by $\mathbf{R}^{-1}\mathbf{n}$. The MMSE detector minimizes the squared error in the presence of channel noise and has a better performance than the decorrelator in the low SNRs. However it requires matrix inversion which is complicated in FPGA implementation. The decision feedback detector is one of the most popular methods in multiuser detection, because of its simplicity and outstanding performance in comparison with the linear detectors. However, its performance mainly depends on the detection order. The zero-forcing DF detector requires the Cholesky decomposition and matrix inversion which are hard for implementation. The semidefinite relaxation detector is a complexity-constrained alternative for the exact ML detector, but the complexity is still very high in large systems. The box-constrained detector is corresponding to nonlinear successive and parallel interference cancelation structures. The sphere-constrained maximum likelihood detector is verified to have a close relationship to the MMSE detector. The PDA detector approaches the single user detector performance, but it requires matrix inversion to obtain the covariance of the noise. The DCD detector is multiplication and division free which is efficient for hardware implementation, however its performance is not as good as that of the PDA detector. The branch and bound detector has low average complexity at high SNRs, however its worst-case computational complexity is identical to that of the optimal multiuser detector, *i.e.* it grows exponentially as increasing K.

1.2.4 MIMO Detection

MIMO communication exploits multi-paths, and turns multipath propagation (traditionally a drawback) into an advantage. There are many advanced techniques for the trade-off between the Diversity and Spatial Multiplexing in MIMO systems [43,44]. Adding redun-

Table 1.2: Comparison of multiuser detection algorithms

Name	**Complexity**	**Benefits**	**Shortcomings**
Maximum Likelihood	$\mathcal{O}(2^K)$	Optimal detection performance.	Exponential computational complexity.
Matched filter	$\mathcal{O}(K)$	Simplicity	Poor performance.
Decorrelating detector	$\mathcal{O}(K^3)$	No requirement for the knowledge of power of the interference	Requires matrix inversion.
MMSE	$\mathcal{O}(K^3)$	Better performance than that of decorrelating detector at low SNRs.	Requires matrix inversion, estimation of user's amplitude and noise variance.
Decorrelating Decision Feedback Detector	$\mathcal{O}(K^2)$ (no Cholesky decomposition)	Better performance than that of the linear detectors.	Performance strongly relies on the detection order. Requires the Cholesky decomposition and the matrix inversion.
Semidefinite Relaxation	$\mathcal{O}(K^{3.5})$	Close to ML detector performance.	Complexity is too high for large systems.
Sphere constrained detector	$\mathcal{O}(K^3)$	Close to the MMSE detector	Not optimal performance.
Box-constrained detector	$\mathcal{O}(K^3)$	Similar to the soft interference cancellation; Better performance than that of the sphere-constrained detector	Not optimal performance
Probabilistic Data Association	$\mathcal{O}(K^3)$	Close to the single user performance.	Requires matrix inversion.
Branch and Bound	the worst case complexity is exponentially in K	Low average complexity	Worst-case complexity is too high at low SNRs.
Dichotomous coordinate descent	$K(2N_u + M_b)$	Multiplication and division free; Efficient for FPGA implementation	Not optimal performance.

dancy to the transmitted binary data, and so, the diversity and the transmission quality are increased. The spatial multiplexing divides the input data stream into several sub-streams that are transmitted over different antennas. This architecture can increase the system capacity. In this work, we consider the MIMO communications with spatial multiplexing.

Fig.1.3 shows a MIMO system with M_T transmit and M_R receive antennas. The received signal vector is given by

$$\mathbf{y} = \mathbf{G}\mathbf{h} + \mathbf{n}\,, \tag{1.36}$$

where

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{21} & \cdots & g_{M_R 1} \\ g_{12} & g_{22} & \cdots & g_{M_R 2} \\ \vdots & \vdots & \vdots & \vdots \\ g_{1M_T} & g_{2M_T} & \cdots & g_{M_R M_T} \end{bmatrix} \tag{1.37}$$

is $M_R \times M_T$ channel matrix composed of independent identically distributed (i.i.d.) complex Gaussian random elements with zero mean and unit variance, $\mathbf{h}$ is an $M_T \times 1$ transmitted complex vector whose entries have real and imaginary parts that are integers, $\mathbf{n}$ is the $M_R \times 1$ i.i.d. complex AWGN vector with zero-mean and covariance matrix $\sigma^2\mathbf{I}$.

Assuming $\mathbf{G}$ is known at the receiver, the ML detection is given by [45]

$$\begin{aligned} \hat{\mathbf{h}} &= \arg \min_{\mathbf{h}\in\mathcal{A}^{M_T}} \|\mathbf{y} - \mathbf{G}\mathbf{h}\|^2 \\ &= \arg \min_{\mathbf{h}\in\mathcal{A}^{M_T}} \left(\mathbf{h}^H\mathbf{R}\mathbf{h} - 2\Re\{\boldsymbol{\theta}^H\mathbf{h}\}\right). \end{aligned} \tag{1.38}$$

We denote

$$J(\mathbf{h}) = \mathbf{h}^H\mathbf{R}\mathbf{h} - 2\Re\{\boldsymbol{\theta}^H\mathbf{h}\} \tag{1.39}$$

as the quadratic cost function, $\mathbf{R} = \mathbf{G}^H\mathbf{G}$ and $\boldsymbol{\theta} = \mathbf{G}^H\mathbf{y}$. Since the computational complexity of the ML decoder is $\mathcal{O}(\mathcal{A}^{M_T})$ arithmetic operations (alphabet $\mathcal{A}$ is the order of digital modulation), it becomes too complex for high $\mathcal{A}$ and M_T.

The MIMO system detection for spatial multiplexing is analogous to the multiuser detection as long as we assume that M_T is equivalent to the number of users K in the multiuser detection, M_R is equivalent to the spreading factor. The matrix $\mathbf{G}^H\mathbf{G}$ is considered to be equivalent to the spreading waveform correlation matrix $\mathbf{R}$ in multiuser detection.

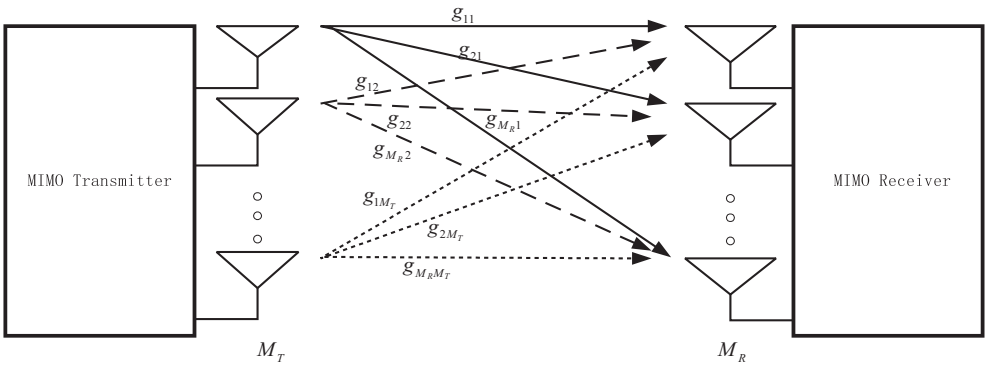


Figure 1.3: MIMO Wireless System

In addition, the complex matrix equation (1.36) can be written in the real matrix representation

$$\begin{bmatrix} \Re(\mathbf{y}) \\ \Im(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \Re(\mathbf{G}) & -\Im(\mathbf{G}) \\ \Im(\mathbf{G}) & \Re(\mathbf{G}) \end{bmatrix} \begin{bmatrix} \Re(\mathbf{h}) \\ \Im(\mathbf{h}) \end{bmatrix} + \begin{bmatrix} \Re(\mathbf{n}) \\ \Im(\mathbf{n}) \end{bmatrix}. \tag{1.40}$$

Solving (1.39) is impractical and exhaustive for high transmission rate, and the complexity grows exponentially [46, 47].

Instead of solving (1.39), the QU decomposition of $\mathbf{G}$, *i.e.* $\mathbf{G} = \mathbf{QU}$, is used to alleviate computations [48], $\mathbf{Q}$ is unitary, and $\mathbf{U}$ is an upper triangular matrix. Left-multiplying (1.36) using $\mathbf{Q}^H$, we can get the modified model as

$$\hat{\mathbf{y}} = \mathbf{Uh} + \mathbf{Q}^H\mathbf{n}\,, \text{ with } \hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}\,. \tag{1.41}$$

Therefore the maximum likelihood detection can be substituted with

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h}\in\mathcal{A}^{M_T}} d(h) = \arg \min_{\mathbf{h}\in\mathcal{A}^{M_T}} \|\hat{\mathbf{y}} - \mathbf{Uh}\|\,. \tag{1.42}$$

We set a partial candidate symbol vector $\mathbf{h}^{(i)} = [h_i\, h_{i+1}\, \cdots\, h_{M_T}]$, $i = 1, 2, \cdots, M_T$, and the $\mathbf{h}^{(i)}$ can be arranged in a tree that has its root on level $i = M_T + 1$ and leaves which correspond to the set of all possible candidate vector symbols at level $i = 1$.

The vector norm in (1.42) can be calculated recursively as $d(h) = d_1$ with the partial Euclidean distances (PEDs)

$$d_i = d_{i+1} + |e_i|^2 \tag{1.43}$$

where $d_{M_T+1} = 0$, start from M_T, and the distance increments

$$|e_i|^2 = |\hat{y}_i - \sum_{j=M_T}^{i+1} U_{ij}h_j - U_{ii}h_i|^2 \tag{1.44}$$

The computation of $d(h)$ can be interpreted as a traversal of the tree from the root to the leaf corresponding to h.When proceeding from a node on level $i+1$ to one of its children on level i, the detector increments the PED by the nonnegative quantity $|e_i|^2$.

The ML solution (1.42), can now be found by an exhaustive tree traversal that searches for the leaf associated with the smallest $d(h)$. Efficient tree-search algorithms reduce the search space by pruning the tree below certain nodes, such that the path leading to the ML solution is preferably not discarded. Pruning criteria are typically based on the PED and resource constraints. There are two algorithms *i.e.* sphere decoding (SD) and K-best, which both have their origins in [49, 50]. With the effect of the searching radius, the SD algorithm has variable throughput and complexity, which is difficult in real-time systems. The K-best detection demonstrates fixed-complexity and fixed-throughput, while the choice of the design parameter K effects a balance between the throughput and BER performance. Computational complexity grows with increasing K, but the BER performance also improves [48].

MIMO OFDM Systems

In broadband wireless systems, the issue of the frequency-selective fading channels becomes an important subject. However, the equalizer is much more complex in MIMO channels because the channel must be equalized over both space and time [51]. The orthogonal frequency-division multiplexing (OFDM) is an attractive approach to reduce the complexity of equalization and decoding [52, 53]. The OFDM divides spectrum into a number of equally spaced sub-carriers and allocates a portion of system information on each subcarrier. The OFDM can be viewed as a form of frequency division multiplexing (FDM) [54]. It allows that the data over narrow-band carriers transmitted in parallel. High bandwidth is achieved by using these parallel sub-channels that are spaced apart at precise frequencies while being as close as possible without overlapping or interfering.

Table 1.3: Comparison of implementations for 4×4 16QAM MIMO detection

Techniques	**BER**	**Hardware platform**	**Clock frequency**	**Throughput at SNR $= 20$ dB(Mbps)**	**FPGA no. slices or gate ASIC**
K-best 1 [1,55]	close ML	ASIC	100MHz	10	52000
K-best 2 [1,56]	close ML	ASIC	100MHz	52	91000
Depth-first SD 1 (ℓ^2-norm) [1,57]	ML	ASIC	51MHz	73	117000
Depth-first SD 2 (ℓ^1-norm) [1,57]	close ML	ASIC	71MHz	169	50000
Depth-first SD 3 [1]	ML	FPGA	50MHz	114.5	21467

1.2.5 Hardware Implementation of MIMO and Multiuser Detectors

Starting from the early work by Foschini, Gans, Teletar, and Paulraj [58–61], many papers have been published in the area of MIMO-based information theory, algorithms, codes, and so on. Much work have been focused on the algorithms and protocols that deliver superior BER for a given SNR. Little attention is given to the real-time implementation of these algorithms. The hardware complexity of the algorithms needs to be concerned so that the MIMO detector can be integrated with the rest of the system. So far the MIMO algorithms are generally implemented in digital signal processors (DSPs), and it is difficult to achieve high data-rate performance. Field-programming gate-array (FPGA) devices are widely used in signal processing, communications, and network applications because of their reconfigurability and support of parallelism.

The FPGA platform has at least three advantages over a DSP processor: the inherent parallelism of the FPGA is equipped for vector processing; it has reduced instruction overhead; the processing capacity is scalable when the FPGA resource is sufficient [62]. The disadvantage is that the development cycle of the FPGA design is usually longer than the DSP implementation. But once an efficient architecture is developed and the parallel implementation is explored, the FPGA design is able to significantly improve the processing speed because of its intrinsic density advantage [63].

In addition, the FPGA platform has several advantages over an application-specific integrated circuit (ASIC) implementation: an FPGA device is reconfigurable to accommodate system configuration changes even in run-time; it has significantly reduced latency comparing to ASIC; it is a cost-effective solution. Furthermore, ASIC implementation is generally applied for a fixed number of antennas and a certain signal constellation. The limitation of an ASIC implementation is lack of flexibility when the number of antennas or the signal constellation changes [62].

The MIMO detectors are generally implemented on DSPs, such as the Bell Labs layered space-time (BLAST) system [64, 65]. Because it does not support parallel computation, the speed of DSP implementation is often limited, especially as the number of antennas increases. Recently many ASIC and FPGA implementations of the SD or close-to-ML detectors were reported in [55–57, 66]. The VLSI design of depth-first detectors using sphere decoding algorithms [67, 68] and breadth-first detectors using the M-algorithm [69] have both attracted many recent attention [55,56,70]. For a 4×4 MIMO transmission

with 16-QAM, hard-output depth-first detectors [57] achieve much higher throughput than their breadth-first counterparts [55,56]. In general, the average computational complexity of depth-first hard-output detection is lower than its breadth-first counterpart due to its ability to adaptively tighten the search radius constraint. For soft-output detection, it is not trivial to adaptively change the search radius for both depth-first search and breadth-first search, while the breadth-first search has the advantage that it can naturally generate an ordered candidates list for *a posteriori probability* (APP) calculation [71].

Table 1.3 gives an overview and comparison of the relevant hardware implementations of 4×4 16-QAM MIMO detection algorithms [70], and most hardware implementations of the MIMO communications so far deal with the small size systems. The depth-first SD1 and SD2 are based on the ASICs implementation. The SD2 implementation has twice the throughput of the SD1 at half chip area. Depth-first tree transversal is implemented in a sequential and non-pipelined, while the K-best algorithm is based on a parallel and pipelined hardware structure with reduced chip area. Besides, the K-best approach is guaranteed constant throughput but with the expense of the performance lose. The average throughput of the depth-first approach could match that of the K-best approach, but the throughput of worst-case of it may drop severely. The FPGA SD3 [57] has a similar performance to the SD1, however its complexity is significantly reduced. It also needs to mention that these real-time implementations of SD in Table 1.3 do not include the channel matrix preprocessing such as QR decomposing, or Cholesky factorization, which consume much more hardware resources.

The multiuser detector implementation is more difficult because they mainly perform for large size systems. Here are some references available *e.g.* the FPGA implementation of an adaptive MMSE algorithm is presented in [72], the FPGA multiuser detector based on a cascade of adaptive filters for asynchronous WCDMA systems is presented in [73].

1.2.6 Summary

From above preliminaries for this work, we believe that the synchronous CDMA system model could be considered as a general model for more complex scenarios *e.g.* frequency selective fading, fast and slow fading channels. In addition, we also showed that solving the MIMO communications is mathematically similar to that of the multiuser detection, and so, the multiuser detection model can also be converted to a MIMO communication

system.

In addition, we know that the traditional matched filter provides poor detection performance. The decorrelating and MMSE detectors improve the detection performance. However, they require matrix inversion which is complicated for real-time implementation *i.e.* hardware implementation. Advanced algorithms *e.g.* semi-definite relaxation, constrained, PDA and BB detectors can provide near or exact ML detection performance, however, the complexity issue is still a challenge for the real-time implementation. The current existing hardware implementation designs only deal with small size systems. Therefore, providing efficient algorithms, which have high throughput while occupying the low cost of the resources, has become our motivation for this work.

1.3 Outline of the Thesis

Rest of the thesis is organized as follows.

Chapter 2 presents a box-constrained DCD algorithm. An efficient FPGA implementation of the box-constrained DCD algorithm based multiuser detector is proposed. Numerical results show that the box-constrained DCD detector can be implemented for large size system with small FPGA slices account, and guarantees outstanding detection performance.

Chapter 3 presents an FPGA design of a box-constrained DCD MIMO detector. This design requires significantly lower area usage than known designs of the MMSE MIMO detector. In addition, this detector can achieve significantly better detection performance than the MMSE detector, especially for large MIMO systems.

Chapter 4 presents the DCD-BTN multiuser detector. The fixed-point DCD-BTN detector has a very close detection performance to the floating-point implementation, even for a large system size. Compared with the box-constrained DCD multiuser detector, the DCD-BTN multiuser detector significantly improves the detection performance only with small increase in the number of slices.

Chapter 5 presents the DCD-BTN-M detector for M-PSK symbols detection. This detector is based on the DCD-BTN algorithm with some modifications. The numerical

results show that the DCD-BTN-M detector offers the detection performance close to the single user bound.

Chapter 6 proposes a multiple phase decoder for M-PSK symbols. The proposed detector provides a solution to the quadratic optimization problem with the constraint that forces elements of the solution to have unit magnitudes. In highly loaded multiuser detection scenarios, the proposed detector has a better performance and a lower complexity than the semi-definite relaxation detector. In the MIMO communication, the proposed detector offers a very similar performance to the sphere decoder with QPSK modulation. The complexity of the proposed detector grows linearly in the system size, the complexity of the sphere decoder is predicted as exponential at the low SNRs. Therefore the proposed detector can be considered as a promising alternative to the sphere decoder for ML decoding in MIMO detection.

Chapter 7 presents a combined DCD-BB detector based on the box-constrained DCD algorithm and the BB algorithm. The BB detector provides the optimal detection performance. However, the worst-case computational complexity of the BB detector is prohibitive, which makes the BB algorithm difficult for real-time applications. While the box-constrained DCD algorithm has a low computational complexity at any SNR because of its property of free multiplication/division operations, its detection performance is inferior to the BB detector. A combination of the BB detector and the box-constrained DCD detector can be used in a highly loaded scenario, the complexity of the DCD-BB detector is reduced significantly with only a small loss in detection performance with respect to the ML detector.

Chapter 8 proposes an approach based on the DCD algorithm to simplify the inversion operations. The idea of the approach is that the DCD algorithm obtains separately the individual columns of the inverse of the matrix. Due to the low complexity of hardware implementation of the individual DCD algorithm, a block of DCD processors can be used to obtain the columns of the inverse of the channel correlation matrix in parallel.

Chapter 9 concludes the thesis.

In this thesis, the results obtained from FPGA platform are fixed-point; others obtained from MATLAB are floating-point. When the two kind of results are shown in the simulation

figures simultaneously, we will use the name of fixed-point and floating-point to differ them. For other performance figures, the results are all from MATLAB unless we specify they are from FPGA platform.

1.4 Notations

Throughout this thesis, matrices and vectors are denoted by uppercase letters and lowercase letters unless otherwise noted. Vectors and matrices are in bold font. $\mathbf{A}_{M\times N}$ is M-row N-column matrix. $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. $\mathbf{A}^H$ is the Hermitian transpose of $\mathbf{A}$. An element of the matrix is denoted as $A_{p,n}$. $\mathbf{I}$ is an identity matrix. In general, notations are introduced at first occurrence.

1.5 Contributions

The goal of our work is to improve the performance of the detection methods of multiuser and MIMO communication, to develop efficient detection algorithms which give high performance with a relatively low computational complexity, which is applicable for FPGA implementation. Throughout the thesis, we have made the following major contributions.

1. Designed the FPGA architectures of the box-constrained DCD algorithm, which can be applied to the detection in the multiuser and MIMO communications. These designs enable a good trade-off between FPGA resources and the transmission throughput. The error rate performance of the FPGA fixed-point implementation of the box-constrained DCD based detector is close to the floating-point solution.

2. Developed the FPGA architecture for the optimal DCD-BTN algorithm. The FPGA DCD-BTN detector shows a good BER performance close to its floating implementation solution and to the optimal ML performance.

3. A multiple phase decoder (MPD) which is based on the phase descent search algorithm for M-PSK modulation is proposed. When the MPD is applied to multiuser detection, it provides an error probablity performance close to the single-user

boundary. In the MIMO communication, the MPD offers a performance which is similar to that of the sphere decoder in QPSK modulation.

4. Designed a matrix inversion approach based on the DCD algorithm. This approach obtains separately the individual columns of the inverse of the matrix and costs a very small number of slices, which is suitable for a large size matrix inversion.

5. Proposed a detector based on combining the box-constrained DCD algorithm and the sphere decoder (branch and bound algorithm). The efficient combined detector provides a better detection performance than the DCD box constrained detector, and also has a lower worst-case complexity than the sphere decoder, especially in low value of SNRs.

1.6 Publication List

Some of the research presented in this thesis has been published, submitted, or will be submitted for publication at the time of the submission of this thesis.

1. Y. Zakharov, Z. Quan, "Multiuser detector based on multiple phase detection of M-PSK symbols" submitted to *IET Electronics Letter*.

2. Y. Zakharov, Z. Quan, "FPGA Design and Implementation of Efficient multi-user detection based on box-constrained relaxation, regularization with negative diagonal loading, and coordinate-descent iterations", to be submitted to *ICAST09.*

3. Z. Quan, J. Liu and Y. Zakarov, "FPGA design of box-constrained MIMO detector" in *Proc. of International Conference On Communications 2009 (ICC'09 SPC).*

4. Z. Quan, Y. Zakharov, Junruo Zhang , "Multiple phase decoder for MIMO systems", in *Proc. of The 42nd Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, U.S.* Oct. 26-29, 2008.

5. J. Liu, Z. Quan, Y. Zakharov, "Parallel FPGA implementation of DCD algorithm", in *Proc. of the 15th International Conference on Digital Signal Processing (DSP2007)*, Cardiff, Wales, July 1-4 2007.

6. Z. Quan, J. Liu, Y. Zakharov, "FPGA Implementation of DCD Based CDMA Multi-user Detector" in *Proc. of the 15th International Conference on Digital Signal Processing (DSP2007)*, Cardiff, Wales, July 1-4 2007.

7. Z. Quan and Y. Zakharov, "Combined multi-user detection with improved complexity-detection performance", in *Proc. of the Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET)*, Liverpool, UK, June 2006.

Chapter 2

Box-Constrained DCD-based Multiuser Detector

Contents

2.1 Introduction

In CDMA systems, multiuser detection is capable of providing a high detection performance [9]. The use of multiuser detection significantly increases the spectral efficiency of wireless communication systems. This technology has now been developed into an important field in multi-access communications. The conventional detector (matched filter), performs poorly in situations where the energies of the multi-user signals are different or there are many users. The exponential computational complexity of the optimal maximum-likelihood (ML) detector makes it infeasible in the real-time operations. The

sphere decoding algorithm has been proved to simplify the ML multiuser detection [1]. However, the matrix factorizations, such as Cholesky decomposition or QR decomposition, are difficult for hardware implementation [62, 74]. Therefore, the sphere decoding algorithm is suitable only for small size systems.

The box-constrained dichotomous coordinate descent (DCD) algorithm allows achieving a multiplication and division free solution of the normal equations, which makes it more suitable for real-time implementation. In this chapter, the box-constrained DCD algorithm [75] will be studied. The box-constrained DCD-based multiuser detector is proposed for solving large size systems. The performance of the box-constrained DCD based multiuser detector applied in systems with a large number of users will be presented. The box-constrained DCD algorithm is implemented on an FPGA board. Two FPGA architecture designs will be described in this chapter. The serial implementation of the box-constrained DCD algorithm has much less complexity than that of the FPGA implementation of the sphere decoders because the algorithm is free of explicit multiplications and divisions and only requires addition and bit-shift operations. The parallel architecture design can improve the data throughput compared to the serial-based design. Therefore, a parallel FPGA design of the box-constrained DCD detector is also proposed.

The rest of this chapter is organized as follows. Section 2.2 presents the multiuser detection system model. Section 2.3 describes the box-constrained DCD algorithm. Section 2.4 shows the FPGA serial architecture of the box-constrained DCD algorithm, and the corresponding numerical results. Section 2.5 presents the FPGA parallel architecture design for the box-constrained DCD algorithm. Section 2.6 concludes this chapter.

2.2 Formulation of Multiuser Detection Problem

We consider a K-user synchronous CDMA system using BPSK modulation in an AWGN channel. The matched filter output in the receiver is given by:

$$\boldsymbol{\theta} = \mathbf{R}\mathbf{h} + \mathbf{n}\,, \tag{2.1}$$

where the vector $\mathbf{h} \in \{-1, +1\}^K$ contains the bits transmitted by K users, $\mathbf{R}$ is a real-valued $K \times K$ matrix, $\mathbf{h}$ and $\boldsymbol{\theta}$ are real-valued $K \times 1$ vectors and $\mathbf{n}$ is a real-valued zero-mean Gaussian random vector with the covariance matrix $\sigma^2\mathbf{R}$. The optimal ML multiuser detector estimates the vector $\mathbf{h}$ by minimizing the following quadratic function

with integer constraint:

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in \{-1,+1\}^K} \left\{ \frac{1}{2}\mathbf{h}^T \mathbf{R}\mathbf{h} - \boldsymbol{\theta}^T \mathbf{h} \right\}. \tag{2.2}$$

Although the ML detector provides the best detection performance, it is not practical due to its high complexity. The sphere decoder can provide a performance identical to that of the ML detector with significantly reduced average complexity. However, at low SNRs, the worst-case complexity of the sphere decoder is exponentially proportional to the number of users, which prevents the use of the sphere decoder in systems with a large number of users [76].

2.3 Box-constrained DCD Algorithm

Table 2.1: Box-constrained DCD algorithm

Initialization: $\mathbf{h} = \bar{\mathbf{h}}$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, $H = 1, p = 0$.
for $m = 1 : M_b$
1. $d = 2^{-m+1}$
2. Flag $= 0$ $\cdots pass$
3. for $j = 1 : K$ $\cdots iteration$
4. if $|r(j)| > (d/2)R(j,j)$ then
5. $x = h(j) + \text{sign}(r(j))d$
6. if $|x| \leq H$ then
7. $h(j) = x$
8. $\mathbf{r} = \mathbf{r} - \text{sign}(r(j)) \cdot d \cdot \mathbf{R}(:, j)$
9. Flag $= 1, p = p + 1$
10. if $p > N_u$ end algorithm
11. end j-loop
12. if Flag=1, go to 2

end m-loop

The box-constrained DCD-based multiuser detector [75] uses the box-constraint $\mathbf{h} \in [-1, +1]^K$ in the quadratic minimization (2.2). Table 2.1 presents the box-constrained DCD algorithm. The box-constrained DCD algorithm is designed to offer a simple solution for the vector $\mathbf{h}$, without explicit multiplications and divisions. The final accuracy of the solution vector $\mathbf{h}$ depends on the number of bits (M_b), the number of iterations, and the

conditional number of the system matrix, ect. The first set of iterations in the algorithm determines the most significant bit $(m = 1)$ for all elements of $\mathbf{h}$ using a step size parameter d. The subsequent sets of iterations determine the less significant bits up to a suitable number of bits (maximally M_b). The residual vector is given by $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, where $\bar{\mathbf{h}}$ is the initialization of $\mathbf{h}$. In this chapter, $\bar{\mathbf{h}}$ is set to zero, and $\mathbf{r}$ is equal to $\boldsymbol{\theta}$. The step-size d is reduced by power of two at step 1, and so, no explicit multiplication or division is carried out as all the multiplications and divisions can be replaced by simple bit shifts. If an element of the solution vector is updated at an iteration, such an iteration is labeled "successful". For every step size update, the algorithm repeats successful iterations until all elements of the residual vector $\mathbf{r}$ become so small that the condition at step 4 is not met for all j or $\mathbf{h}$ overflows the range $[-H, +H]$ at step 6, where $H = 1$ for BPSK modulation. The computational load of the algorithm mainly depends on these successful iterations N_u and the number of bits M_b. A limit for the number of successful iterations N_u can be predefined and used as a stopping condition (at step 10). If there is no such limit, or the limit is high enough, the accuracy of the solution is 2^{-M_b+1}. In our work, wherever the box-constrained DCD algorithm is used, we denote it as DCD-B$(\bar{\mathbf{h}}, \mathbf{R}, \boldsymbol{\theta}, H, N_u, M_b)$.

A successful iteration requires one addition for comparison (at step 4), and $(K + 1)$ additions for updating the residual vector $\mathbf{r}$ and the element $h(j)$. For an unsuccessful iteration, only one addition is used for the comparison. The worst-case complexity corresponds to an unlikely situation, which occurs when only the last mth bit has N_u successful iterations. This means that the calculation of the first $(M_b - 1)$ bits do not contain any successful iteration, and so, require $(M_b - 1)K$ additions. The worst-case complexity for calculating the last bit (corresponds to $m = M_b$) occurs when only one successful iteration happens among the K iterations $(j = 1, \cdots, K)$. This requires K additions for the comparison and $(K + 1)$ additions to update the residual vector $\mathbf{r}$ (at step 8) and element $h(j)$ (at step 5). In total, N_u successful iterations require $N_u(2K + 1)$ additions.

Therefore, the complexity of the box-constrained DCD algorithm is upper bounded by $K(2N_u + M_b - 1) + N_u$ additions. However in a typical situation, there should be several successful iterations in each pass, and the average complexity will be close to $2KN_u$.

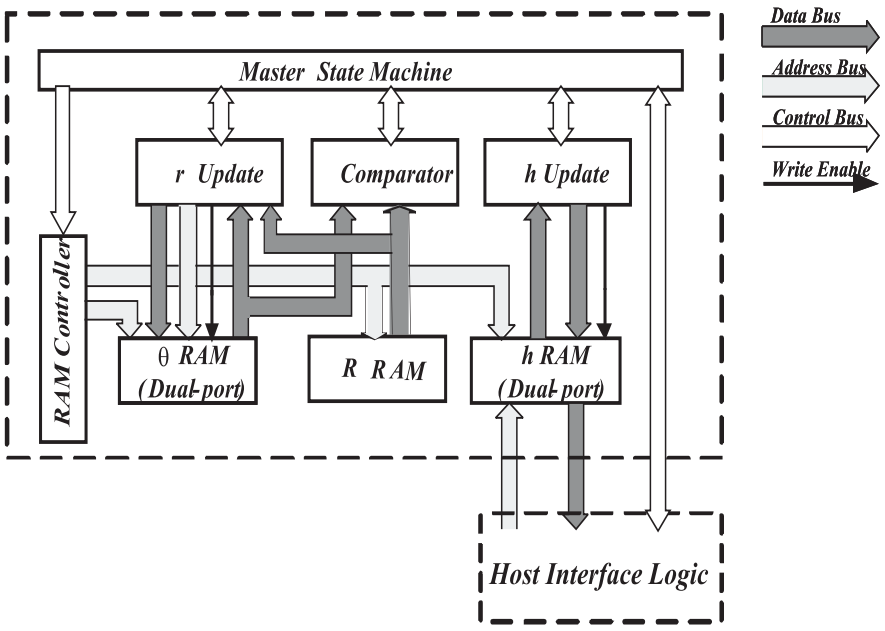


Figure 2.1: DCD Processor Block Diagram

2.4 Fixed-point Serial Implementation of the Box-constrained DCD Algorithm

The fixed-point box-constrained DCD algorithm is implemented directly in VHDL on an FPGA platform. The development board is Xilinx Virtex-II Pro Development System [77] with an FPGA chip XC2VP30 (FFT896 package, speed grade 7). The fixed-point DCD algorithm is synthesized and downloaded to this FPGA chip through the Xilinx ISE 8.1i running at the clock frequency 100MHz. The design uses 16-bit Q15 number format to represent elements of the matrix $\mathbf{R}$. To avoid overflow errors, 32-bit fixed-point integers are used for representation of elements in vectors $\boldsymbol{\theta}$ and $\mathbf{h}$. These elements are limited to the range $[-2^{16}, 2^{16})$. We treat the vector stored in $\boldsymbol{\theta}$ RAM as the residual vector $\mathbf{r}$.

Table 2.2 describes the real-valued serial implementation of the fixed-point box-constrained DCD algorithm. There are six states of the fixed-point box-constrained DCD algorithm as shown in Table 2.2. Fig 2.1 shows the block-diagram of the DCD processor. The transition among these states in the fixed-point box-constrained DCD algorithm is presented in Fig.2.2. The vectors $\mathbf{h}$ and $\mathbf{r}$ are stored in RAM $\mathbf{h}$ and RAM $\boldsymbol{\theta}$, respectively. Vectors $\mathbf{h}$, $\mathbf{r}$, bit counter m, successful iteration counter p, pre-scaling counter Δm, element index j and Flag are initialized in state 0. In state 1, if $m \neq 0$, the algorithm chooses the step-size d to be equal to $d = 2^m$, decrements the bit counter m by one and increments

Table 2.2: Fixed-point real-valued box-constrained DCD algorithm

State	Operation	Cycles
0	Initialization: $\mathbf{h} = \mathbf{0}$, $\mathbf{r} = \boldsymbol{\theta}$, $m = M_b$, $p = 0$, $\Delta m = 0$, $j = 1$, Flag$= 0$	
1	if $m = 0$, algorithm stops else, $m = m - 1$, $d = 2^m$, $\Delta m = \Delta m + 1$	1
2	$c = R(j,j)/2 - \lvert r(j)\rvert \times 2^{\Delta m}$ $h_t = h(j) + \text{sign}(r(j)) \cdot d$ if $\lvert h_t\rvert \leq H$, then $\alpha = 0$; else, $\alpha = 1$	1
3	if $c < 0$ and $\alpha = 0$, then goto state 4 else, goto state 5	1
4	$h(j) = h_t$ $\mathbf{r} = \mathbf{r} \times 2^{\Delta m} - \text{sign}(r(j)) \cdot \mathbf{R}(:, j)$ $\Delta m = 0$, $p = p + 1$, Flag $= 1$ if $p = N_u$, algorithm stops	K
5	$j = (j) \text{mod}(K) + 1$ if $j = 1$ and Flag = 1, then Flag = 0, goto state 2 elseif $j = 1$ and Flag $= 0$, then goto state 1 else, goto state 2	1
Total	$\leq 4KN_u + 3K(M_b - 1) + M_b$	

the prescaling factor $\triangle m$ by one. If the least significant bit of the solution is achieved ($m = 0$), the algorithm stops.

In state 2, two cycles are needed before each comparison because of a latency delay when reading from the RAM. The RAM Controller asserts the addresses of $r(j)$, $R(j,j)$ and $h(j)$, which are stored in $\boldsymbol{\theta}$ RAM, $\mathbf{R}$ RAM and $\mathbf{h}$ RAM, respectively. At the same time, $r(j)$ is scaled by $2^{\triangle m}$ by bit-shifting. In addition, the $\mathbf{h}$ Update Logic reads $h(j)$ from $\mathbf{h}$ RAM, pre-updates $h(j)$ and keeps the updated element in a register h_t. It also checks whether h_t is in the range $[-H, H]$ ($\alpha = 0$) or not ($\alpha = 1$) before proceeding to the state 3.

In state 3, the Master State Machine checks the result of the comparison from the state 2 to decide whether the iteration is "successful". If the iteration is "successful", the algorithm goes to state 4 to update $\mathbf{h}$ and $\mathbf{r}$; otherwise, it proceeds to the state 5.

In state 4, the $\mathbf{r}$-Update Logic updates all elements of the vector $\mathbf{r}$. The RAM Controller indicates addresses of elements of the column $\mathbf{R}(:, j)$ and vector $\mathbf{r}$. The corresponding elements of $\mathbf{r}$ and $\mathbf{R}(:, j)$ are loaded to the $\mathbf{r}$-Update Logic. $h(j)$ is updated by directly

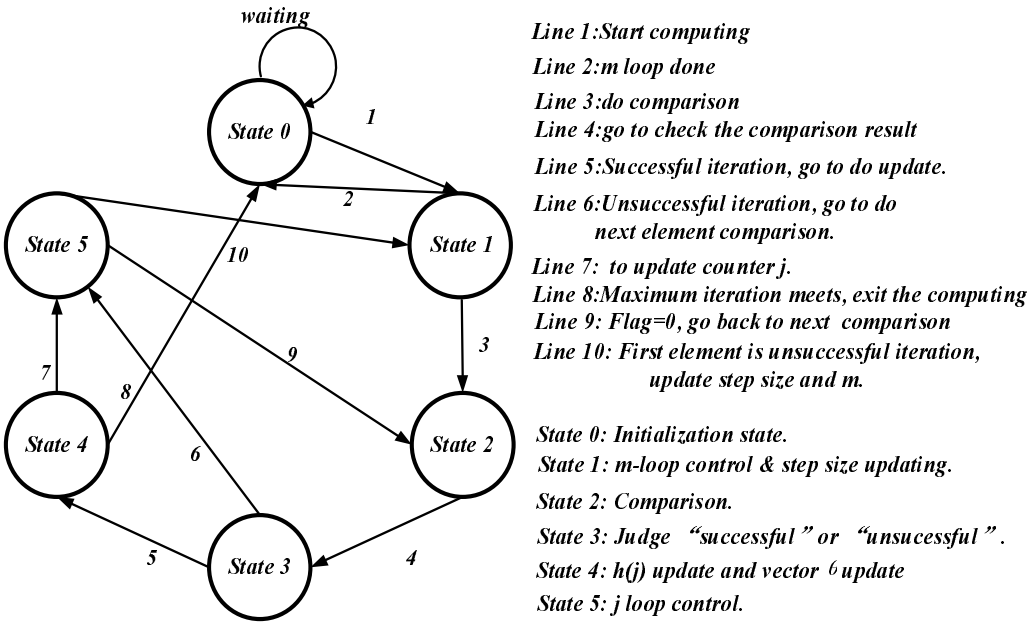


Figure 2.2: DCD Master State Machine

copying the value from the register h_t. Meanwhile, $\triangle m$ is cleared to 0; the iteration counter p is incremented by one and the binary variable Flag is set to 1 which indicates that the current iteration is "successful". The "successful" iteration counter p needs to be checked whether it reaches the maximum number N_u. If p is less than N_u, the Master State Machine proceeds to state 5, otherwise, it stops.

State 5 firstly updates index j. Then, it decides whether to update the Flag and which next state to proceed depending on j and Flag.

The number of clock cycles required for each state is shown in Table 2.2. By using the pipeline technology, state 4 requires K cycles for updating all elements in the vector $\mathbf{r}$ and the element in $\mathbf{h}$. Other states only require one cycle to execute. State 2 also needs 2 clock cycles more because there is a latency when reading data from the RAM. The total cycles required mainly depend on these successful update iterations and the number of bits. The upper number of cycles in Table 2.2 can be considered as the worst case complexity of the fixed-point box-constrained DCD algorithm. We can consider the worst case situation, which occurs when only the last mth bit has N_u successful iteration. This means that the calculation of the first $(M_b - 1)$ bits do not contain any successful iteration, and so, require $(M_b - 1)3K$ cycles. The worst-case for calculating the last bit $(m = 1)$ occurs when only one successful iteration happens among the K iterations $(j = 1, \cdots, K)$. This requires $3K$ cycles for the comparison and K cycles for update of the residual vector $\mathbf{r}$

and element $h(j)$. In total, N_u successful passes require $N_u(4K)$ cycles. In addition to the above cycles needed, there also requires M_b cycles for the step size update in the whole process.

Therefore, the maximum clock cycles of the fixed-point box-constrained DCD algorithm is $4KN_u + 3K(M_b - 1) + M_b$ clock cycles.

Fig.2.1 presents the FPGA architecture of the box-constrained DCD algorithm. There are five sub-modules: Master State Machine, RAM Controller, Comparator, $\mathbf{r}$ Update Logic, and $\mathbf{h}$ Update Logic [78]. The matrix $\mathbf{R}$, vector $\mathbf{r}$, and vector $\mathbf{h}$ are saved in $\mathbf{R}$ RAM, $\boldsymbol{\theta}$ RAM and $\mathbf{h}$ RAM, respectively.

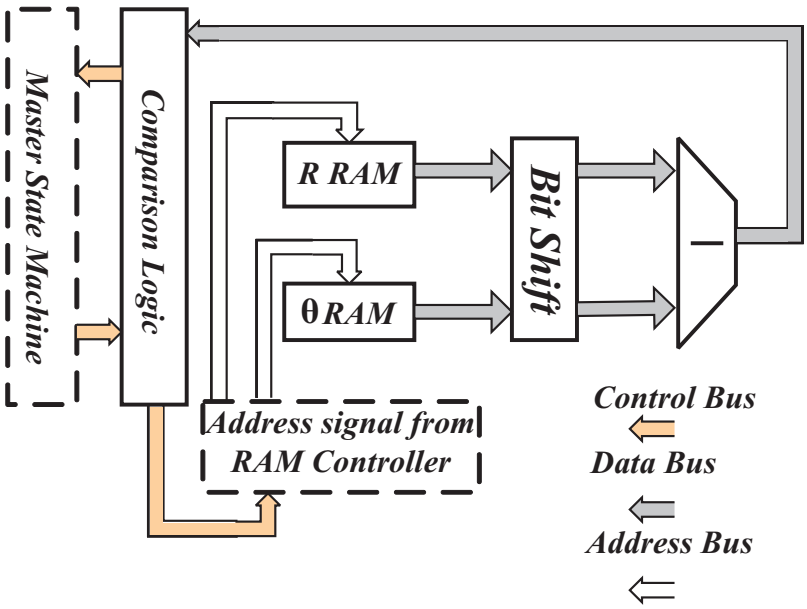


Figure 2.3: Comparator architecture.

Master State Machine: The Master State Machine is required to track the current iteration, select and update the step size and decide which state to execute.

RAM Controller: The RAM Controller drives one of the address ports of $\boldsymbol{\theta}$ RAM and $\mathbf{R}$ RAM, and both address ports of $\mathbf{h}$ RAM. During the "Comparison" in an iteration, the RAM Controller provides the address of $R(j, j)$, $r(j)$, and $h(j)$. The update operation is conditional; if the iteration is "successful", then the RAM controller will sequentially increment addresses in the vector $\mathbf{r}$ and matrix $\mathbf{R}$ so that all K elements in the vector $\mathbf{r}$ are updated. Meanwhile, the RAM Controller indicates the address of $h(j)$ in the $\mathbf{h}$ RAM

for element $h(j)$ update. The RAM Controller then indicates the addresses $r(j+1)$ and $R(j+1, j+1)$ for the next comparison. However, if the iteration is "unsuccessful", the RAM controller immediately indicates the address $r(j+1)$ and $R(j+1, j+1)$ (without update) for the next comparison.

Comparator: Fig.2.3 shows the Comparator architecture. It is used for comparing $r(j)$ and $(d/2)R(j, j)$ at step 4 in Table 2.1. The comparator also scales $r(j)$ to compensate for d at state 2 in Table2.2 and passes the sign bit of the result c to the Master State Machine.

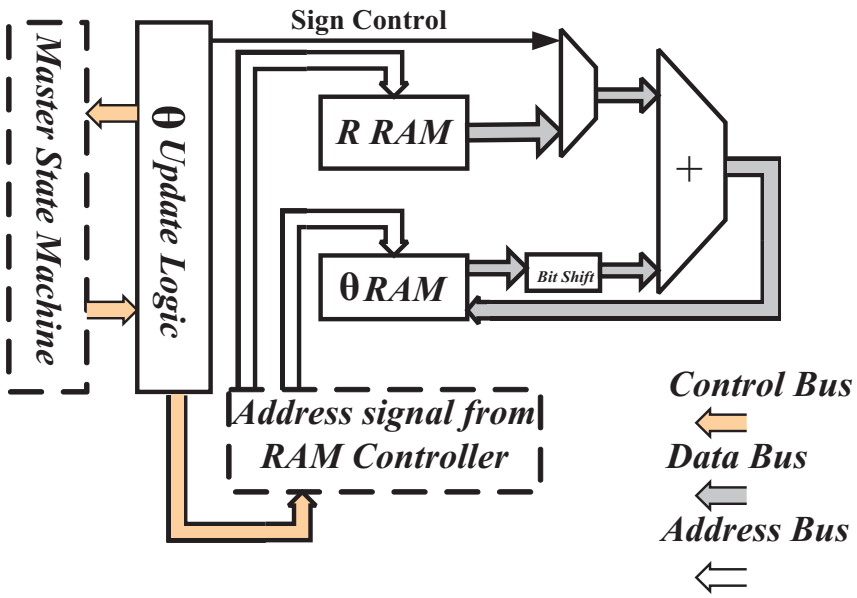


Figure 2.4: Architecture of DCD **r**-Update logic.

r Update Logic: Fig.2.4 presents the architecture of the **r** Update Logic. The initialization vector **r** and **h** are stored concurrently in $\boldsymbol{\theta}$ RAM and **h** RAM, respectively. The $\boldsymbol{\theta}$ RAM is filled with scaled data, whilst the **h** RAM is cleared to zeros. Each element is read from $\boldsymbol{\theta}$ RAM, updated as $\mathbf{r} = \mathbf{r} \times 2^{\Delta m} - \text{sign}(r(j)) \cdot \mathbf{R}(:, j)$, and written back to $\boldsymbol{\theta}$ RAM.

h Update Logic: Fig.2.5 shows the **h** Update Logic. The **h** Update Logic reads the element $h(j)$ from **h** RAM. $h(j)$ is updated by adding or subtracting the step size d according to the sign of $r(j)$. The updated $h(j)$ is written back in the **h** RAM.

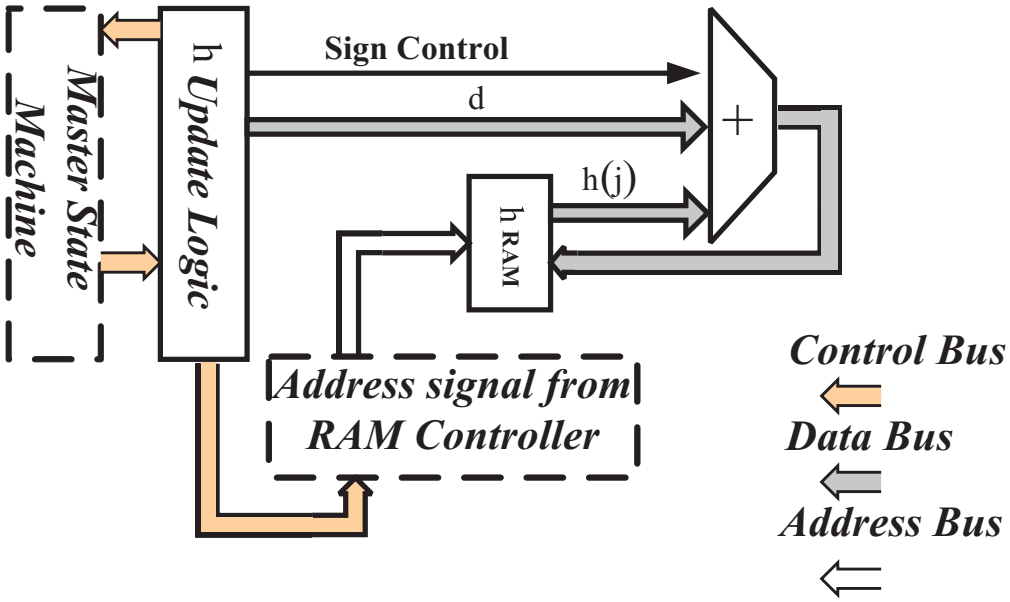


Figure 2.5: Architecture of the DCD **h**-Update Logic.

2.4.1 Detection Performance of the DCD-based Box-constrained Detector

The BER performance of the box-constrained DCD detector is evaluated using 10^5 simulation trials. We assumed the simulation scenario is AWGN channel with perfect power control, where the users employ randomly generated spreading sequences. We consider a high loaded scenario, for which the number of users $K = 50$ and the spreading factor SF $= 53$, and a less loaded scenario, for which $K = 20$ and SF $= 31$. We also investigate the detector using both the fixed-point and floating-point implementations for these scenarios. The floating-point implementation results are obtained from MATLAB, and the fixed point implementation results are obtained from FPGA board.

Fig.2.6 shows the BER performance as a function of SNR for a high loaded scenario and different M_b. It is seen that there is no distinguishable difference between the fixed-point FPGA results and the floating-point results. The box-constrained DCD detector even with two-bit representation ($M_b = 2$), still outperforms the MMSE detector over the SNR range from 0 dB to 17.5 dB. Furthermore, when BER $= 10^{-3}$, the box-constrained DCD detector with $M_b = 4$ demonstrates a 7.5 dB improvement of the BER performance in comparison with the MMSE detector. When $M_b = 14$, the BER performance is not much improved in comparison with the case $M_b = 4$. It shows that a further increase in the number of bits M_b will not achieve much improvement of the BER performance.

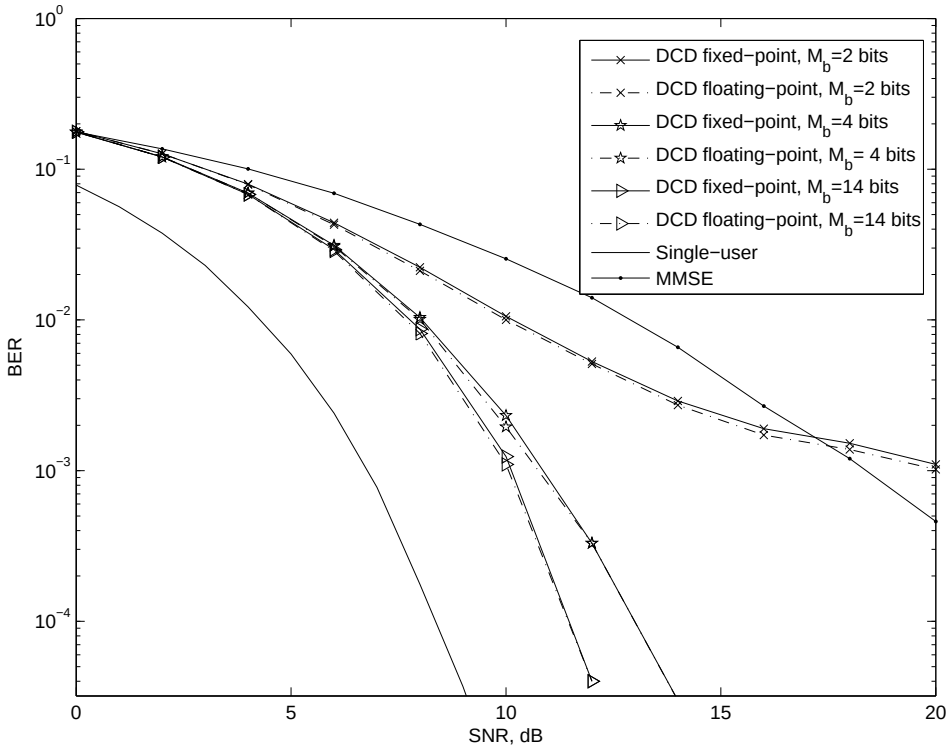


Figure 2.6: BER performance of the DCD based box-constrained multiuser detector for different number of bits M_b in a highly loaded multiuser scenario; $K = 50$, SF $= 53$.

Fig.2.7 shows the BER performance for a less loaded scenario. The parameter M_b in the box-constrained DCD algorithm can be chosen any integer to represent the estimate data. In this simulation, we use different M_b *i.e.* $2, 4, 14$ to show the detection performance. It shows that the BER performance of the FPGA fixed-point implementation of the box-constrained DCD detector is close to the floating-point solution. The box-constrained DCD detector even with two-bit representation ($M_b = 2$), outperforms the MMSE detector over the SNR range from 0 dB to 20 dB. Moreover, Fig.2.7 also shows that the detection performance of the box-constrained DCD detector with $M_b = 4$ is almost similar to that with $M_b = 14$. Therefore, only $M_b = 4$ is enough for implementation of the box-constrained DCD detector.

Table 2.3: FPGA resources needed for the sphere decoding algorithm [1] $K = 4$;

FPGA Resources	**Total available**	**Usage**
Slices	33088	12721
Multipliers	328	160
Block RAMs	328	82

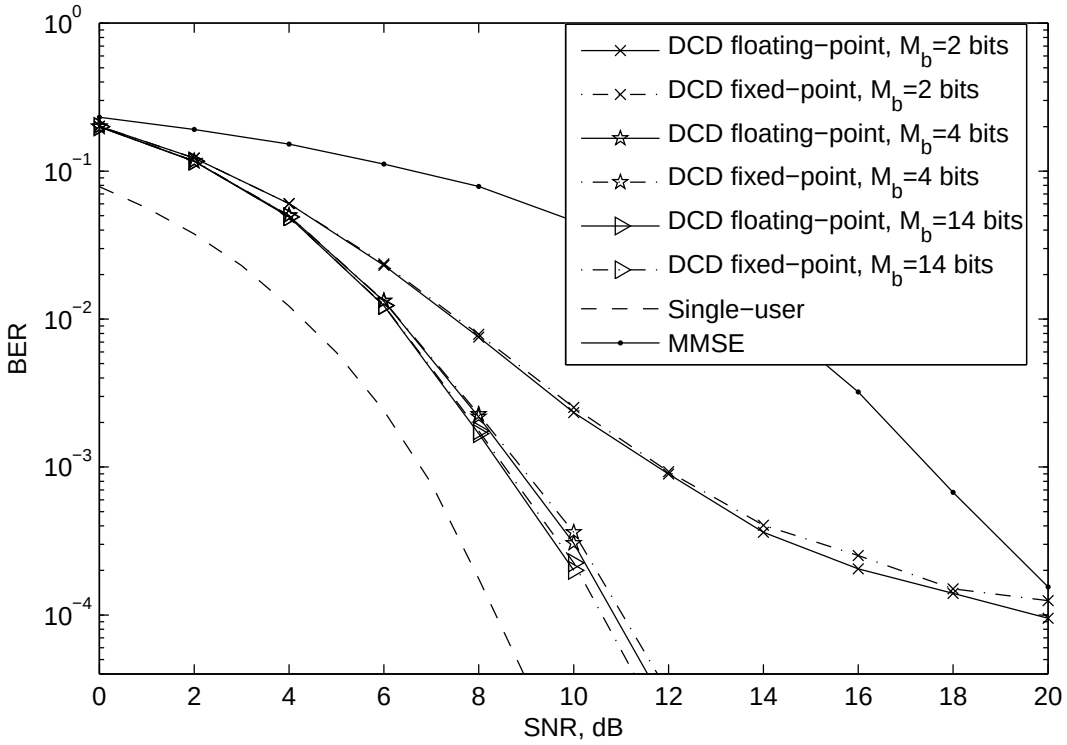


Figure 2.7: BER performance of the box-constrained DCD detector with different M_b; $K = 20$, SF $= 31$.

Table 2.4: FPGA resources needed for the DCD box-constrained algorithm $K = 50$, $M_b = 4$.

FPGA Resources	Total available	Usage
Slices	13696	387
Multipliers	136	0
Block RAMs	136	4

2.4.2 FPGA Resources and Maximum Clock Cycles Required for the Serial DCD-based Box-constrained Detector

Table 2.3 summarizes the resources needed for the FPGA implementation of sphere decoding algorithm for $K = 4$ [1]. This does not take into account the computational complexity of the pseudoinverse, and the Cholesky decomposition which are very complex for hardware implementation. It is also difficult to provide results for higher K because of the infeasibility to implement this algorithm for more users. In comparison, Table 2.4 summarizes the FPGA resources needed for the DCD box-constrained algorithm for $K = 50$ and $M_b = 4$. This algorithm uses a much larger number of users than that used for the sphere decoding algorithm, and so, it is not an equivalent comparison; however, it is still

Table 2.5: Worst-case number of clock cycles required for the box-constrained DCD algorithm for different M_b.

M_b \ K	**4**	**20**	**50**	**110**
2	$16N_u$+14	$80N_u$+62	$200N_u$+152	$440N_u$+332
4	$16N_u$+40	$80N_u$+184	$200N_u$+454	$440N_u$+ 994
14	$16N_u$+170	$80N_u$+794	$200N_u$+1964	$440N_u$+ 4304

a worth comparison since the sphere detector can provide optimal detection performance. The number of multipliers used in the box-constrained DCD algorithm implementation is zero, because multiplication operations are achieved by simple bit-shift operations. The number of slices used in the sphere decoding algorithm implementation is approximately 33 times greater than that used in the box-constrained DCD algorithm. Furthermore, the number of RAMs used in the box-constrained DCD detector implementation is approximately 20 times less than that used in the sphere decoder. We have also implemented box-constrained DCD detector when $K = 4$ (not show in this thesis). We found that that the number of slices used in the box-constrained DCD algorithm does not vary significantly when changing the system size, *i.e.* K from 4 to 50, however the RAM storage for the equation operands will be increased when the system size increases.

Table 2.5 shows the maximum number of clock cycles of the box-constrained DCD algorithm for different K and different M_b. For the same K, the maximum number of clock cycles required increases as M_b increases. For the same M_b, the maximum number of clock cycles needed increases as K increases. In conclusion, the update time increases as either M_b or K increases.

2.5 Fixed-point Parallel Implementation of the Box-constrained DCD Algorithm

We have shown that the serial implementation of the box-constrained DCD algorithm requires very few hardware resources even for the large number of users. However, due to the sequential processing, this architecture might not provide satisfactory data throughput. In section 2.4, we considered that elements of the vector $\mathbf{r}$ are stored in $\boldsymbol{\theta}$ RAM and are updated sequentially, one-by-one. In this section, we consider a change in the way of

Table 2.6: Parallel Implementation of Box-constrained DCD Algorithm

State	Operation	Cycles
0	Initialization: $\mathbf{h} = \mathbf{0}$, $\mathbf{r} = \boldsymbol{\theta}$, $m = M_b$, $p = 0$, $j = 1$, Flag $= 0$	
1	if $m = 0$, algorithm stops else, $m = m - 1$, $d = 2^m$, $\mathbf{r} = 2\mathbf{r}$	1
2	$\mathbf{r}_t = \mathbf{r} - \text{sign}(r(j))\mathbf{R}(:, j)$	1
3	$c = R(j, j)/2 - \lvert r(j)\rvert$ $h_t = h(j) + \text{sign}(r(j)) \cdot d$ if $\lvert h_t\rvert \leq H$, then $\alpha = 0$; else, $\alpha = 1$	1
4	if $c < 0$ and $\alpha = 0$ $h(j) = h_t$ $\mathbf{r} = \mathbf{r}_t$ $p = p + 1$, Flag $= 1$ if $p = N_u$, algorithm stops $j = (j)\text{mod}(K) + 1$ if $j = 1$ and Flag $= 1$, then Flag $= 0$, goto state 2 elseif $j = 1$ and Flag $= 0$, then goto state 1 else, goto state 2	1
Total:	$\leq 3KN_u + 3K(M_b - 1) + M_b$	

storing elements of the residual vector $\mathbf{r}$. These elements are now stored in registers, which allows all K elements of $\mathbf{r}$ to be updated in one clock cycle for each "successful" iteration. In addition, all elements in the column $\mathbf{R}(:, j)$ are accessible simultaneously, which can be done in two ways. One way is that when the elements of the column $\mathbf{R}(:, j)$ are stored in registers, it is called the *R-in-Register*. Another one is, when the elements of the column $\mathbf{R}(:, j)$ are stored in block-RAMs, it is called *R-in-RAM* [79].

The box-constrained DCD algorithm in parallel architecture is presented in Table 2.6. In state 0, the control signals are initialized. Elements of the vector $\mathbf{r}$ are stored into the registers and elements of the matrix $\mathbf{R}$ are stored into registers or RAMs. In state 1, if $m \neq 0$, the operations of left-shifting of the elements of $\mathbf{r}$ are executed simultaneously in one cycle. Meanwhile, the step-size d is updated and the bit counter m is decremented by one in the same clock cycle. In state 2, the Master State Machine passes the elements of $\mathbf{R}(:, j)$, and vector $\mathbf{r}$ to the $\mathbf{r}$ Update Logic. The time needed here for accessing the elements and computing $\mathbf{r}_t$ is about two clock cycles. In state 3, the Master State Machine compares $R(j, j)$ (right-shifted) and $r(j)$. In addition, the $\mathbf{h}$ Update Logic reads $h(j)$ from $\mathbf{h}$ RAM, pre-updates $h(j)$ and keeps the updated element in a register h_t. It also checks whether h_t is in the range $[-H, H]$ $(\alpha = 0)$ or not $(\alpha = 1)$ before proceeding to the state

4. State 4 checks c and α to decide whether the iteration is successful. If it is successful, $\mathbf{r}$, $h(j)$ and index j are updated. Elements of vector $\mathbf{r}$ are updated at the same clock cycle because they are already stored in the registers. In addition, the iteration counter p is incremented by one and the binary variable Flag is set to 1 which indicates that the current iteration is "successful". The "successful" iteration counter p needs to be checked whether it reaches the maximum number N_u, and if so, the algorithm stops. If not, it decides whether to update the Flag and which next state to proceed depending on j and Flag.

The number of clock cycles required for each state is shown in Table 2.6. In each iteration, three clock cycles are needed. The upper number of the cycles in Table 2.6 can be considered as the worst case complexity of the parallel implementation of the fixed-point box-constrained DCD algorithm. We can consider the worst case situation, which occurs when only the last bit has N_u successful iteration. This means that the calculation of the first $(M_b - 1)$ bits do not contain any successful iteration, and so, require $(M_b - 1)3K$ cycles. The worst-case for calculating the last bit (m=1) occurs when only one successful iteration happens among the K iterations $(j = 1, \cdots, K)$. This requires $3K$ cycles for the comparison, and pre-update residual vector $\mathbf{r}$ and element $h(j)$. In total, N_u successful passes require $3KN_u$ cycles. Besides above cycles needed, there also requires M_b cycles for the step size update in the whole process. Therefore, the clock cycles of the fixed-point box-constrained DCD algorithm is upper bounded by $3KN_u + 3K(M_b - 1) + M_b$ clock cycles.

R-in-Register: Elements of the matrix $\mathbf{R}$ and vector $\mathbf{r}$ are stored in registers (see Fig.2.8). In state 1, the process of one-bit left shift of the vector $\mathbf{r}$ and the update of step size d are performed. In state 2, the elements in vector $\mathbf{r}$ and $\mathbf{R}(:, j)$ are read out to update the residual vector $\mathbf{r}$. In state 3, the comparison of $r(j)$ and $R(j, j)$ and pre-update of $h(j)$ are executed in one clock cycle. In state 4, the final updated elements in the vector $\mathbf{r}$ and $h(j)$ are obtained in one clock cycle. In this case, there is a high area expense since all elements of the matrix $\mathbf{R}$ are stored in registers. To further improve the design, we consider storing $\mathbf{R}$ in a block of RAM.

R-in-RAM: A row of elements of the matrix $\mathbf{R}$ are stored in an array of RAMs instead of registers. This achieves a significant reduction in the required number of slices compared to that required for the case where elements of $\mathbf{R}$ are stored in registers. This design

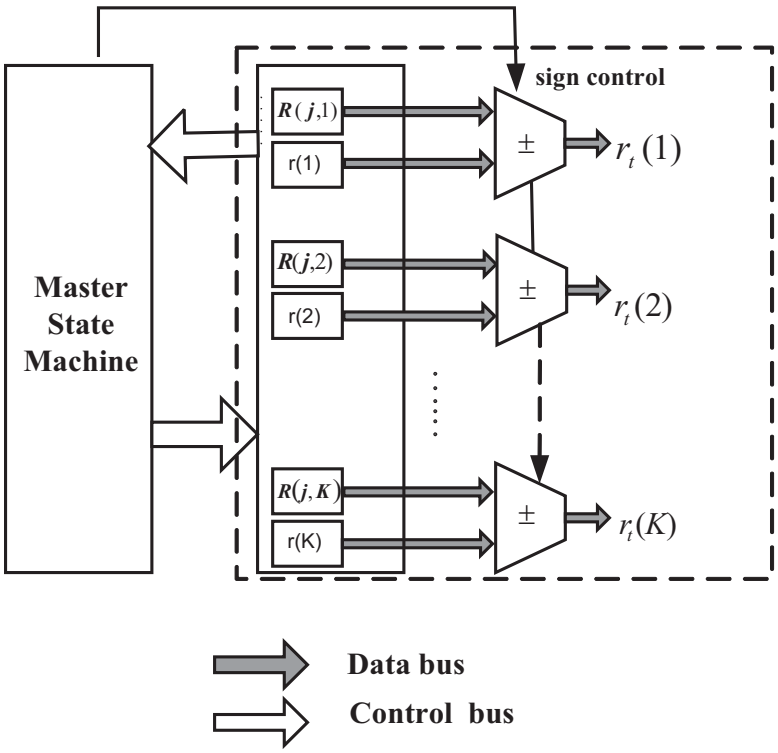


Figure 2.8: $\mathbf{r}$ Update Logic Architecture in $\mathbf{R}$-in-Register.

is represented in Fig.2.9.

Table 2.7: FPGA resources of parallel implementation of the box-constrained DCD algorithm for $K = 16$ and $M_b = 15$.

FPGA Resources	R-in-Register	R-in-RAM
Slices	7176	1465
D-FFs	5123	802
LUT4s	5646	2754
Block RAMs	2	18

The parallel designs have improved the throughput in comparison with the serial design of the box-constrained DCD detector in FPGA implementation. However, the two parallel design implementations are more applicable for scenarios with a small number of users due to the high hardware resources usage.

The FPGA resources of both $\mathbf{R}$-in-Register and $\mathbf{R}$-in-RAM implementations are presented in Table 2.7 for the case of $K = 16$ and $M_b = 15$. The FPGA resources of the $\mathbf{R}$-in-Register implementation is higher than that of the $\mathbf{R}$-in-RAM implementation, because it needs more registers. Therefore $\mathbf{R}$-in-Register implementation is suitable for

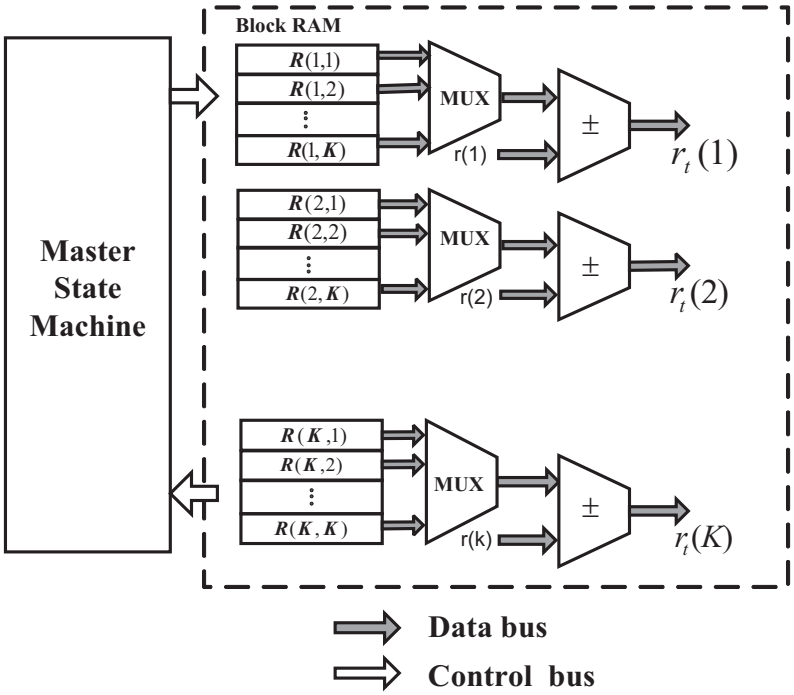


Figure 2.9: $\mathbf{r}$ Update Logic in $\mathbf{R}$-in-RAM.

small size systems. The $\mathbf{R}$-in-RAM implementation reduces the area usage in comparison with that of the $\mathbf{R}$-in-Register implementation. However it still requires more slices than that of the serial architecture box-constrained DCD algorithm implementation.

2.6 Conclusions

In this chapter, we have investigated the box-constrained DCD-based multiuser detector. This algorithm is multiplication-free and division-free; therefore, it is suitable for hardware implementation. We have presented and compared the serial and parallel implementations of the FPGA architecture of the box-constrained DCD algorithm. The proposed fixed-point implementation of the box-constrained DCD algorithm provides a high-accuracy performance which is very close to that of floating-point implementation. This algorithm outperforms the well-known MMSE detector in differently loaded scenarios, and even using a small number of bits M_b.

When the box-constrained DCD algorithm is applied in the multiuser detection, the serial implementation of the FPGAarchitecture requires less than 400 FPGA slices for a number

of users as large as 50. However, the throughput of this design can be further improved by using an alternative parallel implementation. Two efficient designs for FPGA parallel implementation have been proposed as: $\mathbf{R}$-in-Register design and $\mathbf{R}$-in-RAM design.

The parallel implementation of the algorithm provides higher data throughput than that of the serial implementation. The drawback of the parallel implementation designs is that when the system size increases, the FPGA resources required in the $\mathbf{R}$-in-Register based implementation increase significantly; *e.g.* it needs 7176 slices when $K = 16$. The number of slices used in the $\mathbf{R}$-in-RAM design is reduced at the expense of a few more RAM blocks.

Chapter 3

Box-constrained DCD Algorithm for MIMO Detection of Complex-valued Symbols

Contents

3.1 Introduction

Multiple-Input Multiple-Output (MIMO) communication systems with spatial multiplexing allow the channel capacity to be increased compared to Single-Input Single-Output (SISO) communication systems [59]. This requires efficient detection techniques to be used at the receiver. In practical scenarios, we need to consider a reasonable way for hardware implementation of MIMO detectors, *e.g.* using an FPGA platform. The maximum likelihood (ML) MIMO detector provides optimal performance; however, it is complicated for real-time implementation. The ML detector can be directly implemented in

hardware only for small size MIMO systems with low order modulation (*e.g.* for a 4 transmit and 4 receive antennas MIMO system with QPSK modulation) [80].

The sphere decoder is considered as a good candidate for hardware implementation of the optimal detector. However, it also becomes more complicated when the size of the system and modulation order increases. In addition, this algorithm requires high complexity at low SNRs. Usually, the decoder's throughput is given at an SNR of 20 dB [70, 81], whereas at lower SNRs the throughput becomes significantly lower. This is why the sphere decoders are usually implemented on FPGA platforms for small size MIMO systems, such as 4×4 systems [81, 82]. Moreover, a matrix factorization, such as Cholesky decomposition or QR decomposition, and decorrelating detection are required before applying the sphere decoder. This is difficult for real-time hardware implementation [74,81].

The optimal detector and its sub-optimal approximations such as sphere decoders [46, 70] provide hard decisions. However, the soft decisions are of interest as they allow further efficient decoding. The MIMO detection with soft decision can be based on MMSE detection. Recently, different FPGA designs of the MMSE MIMO detection were reported [83–85]. However, the hardware complexity of a MIMO system increases rapidly with the number of antennas, and only MMSE MIMO detectors with small size were implemented on FPGA platform. Moreover, the performance of MMSE detectors can be significantly inferior to the optimal detection performance in large size systems.

In this chapter, we consider box-constrained MIMO detection that also provides soft output. The box-constrained detection is well known in application to multiuser CDMA systems [86]. It shows a better detection performance compared to the MMSE detection performance. Moreover, it can be efficiently implemented using dichotomous coordinate descent (DCD) iterations [75]. An FPGA design of a box-constrained DCD-based multiuser detector was presented in Chapter 2. However, that design can only be used for real-valued systems, *e.g.* for systems with BPSK modulation and real-valued system matrix $\mathbf{R}$. In MIMO systems, complex-valued modulation schemes, such as QPSK, 16-QAM and others, are of interest. In this chapter, we present an FPGA design of a complex-valued DCD-based box-constrained MIMO detector of symbols with various QAM modulations and compare it with designs of MMSE MIMO detectors in terms of the design area, throughput and detection performance.

This chapter is organized as follows. In Section 3.2, the signal and channel model for MIMO systems are presented. In Section 3.3, the FPGA implementation of the box-

constrained DCD detector is described. Simulation results and performance analysis are given in Section 3.4, and Section 3.5 concludes the chapter.

3.2 System Model and Box-constrained MIMO Detector

We consider an $M_T \times M_R$ ($M_T = M_R$) MIMO system with M_T transmit and M_R receive antennas in frequency-flat Rayleigh fading channels. The received signal is given by

$$\mathbf{y} = \mathbf{G}\mathbf{h} + \mathbf{n}\,, \tag{3.1}$$

where $\mathbf{G}$ is an $M_R \times M_T$ channel matrix whose entries are independent identically distributed (i.i.d.) zero mean Gaussian random numbers, $\mathbf{h}$ is an $M_T \times 1$ vector of transmitted data from a QAM constellation $\mathcal{A}$, and $\mathbf{n}$ is the noise vector whose entries are i.i.d. zero mean Gaussian random numbers. The ML detector solves the quadratic optimization problem with integer constraints:

$$\begin{aligned} \hat{\mathbf{h}}_{ML} &= \arg \min_{\mathbf{h} \in \mathcal{A}^{M_T}} \left\{ ||\mathbf{y} - \mathbf{G}\mathbf{h}||^2 \right\} \\ &= \arg \min_{\mathbf{h} \in \mathcal{A}^{M_T}} \left\{ \mathbf{h}^H \mathbf{R}\mathbf{h} - 2\boldsymbol{\theta}^H \mathbf{h} \right\}\,, \end{aligned} \tag{3.2}$$

where $\mathbf{R} = \mathbf{G}^H\mathbf{G}$ and $\boldsymbol{\theta} = \mathbf{G}^H\mathbf{y}$. The box-constrained detector relaxes the constraint $\mathbf{h} \in \mathcal{A}^{M_T}$ into $\Re\{\mathbf{h}\} \in [-H, H]^{M_T}$ and $\Im\{\mathbf{h}\} \in [-H, H]^{M_T}$, where H is the maximum magnitude among real and imaginary elements of the constellation $\mathcal{A}$. For example, for 16-QAM modulation, $H = 3$. Thus, the box-constrained MIMO detector solves the problem

$$\hat{\mathbf{h}}_{box} = \arg \max_{\Re\{\mathbf{h}\} \& \Im\{\mathbf{h}\} \in [-H,H]^{M_T}} \left\{ \mathbf{h}^H \mathbf{R}\mathbf{h} - 2\boldsymbol{\theta}^H \mathbf{h} \right\}. \tag{3.3}$$

This solution provides a soft output $\hat{\mathbf{h}}_{box}$, which is then mapped into $\mathcal{A}^{M_T}$. Solving (3.3) is equivalent to solving the normal equations

$$\mathbf{R}\mathbf{h} = \boldsymbol{\theta} \tag{3.4}$$

with the box-constraint in solution $\mathbf{h}$.

Table 3.1: Complex-valued box-constrained DCD algorithm

State	Operation	Cycles
0	Initialization: $\mathbf{h} = \bar{\mathbf{h}}$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, $m = M_b$, $p = 0$ $\Delta m = 0$, $s = 1$, $j = 1$, Flag$= 0$	
1	if $m = 0$, algorithm stops else, $m = m - 1$, $d = 2^m$, $\Delta m = \Delta m + 1$	1
2	if $s = 1$, then $r_t = \Re(r(j))$; else, $r_t = \Im(r(j))$ $c = R(j,j)/2 - \lvert r_t \rvert \times 2^{\Delta m}$ $h_t = h(j) + \text{sign}(r_t)sd$ if $\lvert\Re(h_t)\rvert \leq H$ and $\lvert\Im(h_t)\rvert \leq H$, then $\alpha = 0$; else, $\alpha = 1$	1
3	if $c < 0$ and $\alpha = 0$, then goto state 4 else, goto state 5	1
4	$h(j) = h_t$ $\mathbf{r} = \mathbf{r} \times 2^{\Delta m} - \text{sign}(r_t)s\mathbf{R}^{(j)}$ $\Delta m = 0$, $p = p + 1$, Flag = 1 if $p = N_u$, algorithm stops	M_T
5	if $s = 1$, then $s = i$, goto state 2 else, $s = 1$, $j = (j)\text{mod}(M_T) + 1$ if $j = 1$ and Flag = 1, then Flag = 0, goto state 2 elseif $j = 1$ and Flag = 0, then goto state 1 else, goto state 2	1
Total:	$\leq 7M_TN_u + 6M_T(M_b - 1) + M_b$ cycles	

3.3 FPGA Implementation of DCD-based Box-constrained MIMO Detector

The DCD algorithm solves the normal equations (3.4) using coordinate descent iterations with a varying power-of-two step-size [8]. There are two types of iterations: *successful* and *unsuccessful*. At a successful iteration, one element of the solution vector $\mathbf{h}$ and all elements of the residual vector $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}$ are updated; these iterations contribute most in the algorithm complexity. At an unsuccessful iteration, there is no update. The maximum number of successful iterations (updates) N_u is predefined. Another parameter M_b is predefined, which indicates the number of bits representing elements of $\mathbf{h}$, and so, controls the final accuracy of the solution. It also decides how many times the step-size d is reduced by a factor of two.

The structure of the box-constrained DCD algorithm for complex valued modulation schemes is optimized for FPGA implementation and presented in Table 3.1. The archi-

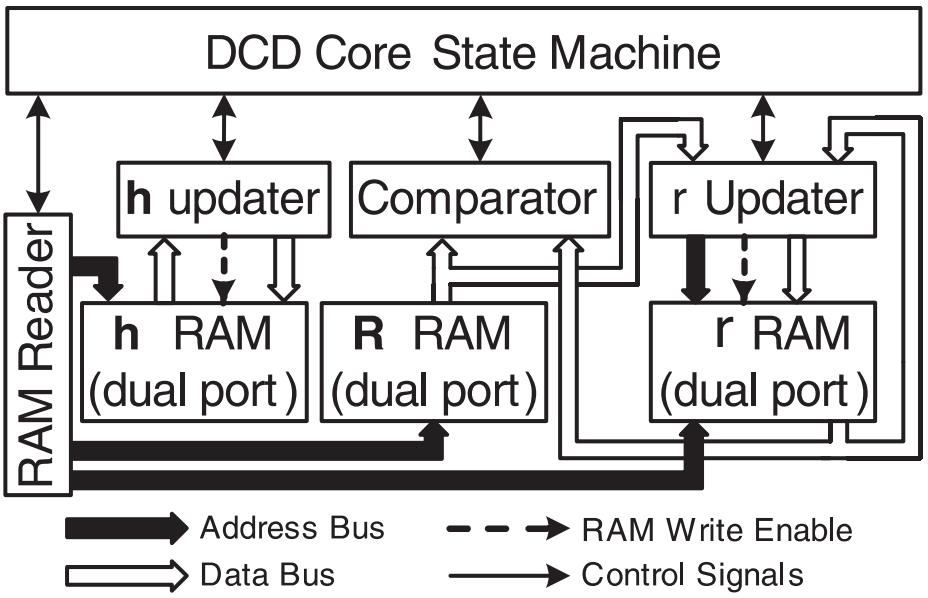


Figure 3.1: Block-diagram of the DCD processor

tecture of the DCD processor is shown in Fig. 3.1.

A DCD Core State Machine controls the operations of the algorithm. In state 0, it initializes several control signals as shown in Table 3.1. The residual vector $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, where $\bar{\mathbf{h}}$ is the initialization of $\mathbf{h}$. In this chapter, $\bar{\mathbf{h}}$ is set to zero, and $\mathbf{r}$ is equal to $\boldsymbol{\theta}$. Bit counter m is set to M_b, successful iteration counter p is set to 0, pre-scaling counter Δm is set to 0, signal s is set to 1 for real or set to i for imaginary component of the element $r(j)$, element index j is set to 1 and successful iteration indicator 'Flag' is set to 0.

In state 1, it updates a bit counter m, step size $d = 2^m$, and prescaling counter Δm. The condition $m = 0$ implies that the least significant bits of elements of the solution vector have been decided and the DCD processor stops; otherwise, the algorithm proceeds to state 2.

In state 2, RAM Reader asserts addresses of $r(j)$ in $\boldsymbol{\theta}$ RAM, $R(j, j)$ in $\mathbf{R}$ RAM, and $h(j)$ in $\mathbf{h}$ RAM. Then, according to s, the Comparator selects the real (if $s = 1$) or imaginary (if $s = i$) component of the element $r(j)$. If the vector $\mathbf{r}$ has not been prescaled for a new mth bit, the Comparator scales r_t. It then performs the comparison and passes the sign bit of the result c to DCD Core State Machine. Simultaneously, $\mathbf{h}$ Updater reads $h(j)$ from $\mathbf{h}$ RAM, pre-updates the (real or imaginary according to s) component of $h(j)$, keeps the updated element as h_t in a register, and checks whether it is in the range $[-H, H]$ ($\alpha = 0$) or not ($\alpha = 1$), where the value of H depends on the modulation, *i.e.* $H = 1$ for M-PSK.

In state 3, DCD Core State Machine examines the comparison results c and α to decide which step to proceed. If $c < 0$ and $\alpha = 0$ (the iteration is successful), the algorithm proceeds to state 4 to update $h(j)$ and $\mathbf{r}$; otherwise, it proceeds to state 5 without updates (the iteration is unsuccessful).

In state 4, $\mathbf{r}$ Updater updates $\mathbf{r}$. RAM Reader generates addresses of elements of the column $\mathbf{R}(:, j)$ and vector $\mathbf{r}$ in $\mathbf{R}$ RAM and $\boldsymbol{\theta}$ RAM, respectively. $\mathbf{r}$ Updater reads elements of $\mathbf{r}$ from $\boldsymbol{\theta}$ RAM, updates them, and writes the result back. $\mathbf{r}$ Updater has two adders for simultaneous updating real and imaginary components of $\mathbf{r}$. $\mathbf{h}$ Updater writes h_t into $\mathbf{h}$ RAM to replace $h(j)$. Then DCD Core State Machine sets the prescaling counter Δm to 0 and the variable Flag to 1, indicating a successful iteration. The counter p of successful iterations is also updated; if p is equal to the predefined limit N_u, the DCD processor stops; otherwise, it proceeds to state 5.

In state 5, DCD Core State Machine firstly tests s to decide which component of $\mathbf{h}$ should be analyzed next. Then, it updates the index j and signal s. After that, it decides whether to update the Flag and which next state to proceed depending on j and Flag.

The number of clock cycles required in each state is shown in Table 3.1. Pipelining is used in state 4, and so, the design requires M_T cycles to update $h(j)$ and all elements in $\mathbf{r}$. The other states require a single cycle each. The total number of cycles for solving a system of equations varies depending on the system size, the condition number of the system matrix, and the algorithm parameters, N_u and M_b. For a given M_T, N_u and M_b, the number of cycles can be considered to be a random number with an upper bound corresponding to a worst-case scenario. The upper bound can be shown to be $7M_TN_u + 6M_T(M_b - 1) + M_b$, or for high N_u we have approximately $7M_TN_u$. This corresponds to an unlikely situation when, in every pass, only one successful iteration (one update) happens. In a typical situation, there are many successful iterations in every pass, and so, the average number of clock cycles will be smaller as explained below.

Each element of a complex-valued system of equations is represented using two 16-bit Q15 numbers [87] for representing real and imaginary components, these are limited to the range $[-1, 1)$. To avoid overflow, real and imaginary components of $\mathbf{r}$ and $\mathbf{h}$ are stored using the 32-bit fixed-point Q15 format and are limited to the range $[-2^{16}, 2^{16})$. To obtain a high update rate, the real and imaginary components of $\mathbf{r}$ are processed in parallel. $\mathbf{R}$ RAM has a 32-bit data width and $\boldsymbol{\beta}$ RAM has a 64-bit data width. This enables them to support both components simultaneously at each read or write operation. $\mathbf{h}$ RAM has a

Table 3.2: FPGA resources required for box-constrained DCD-based MIMO detector of complex-valued symbols.

Resources	$M_T = 4$	$M_T = 8$	$M_T = 16$
Slices	637 (4.7%)	658 (4.8%)	667 (4.9%)
D-FFs	305 (1.1%)	318 (1.2%)	329 (1.2%)
LUT4s	1033 (3.8%)	1062 (3.9%)	1084 (4.0%)
Block RAMs	5 (3.7%)	5 (3.7%)	5 (3.7%)

32-bit data width, since only one component of $h(j)$ is required at each iteration.

Table 3.2 presents FPGA resources required for box-constrained DCD-based MIMO detector of complex-valued symbols. It can be seen that the area usage for the box-constrained DCD-based algorithm is very small, and is 637 to 667 slices for systems of equations of the size $M_T = 4$ to $M_T = 16$, respectively. In all the cases, the design occupies less than 5% of the chip area, and it does not use any built-in multipliers. This design has a significantly lower area usage compared to the designs of the MMSE detector for 4×4 MIMO systems reported in [83–85], which require 8513, 7679, and 9474 slices, respectively. In addition, the MMSE detectors presented in [83, 84] use 64 and 58 area-expensive multipliers, respectively, compared to the zero number of multipliers in the DCD-based algorithm.

3.4 Numerical Results

In the AWGN channel, we will present numerical results that allow us to estimate the throughput of the proposed design. Specifically, the convergence speed of the design, in terms of the number of updates and number of clock cycles, is demonstrated for 4×4, 8×8, and 16×16 MIMO systems with 16-QAM modulation.

By solving (3.3), the DCD algorithm ($M_b = 15$) obtains a solution $\hat{\mathbf{h}}_{box}$. The misalignment between estimated data vectors $\hat{\mathbf{h}}_{box}$ and transmit data vector $\mathbf{h}$ is calculated as

$$\xi = \frac{||\hat{\mathbf{h}}_{box} - \mathbf{h}||^2}{||\mathbf{h}||^2}. \tag{3.5}$$

The misalignment is averaged over a number of $T = 1000$ simulation trials and is given

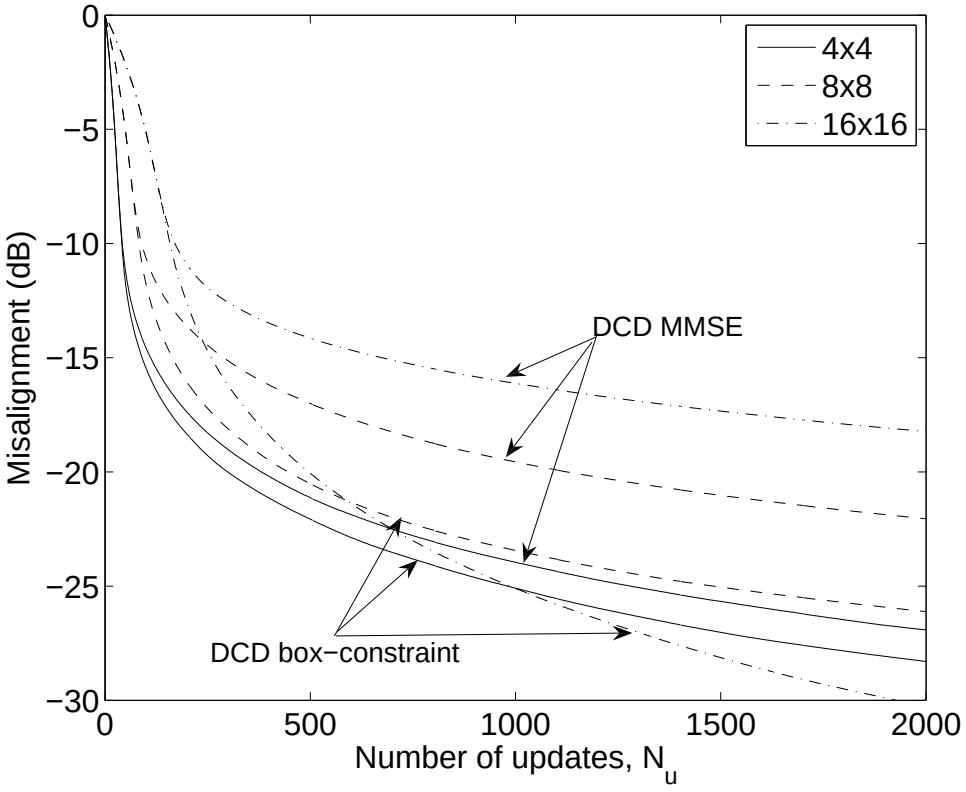


Figure 3.2: Misalignment *vs* number of updates N_u in 16-QAM MIMO systems.

in decibels by

$$\bar{\xi} = 10\lg\left\{\frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{j=1}^{M_T}|\hat{h}_{box}(j)-h(j)|^2}{\sum_{j=1}^{M_T}|h(j)|^2}\right\}. \tag{3.6}$$

This misalignment is plotted against the number of updates N_u in Fig.3.2, which is obtained from MATLAB. For the purpose of comparison, we also show results for the MMSE MIMO detector, which is implemented using the DCD algorithm. This MMSE detector algorithm is different than the box-constrained detector in that the comparison with the threshold H in states 2 and 3 in table 3.1 is removed and the matrix $\mathbf{R}$ is replaced by $\mathbf{R}+\frac{1}{SNR}\mathbf{I}$, where $\mathbf{I}$ is an $M_T \times M_T$ identity matrix. It can be seen that the box-constrained solution provides significantly lower misalignment than the MMSE solution, and the difference in the performance increases as the system size M_T increases. Fig.3.3 shows the misalignment against the number of cycles, which is obtained from FPGA platform. Comparing Fig.3.3 and Fig.3.2 for a fixed misalignment *i.e.* -25 dB, we can conclude that one update on average requires approximately $2.5M_T$, $2M_T$ and $1.7M_T$ cycles for 4×4, 8×8 and 16×16 MIMO systems, respectively. This is significantly lower than that in the worst-case scenario discussed above. Thus, the total number of clocks required is approximately $2.5M_TN_u$ for 4×4 MIMO systems, $2M_TN_u$ for 8×8 MIMO systems, and $1.7M_TN_u$ for 16×16 MIMO systems.

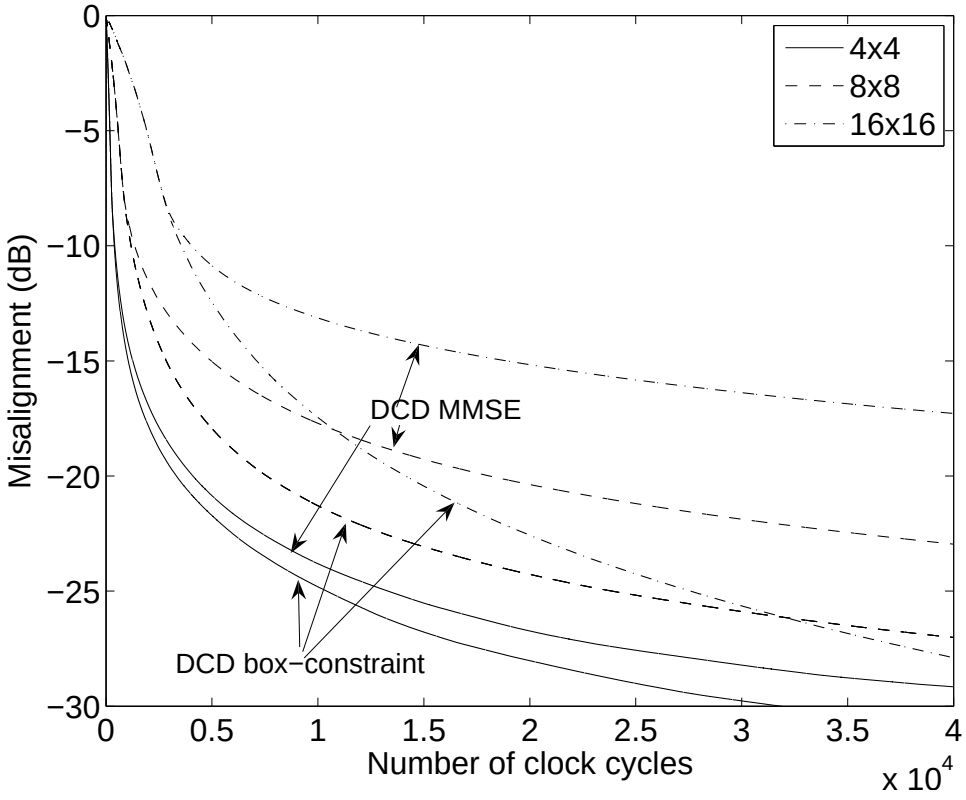


Figure 3.3: Misalignment *vs* number of clock cycles in 16-QAM MIMO systems.

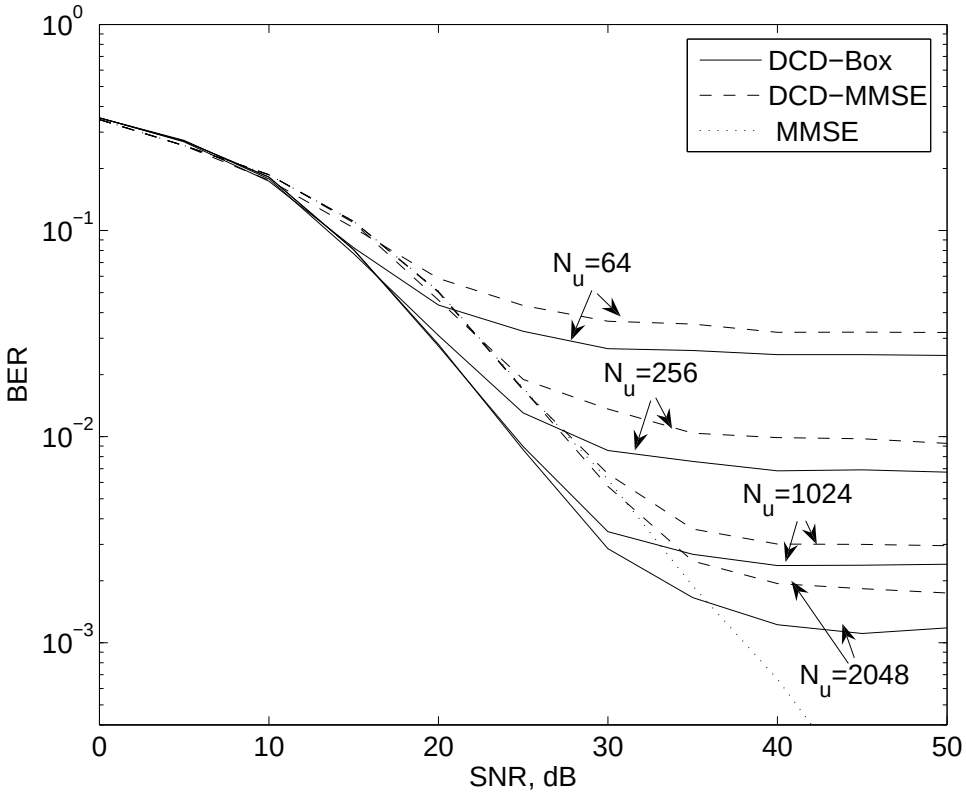


Figure 3.4: BER performance of the detectors in 4×4 MIMO systems.

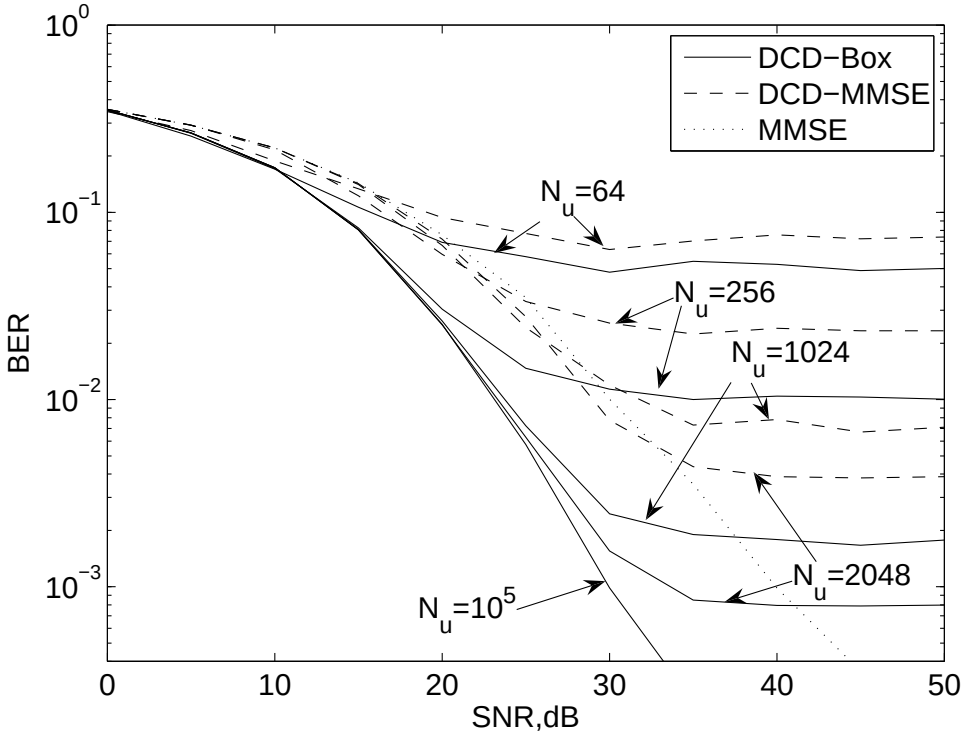


Figure 3.5: BER performance of the detectors in 8×8 MIMO systems.

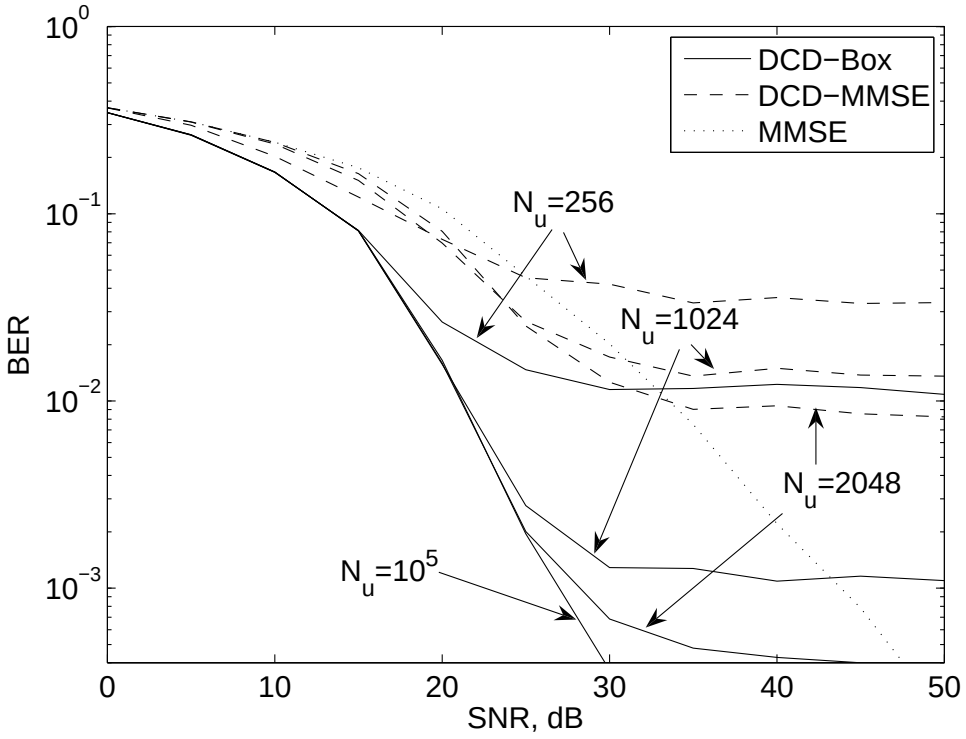


Figure 3.6: BER performance of the detectors in 16×16 MIMO systems.

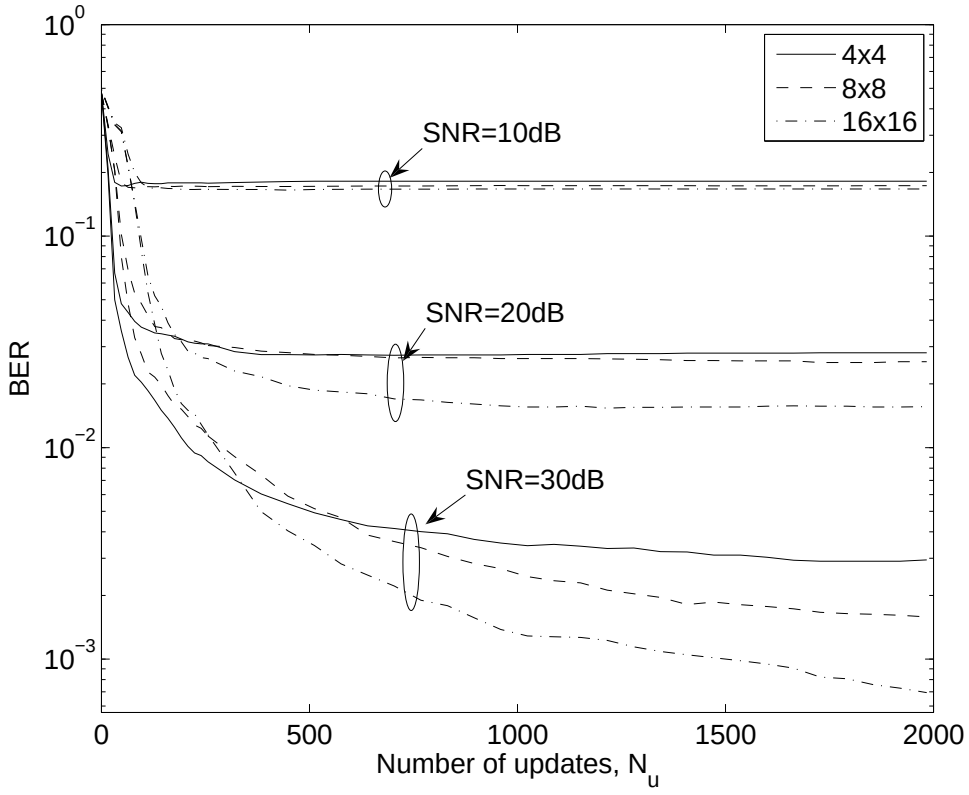


Figure 3.7: BER *vs* number of updates N_u in 16-QAM MIMO systems.

Fig.3.4, Fig.3.5 and Fig.3.6 show the bit-error-rate (BER) performance against SNR for the box-constrained detector compared with that of the DCD-based MMSE detector and the classical MMSE detector implemented in floating point for 4×4, 8×8, and 16×16 MIMO systems, respectively. The figures show that the box-constrained detector can significantly outperform the MMSE detectors, especially for large MIMO systems. For a fixed N_u, the detectors based on DCD iterations exhibit a BER floor, which is reduced as N_u increases.

Fig.3.7 shows improvement in the BER performance achieved by the box-constrained detector as the number of updates increases for 4×4, 8×8, and 16×16 MIMO systems. For the 4×4 case, it can be seen from Fig.3.4 that the BER of the MMSE detector is 0.18, 0.05 and 0.006 at SNR = 10 dB, 20 dB and 30 dB, respectively. Fig.3.7 shows that the box-constrained detector can achieve these BERs with about $N_u = 27$, 47 and 410 updates corresponding to 270, 470 and 3800 cycles, respectively. We can conclude that, from the viewpoint of the throughput (refers to the amount of data that is processed per clock cycle), the proposed design is inferior to the MMSE designs in [83–85], that require from 270 to 388 cycles. However, due to the very low area usage of our design (less than 5% of the Xilinx Virtex-II Pro Development System [77] with an XC2VP30 (FFT896 pack-

Table 3.3: Number of cycles required by the box-constrained DCD-based MIMO detector to achieve BER performance of the MMSE detector

SNR	4×4	8×8	16×16
10 dB	270	650	1800
20 dB	470	1100	2600
30 dB	3800	4800	4600

age, speed grade 7), we can implement approximately 20 DCD-based box-constrained detectors using the entire area of this FPGA chip. By having 20 box-constrained detectors working in parallel, we can reduce the average number of cycles for detection of one MIMO symbol to 14, 24 and 190 cycles at SNR = 10 dB, 20 dB and 30 dB, respectively. These are significantly lower than those of the MMSE detector. The use of more advanced FPGA chips, such as Virtex-5 XCE5VLX330 [88], would allow increasing the number of parallel DCD-based MIMO detectors to 200. Note that in OFDM MIMO systems, implementation of a number of detectors working in parallel, *e.g.* one detector per a subcarrier, can be beneficial.

As the size of the MIMO system increases, the proposed design becomes relatively more efficient. In particular, for the 8×8 MIMO system, at SNR = 10 dB, 20 dB and 30 dB, the BER performance of the MMSE detector (BER = 0.22, 0.07 and 0.01) is achieved with $N_u = 40$, 68 and 297 updates, *i.e.* approximately 650, 1100 and 4800 cycles, respectively. For 16×16 MIMO systems, the box-constrained detector needs about 1800 cycles ($N_u = 68$) at SNR = 10 dB, 2600 cycles ($N_u = 95$) at SNR = 20 dB and 4600 cycles ($N_u = 170$) at SNR = 30 dB to achieve the BER performance of the MMSE detector (BER = 0.25, 0.1 and 0.02, respectively). Table 3.3 shows the cycle count results. It can be seen that the proposed design can be especially useful for large MIMO systems.

3.5 Conclusions

In this chapter, an FPGA design of a box-constrained MIMO detector based on DCD iterations has been presented. This design requires significantly lower area usage than known designs of the MMSE MIMO detector. Moreover, the box-constrained detector can achieve significantly better detection performance than that of the MMSE detector, especially for large MIMO systems. For small (4×4) MIMO systems, our design may require a higher cycle count than the MMSE designs. However, the parallel implemen-

tation of many box-constrained detectors (due to its low area usage) in one FPGA chip can proportionally increase the throughput and make it higher than that of the MMSE designs. Since SNR decreases, the required number of cycles for the DCD-based box-constrained detector is significantly reduced, therefore, a combined detector that uses the sphere decoding for high SNRs, and uses the proposed detector for low SNRs, can be a useful practical solution for MIMO detection. This is similar to a combined multiuser detector as proposed in Chapter 7 below. The proposed design provides a soft output and requires a relatively small number of cycles at low SNRs, and so, is attractive for MIMO systems based on coded transmission (which can operate at low SNRs and usually require a soft output from the detector).

Chapter 4

DCD-BTN Algorithm and FPGA Design

Contents

4.1 Introduction

An iterative multiuser detection technique based on box-constrained solution with Gauss-Seidel iterations and nonstationary iterative deregularization (GS-BD) has been proposed in [89]. This multiuser detection is based on the iterated Tikhonov regularization [90]. In [89], deregularization is used. However, the deregularization maximizes the energy of the solution, in opposite to minimizing it in the Tikhonov regularization. Combined with box-constraints, the deregularization forces the solution to be close to the binary set

and improves the "efficient frontier" with the complexity as low as $K^{3.2}$ FLOPS. In this chapter, we further develop the GS-BD technique by exploiting the box-constrained DCD algorithm and adapting it to the nonstationary iterative Tikhonov regularization to propose the DCD-BTN detector. As a result, the worst-case and average complexity are reduced down to $K^{2.8}$ and $K^{2.5}$ FLOPS, respectively. In this chapter, the implementation of the proposed detector will be discussed. The detection performance obtained from the fixed-point FPGA implementation shows a good match to the floating-point implementation.

This chapter is organised as follows. In Section 4.2, the signal and channel models for multiuser detection are presented. In Section 4.3, the Non-stationary Tikhonov Iterative Regularization is described. In Section 4.4, the detectors with Box-Constrained Relaxation are presented. In Section 4.5, the detector based on the nonstationary Tikhonov iterations with box-constraint and negative diagonal loading (DCD-BTN) is derived. In Section 4.6, the way of updating residual vector is derived. In Section 4.7, the simulation results for the DCD-BTN detector are presented. In Section 4.8, the hardware architecture design of the DCD-BTN detector is described and corresponding numerical results for FPGA implementation are given. Section 4.9 concludes the chapter.

4.2 Problem Formulation of Multiuser detection

The matched-filter output at a symbol synchronous CDMA receiver is given by the K-length vector

$$\boldsymbol{\theta} = \mathbf{R}\mathbf{h} + \mathbf{n}\,, \tag{4.1}$$

where the vector $\mathbf{h} \in \{-1, +1\}^K$ contains bits transmitted by the K users, $\mathbf{R}$ is a positive definite signature waveform correlation $K \times K$ matrix, and $\mathbf{n}$ is a real-valued zero-mean Gaussian random vector with covariance matrix $\sigma^2\mathbf{R}$. The optimal ML multiuser detector estimates the vector $\mathbf{h}$ by minimizing the following quadratic function

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - \boldsymbol{\theta}^T\mathbf{h} \right\} \tag{4.2}$$

with binary constraints $\mathbf{h} \in \{-1, +1\}^K$. Although the ML detector provides the best detection performance, it is not practical due to its high complexity.

4.3 Non-stationary Iterative Tikhonov Regularization

The MMSE detector solves the unconstrained quadratic optimization problem [9]

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in \mathcal{R}^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - \boldsymbol{\theta}^T\mathbf{h} + \frac{\lambda}{2}\mathbf{h}^T\mathbf{h} \right\} \tag{4.3}$$

using the regularization term $(\lambda/2)\mathbf{h}^T\mathbf{h}$, where $\mathcal{R} = (-\infty, +\infty)$. This regularization is also known as diagonal loading [91], which promotes a solution with a small energy. A more accurate solution to the problem (4.2) can be obtained using the non-stationary Tikhonov iterative regularization [92, 93]. Instead of solving (4.3), a sequence of $(N+1)$ unconstrained optimization problems can be solved, with the nth unconstrained optimization problem

$$\hat{\mathbf{h}}^{(n)} = \arg \min_{\mathbf{h} \in \mathcal{R}^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - (\boldsymbol{\theta} + \lambda\tilde{\mathbf{h}}^{(n-1)})^T\mathbf{h} + \frac{\lambda}{2}\mathbf{h}^T\mathbf{h} \right\}, \tag{4.4}$$

where $n = 0, \cdots, N$ is the Tikhonov iteration index, $\tilde{\mathbf{h}}^{(-1)} = \mathbf{0}$, $\tilde{\mathbf{h}}^{(n-1)} = \hat{\mathbf{h}}^{(n-1)}$ and $\lambda > 0$.

The solution of (4.4) can be regarded as an unconstrained solution to the equation

$$(\mathbf{R} + \lambda\mathbf{I})\hat{\mathbf{h}}^{(n)} = \boldsymbol{\theta} + \lambda\tilde{\mathbf{h}}^{(n-1)}. \tag{4.5}$$

When $N = 0$, we obtain the MMSE detector where the final solution is $\hat{\mathbf{h}} = \text{sign}[\tilde{\mathbf{h}}^{(0)}]$. When $N \geq 1$, the described multiuser detector provides a better performance than the MMSE detector [94], and in this case, the final solution is given by the hard decision $\hat{\mathbf{h}} = \text{sign}\{\tilde{\mathbf{h}}^{(N)}\}$.

As proposed in [94], λ varies with n and according to

$$\lambda_n = \begin{cases} \sigma^2 & : \quad n = 0 \\ n\sigma^2 & : \quad n = 1, \cdots, N, \end{cases} \tag{4.6}$$

$\tilde{\mathbf{h}}^{(n)}$ is a semi-hard version of $\hat{\mathbf{h}}^{(n)}$, for which the kth element is given by [94]

$$\tilde{h}_k^{(n)} = \begin{cases} +1 & : \quad \text{if} \quad \hat{h}_k^{(n)} > n^\alpha \\ -1 & : \quad \text{if} \quad \hat{h}_k^{(n)} < -n^\alpha \\ \hat{h}_k^{(n)} & : \quad \text{otherwise}, \end{cases} \tag{4.7}$$

and $\alpha < 0$ is a pre-defined coefficient.

Obtaining a direct solution of (4.5) requires approximately K^3 FLOPS. Consequently, a direct implementation of the detector with N Tikhonov iterations requires $(N+1)K^3$ FLOPS. In our simulation below, we call this direct implementation Direct-T detector. This detector achieves a good detection performance [89]; however, its complexity is high. In order to reduce the complexity, equation of (4.5) can be solved approximately using the classical Gauss-Seidel (GS) iterations [89]. In most cases, the GS iterations can significantly reduce the detector complexity without a significant degradation in performance. We call the Direct-T detector using GS iterations as GS-T detector. The GS-T detector approaches the "efficient frontier" (described in (1.2.3)), but does not improve it.

4.4 Detectors with Box-constrained Relaxation

The complicated optimization problem (4.2) can be approximately solved by relaxing the binary constraint $\mathbf{h} \in \{-1,+1\}^K$. A multiuser detector with sphere constraint provides a detection performance similar to that of the MMSE detector [33]. Using the box-constraint $\mathbf{h} \in [-1,+1]^K$ results in a better performance than the sphere-constrained minimization [11, 15, 33]. A multiuser detector with box-constraint solves the convex optimization problem given as

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h}\in[-1,+1]^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - \boldsymbol{\theta}^T\mathbf{h} \right\}. \tag{4.8}$$

The solution to (4.8) can be regarded as a box-constrained solution to the equation

$$\mathbf{R}\mathbf{h} = \boldsymbol{\theta}. \tag{4.9}$$

The equation (4.9) can be solved by nonlinear GS iterations [95]. Each iteration consists of two steps. At the first step, the classical GS iteration is performed as

$$\hat{h}_i^{(j)} = \frac{1}{R_{ii}} \left\{ \theta_i - \sum_{k=1}^{i-1} R_{ik}\tilde{h}_k^{(j)} - \sum_{k=i+1}^{K} R_{ik}\tilde{h}_k^{(j-1)} \right\}, \tag{4.10}$$

where $j = 1, \cdots, N_{GS}$ is the iteration index, $\hat{h}_i^{(j)}$ and $\tilde{h}_i^{(j)}$ are elements of vectors $\hat{\mathbf{h}}^{(j)}$ and $\tilde{\mathbf{h}}^{(j)}$, respectively, $\tilde{\mathbf{h}}^{(0)} = \mathbf{0}$ and R_{ik} is the (i,k)th element of the matrix $\mathbf{R}$. At the second step, the semi-hard update is performed as

$$\tilde{h}_i^{(j)} = \begin{cases} +1 & : \quad \text{if} \quad \hat{h}_i^{(j)} > +1 \\ -1 & : \quad \text{if} \quad \hat{h}_i^{(j)} < -1 \\ \hat{h}_i^{(j)} & : \quad \text{otherwise.} \end{cases} \tag{4.11}$$

The final solution is obtained by applying the hard decision: $\hat{\mathbf{h}} = \text{sign}[\tilde{\mathbf{h}}^{(N_{GS})}]$.

The coordinate descent iterations used by the box-constrained DCD algorithm, which was described in Chapter 2, can also solve equation (4.9).

4.5 Nonstationary Tikhonov Iterations with Box-constraint and Negative Diagonal Loading

In this section, we introduce the regularization with negative loading and justify the use of the regularization together with the box-constraint. Then, we introduce the detector that performs Tikhonov iterations with such regularization and box-constraint. Finally, we show how the proposed DCD based detector can be further simplified by exploiting inherent properties of the DCD algorithm.

The box-constraint tightens the solution set, which results in a better detection performance compared to that with the unconstrained solutions. An additional tightening of the solution could further improve the performance. The binary constraint $\mathbf{h} \in \{-1, +1\}^K$ implies that $h_i^2 = 1$, where h_i is the ith element of the vector $\mathbf{h}$, hence $\mathbf{h}^T\mathbf{h} = K$. Within the K dimensional box $[-1, +1]^K$, the solution $\mathbf{h}$ satisfies the inequality $(K - \mathbf{h}^T\mathbf{h}) \geq 0$. Therefore, for forcing the vector $\mathbf{h} \in [-1, +1]^K$ to be close to the binary set $\mathbf{h} \in \{-1, +1\}^K$, we need to minimise the term $(K - \mathbf{h}^T\mathbf{h})$. Thus an additional tightening of the solution set can be achieved by introducing the following optimization problem

$$\begin{aligned} \tilde{\mathbf{h}} &= \arg \min_{\mathbf{h} \in [-1,+1]^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - \boldsymbol{\theta}^T\mathbf{h} + \frac{\lambda}{2}(K - \mathbf{h}^T\mathbf{h}) \right\} \\ &= \arg \min_{\mathbf{h} \in [-1,+1]^K} \left\{ \frac{1}{2}\mathbf{h}^T(\mathbf{R} - \lambda\mathbf{I})\mathbf{h} - \boldsymbol{\theta}^T\mathbf{h} \right\}, \end{aligned} \tag{4.12}$$

where $\lambda > 0$. The solution in (4.12) can be considered as a box-constrained solution to the equation

$$(\mathbf{R} - \lambda\mathbf{I})\mathbf{h} = \boldsymbol{\theta}. \tag{4.13}$$

The matrix $\mathbf{R}$ in (4.9) is now replaced by the matrix $(\mathbf{R} - \lambda\mathbf{I})$, which implies that the joint box-constraint and regularization with negative diagonal loading are used. The regularization parameter λ is chosen as in the MMSE detector, *i.e.* $\lambda = \sigma^2$. The final solution $\hat{\mathbf{h}}$ is obtained by projecting the vector $\tilde{\mathbf{h}}$ to the binary set as $\hat{\mathbf{h}} = \text{sign}[\tilde{\mathbf{h}}]$. The solution to (4.13)

can be found either by using nonlinear GS iterations as in [89] or by using the DCD algorithm with the matrix $\mathbf{R}$ replaced by $\mathbf{R} - \lambda\mathbf{I}$. The GS iterations with box-constrained and negative diagonal loading (GS-BN) detector uses the orthogonal projection operation to perform the semi-hard update. In the DCD box constrained detector, the projection is not used. Instead, before deciding whether an iteration is "successful" or "unsuccessful", the box-constraint test is applied to a possible solution (*i.e.* pre-update of $\mathbf{h}$). If the possible solution is outside the box, the iteration is considered as "unsuccessful" and the update is cancelled. Consequently, we obtain the GS-BN or DCD box-constrained negative loading (DCD-BN) detectors with improved performance compared to the box-constrained multiuser detectors.

Now we take the Tikhonov iterations into account. At zero Tikhonov iteration, we solve the optimization problem (4.12) that corresponds to the GS-BN or DCD-BN detectors. In further Tikhonov iterations, we find the solution of the optimization problem

$$\tilde{\mathbf{h}}^{(n)} = \arg \min_{\tilde{\mathbf{h}} \in [-1,+1]^K} \left\{ \frac{1}{2}\mathbf{h}^T\mathbf{R}\mathbf{h} - (\boldsymbol{\theta} + \lambda_n \tilde{\mathbf{h}}^{(n-1)})^T\mathbf{h} - \frac{\lambda_n}{2}\mathbf{h}^T\mathbf{h} \right\}, \tag{4.14}$$

which can be considered as a box-constrained solution of the equation

$$(\mathbf{R} - \lambda_n\mathbf{I})\tilde{\mathbf{h}}^{(n)} = \boldsymbol{\theta} + \lambda_n\tilde{\mathbf{h}}^{(n-1)}, \quad n = 1, \cdots, N. \tag{4.15}$$

The regularization parameter λ_n is determined as in equation (4.6). The final solution is obtained by using the hard decision $\hat{\mathbf{h}} = \text{sign}[\tilde{\mathbf{h}}^{(N)}]$. The equation (4.15) can be solved either by the GS iterations or the DCD algorithm. Consequently, we obtain the GS-Boxed constraint with negative diagonal loading and Tikhonov iterations (GS-BTN) or DCD-Boxed constraint with negative diagonal loading and Tikhonov iterations (DCD-BTN) detectors.

So far, it was assumed that the box-constrained DCD algorithm is implemented with zero-initialization of the solution vector $\mathbf{h}$ and an initialization of the residual vector as $\mathbf{r} = \boldsymbol{\theta}$. However, a non-zero initialization vector $\mathbf{h}$ can also be used, for which the initialization of the residual vector becomes as $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}$. If the initialization vector $\mathbf{h}$ is close to the final solution, the number of "successful" updates in the box-constrained DCD algorithm can be reduced, resulting in a reduction in the detection complexity. For the proposed DCD-BTN detector, and after each Tikhonov iteration, the accuracy of the solution is improved in general, and so, it can be used for initializing the vector $\mathbf{h}$ in the following Tikhonov iteration. It is expected that with such initialization, the complexity of the box-constrained DCD algorithm decreases as n increases. However, this also requires extra

$2NK^2$ FLOPS for initialization of the residual vector in the N Tikhonov iterations, which can be substantial for a large N. To avoid this complication, we first use zero initialization for $\mathbf{h}$ (*i.e.* $\boldsymbol{\theta}$ initialization for $\mathbf{r}$) in the box-constrained DCD algorithm, where the solution $\tilde{\mathbf{h}}^{(0)}$ and the residual vector $\mathbf{r}^{(0)}$ will be used later in the Tikhonov iterations of the DCD-BTN algorithm. The proposed DCD-BTN detector can be implemented according to the following steps:

1. Solve the equation $(\mathbf{R} - \lambda_0\mathbf{I})\mathbf{h} = \boldsymbol{\theta}$ by using the box-constrained DCD algorithm with the zero-initialization of $\mathbf{h}$ and matrix $\mathbf{R}$ is replaced by $\mathbf{R} - \lambda_0\mathbf{I}$, *i.e.* using DCD-B$(\mathbf{0}, \mathbf{R} - \lambda_0\mathbf{I}, \boldsymbol{\theta}, H, N_u, M_b)$, to obtain a solution vector $\tilde{\mathbf{h}}^{(0)}$ and a residual vector $\mathbf{r}^{(0)}$.

2. For $n = 1, \cdots, N$, solve (4.15) by using the box-constrained DCD algorithm with $\tilde{\mathbf{h}}^{(n-1)}$ as an initialization of the solution and

$$\mathbf{r}^{(n)} = \begin{cases} \mathbf{r}^{(0)} + \lambda_0\tilde{\mathbf{h}}^{(0)} & : \quad \text{if} \quad n = 1 \\ \mathbf{r}^{(n-1)} - \lambda_{n-1}\tilde{\mathbf{h}}^{(n-2)} + (2\lambda_n - \lambda_{n-1})\tilde{\mathbf{h}}^{(n-1)} & : \quad \text{if} \quad n \geq 2, \end{cases} \tag{4.16}$$

   as an initialization of the residual vector and matrix $\mathbf{R}$ is replaced by $\mathbf{R} - \lambda_n\mathbf{I}$, *i.e.* using DCD-B$(\tilde{\mathbf{h}}^{(n-1)}, \mathbf{R} - \lambda_n\mathbf{I}, \mathbf{r}^{(n)}, H, N_u, M_b)$, to obtain a solution vector $\tilde{\mathbf{h}}^{(n)}$.

3. Apply the hard decision to $\tilde{\mathbf{h}}^{(N)}$ to obtain the final solution $\hat{\mathbf{h}} = \text{sign}[\tilde{\mathbf{h}}^{(N)}]$.

4.6 Simplified Implementation of DCD-BTN Multiuser Detector

In the DCD-BTN detector, at the zero Tikhonov iteration, we solve the equation $(\mathbf{R} - \lambda_0\mathbf{I})\mathbf{h} = \boldsymbol{\theta}$ with respect to $\mathbf{h}$ by using the DCD algorithm with zero-initialization of the solution vector and initialization of the residual vector $\mathbf{r}$ by the vector $\boldsymbol{\theta}$. The DCD algorithm provides a solution vector $\hat{\mathbf{h}}^{(0)}$ and the residual vector

$$\mathbf{r}^{(0)} = \boldsymbol{\theta} - (\mathbf{R} - \lambda_0\mathbf{I})\hat{\mathbf{h}}^{(0)}. \tag{4.17}$$

If at the first Tikhonov iteration, the vector $\hat{\mathbf{h}}^{(0)}$ is used for initialization the solution vector of the equation $(\mathbf{R} - \lambda_1\mathbf{I})\mathbf{h} = \boldsymbol{\theta} + \lambda_1\hat{\mathbf{h}}^{(0)}$, the residual vector $\mathbf{r}$ should be initialized by the vector

$$\boldsymbol{\theta}^{(1)} = (\boldsymbol{\theta} + \lambda_1\hat{\mathbf{h}}^{(0)}) - (\mathbf{R} - \lambda_1\mathbf{I})\hat{\mathbf{h}}^{(0)} = \mathbf{r}^{(0)} + \lambda_0\hat{\mathbf{h}}^{(0)}\,, \tag{4.18}$$

where the last equality is due to (4.17). The DCD algorithm produces a solution vector $\mathbf{h}^{(1)}$ and the residual vector

$$\mathbf{r}^{(1)} = \boldsymbol{\theta} + \lambda_1 \hat{\mathbf{h}}^{(0)} - (\mathbf{R} - \lambda_1 \mathbf{I})\hat{\mathbf{h}}^{(1)}. \tag{4.19}$$

At the 2nd Tikhonov iteration, we solve the problem $(\mathbf{R} - \lambda_2 \mathbf{I})\mathbf{h} = \boldsymbol{\theta} + \lambda_2 \hat{\mathbf{h}}^{(1)}$. For initializing the solution of this equation by the vector $\hat{\mathbf{h}}^{(1)}$, we should also initialize the residual vector $\mathbf{r}$ by the vector

$$\begin{aligned} \boldsymbol{\theta}^{(2)} &= (\boldsymbol{\theta} + \lambda_2 \mathbf{h}^{(1)}) - (\mathbf{R} - \lambda_2 \mathbf{I})\mathbf{h}^{(1)} \\ &= \mathbf{r}^{(1)} - \lambda_1 \hat{\mathbf{h}}^{(0)} + (2\lambda_2 - \lambda_1)\hat{\mathbf{h}}^{(1)}, \end{aligned} \tag{4.20}$$

where the last equality is due to (4.19). The DCD algorithm produces a solution vector $\hat{\mathbf{h}}^{(2)}$ and the residual vector $\mathbf{r}^{(2)} = \boldsymbol{\theta} + \lambda_2 \mathbf{h}^{(1)} - (\mathbf{R} - \lambda_2 \mathbf{I})\hat{\mathbf{h}}^{(2)}$. Similarly, at the $(n-1)$th Tikhonov iteration, $n \geq 3$, the DCD algorithm produces a solution vector $\hat{\mathbf{h}}^{(n-1)}$ and the residual vector

$$\mathbf{r}^{(n-1)} = \boldsymbol{\theta} + \lambda_{n-1} \hat{\mathbf{h}}^{(n-2)} - (\mathbf{R} - \lambda_{n-1} \mathbf{I})\hat{\mathbf{h}}^{(n-1)}. \tag{4.21}$$

At the nth Tikhonov iteration, $n \geq 3$, we solve the problem $(\mathbf{R} - \lambda_n \mathbf{I})\mathbf{h} = \boldsymbol{\theta} + \lambda_n \hat{\mathbf{h}}^{(n-1)}$. For initializing the solution by the vector $\hat{\mathbf{h}}^{(n-1)}$, we should also initialize the residual vector $\mathbf{r}$ by the vector

$$\begin{aligned} \boldsymbol{\theta}^{(n)} &= (\boldsymbol{\theta} + \lambda_n \hat{\mathbf{h}}^{(n-1)}) - (\mathbf{R} - \lambda_n \mathbf{I})\hat{\mathbf{h}}^{(n-1)} \\ &= \mathbf{r}^{(n-1)} - \lambda_{n-1} \hat{\mathbf{h}}^{(n-2)} + (2\lambda_n - \lambda_{n-1})\hat{\mathbf{h}}^{(n-1)} \end{aligned} \tag{4.22}$$

where the last equality is due to (4.21). Combining (4.18), (4.20) and (4.22) together, we obtain (4.16).

4.7 Simulation Results for the DCD-BTN Detector

In this section, we present numerical results that show the detection performance and complexity of the DCD-BTN multiuser detector. We use the same simulation environment (AWGN channel), with randomly generated binary spreading codes and highly loaded scenarios as in [15]. For the box-constrained DCD algorithm, we use the maximum number of successful iterations $N_u = 1000$. Although using a smaller N_u can reduce the complexity of the algorithm, we set N_u at a high value to ensure the best detection performance. In Fig.4.1, we study the performance of the DCD-BTN algorithm with respect

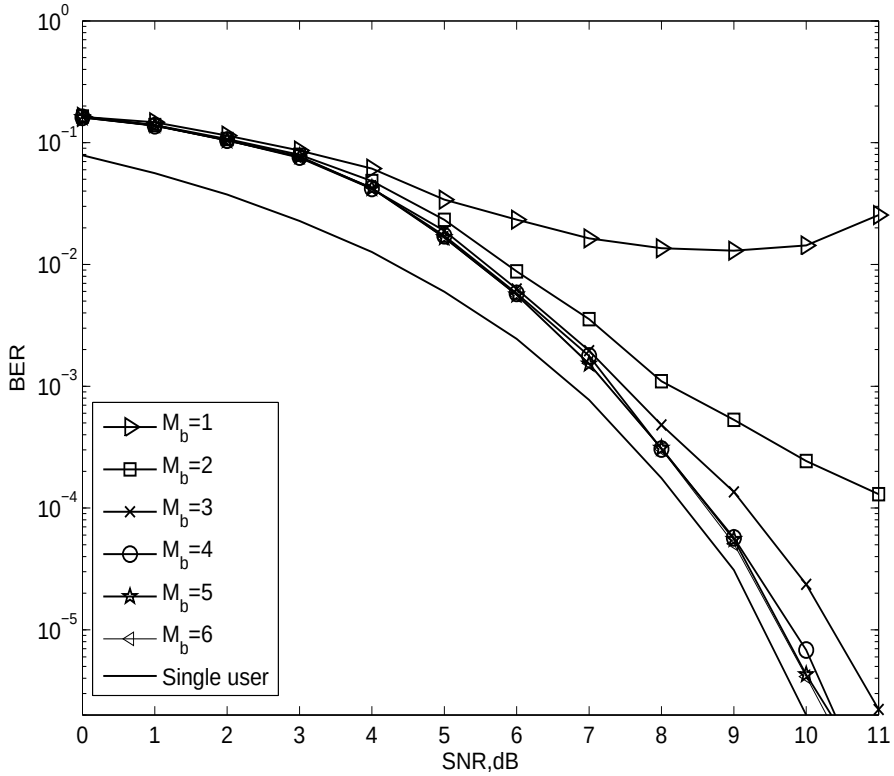


Figure 4.1: BER Performance of DCD-BTN in different M_b; $K = 64$, SF $= 67$ and $N = 5$.

to different numbers of bits M_b in the scenario of the number of user $K = 64$ and the random binary spreading codes SF $= 67$. It shows that the detection BER performance of the DCD-BTN algorithm is significantly improved when $M_b = 3$. The performance of the DCD-BTN algorithm with $M_b = 5$ and $M_b = 6$ is approximately the same. Therefore, we choose $M_b = 6$ for the following simulation scenarios. It is also seen that the detection performance of the DCD-BTN detector is close to the single-user bound.

We further compare the proposed DCD-BTN algorithm with the "efficient frontier" which was presented in [15]. We use a highly loaded scenario (as in [15]), for which $K = 60$ and SF $= 63$. Fig.4.2 shows the group detection error (probability of at least one error among all K user data) as a function of the detector worst-case complexity for SNR $= 10$ dB. The known BB, PDA, and DF detectors form the "efficient frontier", *i.e.* they provide the best trade-off in terms of the complexity and detection performance. The Direct-T detector exploits the nonstationary Tikhonov iterative regularization as described in [94]. The case of $N = 0$ corresponds to the MMSE detector whose performance is poor. As N increases, the detection performance significantly improves, but is still worse than that of the BB detector. The performance improvement with respect to the MMSE detector is achieved at the expense of an increase in the complexity. We have run simulations

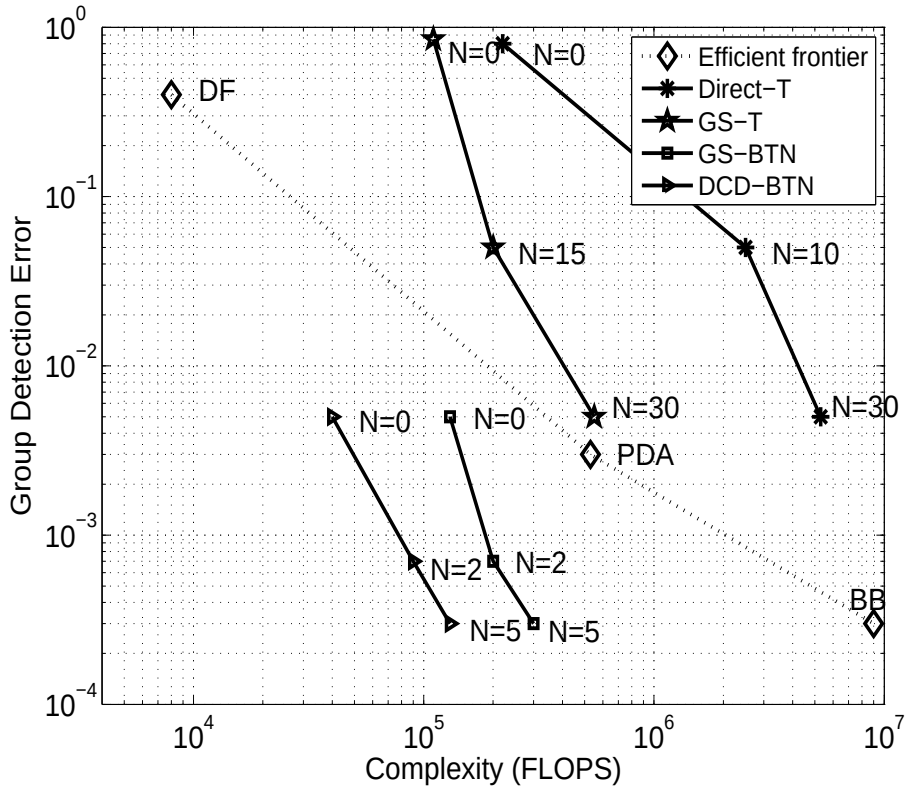


Figure 4.2: Group detection error vs worst-case complexity; $K = 60$, SF $= 63$, SNR $= 10$ dB.

for high N beyond 30, but the performance is not be improved, which is still worse than that of the BB detector. The results marked 'GS-T' are related to the GS implementation of the detector using Tikhonov iterative regularization as described in [96]. Within each Tikhonov iteration, GS iterations reduce the complexity without degrading the detection performance. However, even with GS iterations, the detector cannot improve the "efficient frontier".

The GS-BTN detector that uses GS iterations to implement Tikhonov regularization with negative diagonal loading and box-constraint, outperforms the GS-T detector in terms of the group detection error. When $N = 5$, the GS-BTN detector achieves the same performance of the BB detector, which also presents the optimal ML detection performance.

Compared to the Direct-T and GS-T detectors, the GS-BTN detector allows a significant improvement in the detection performance. It also allows a reduction in the required number of Tikhonov iterations (to achieve the best reached performance) from $N = 30$ to $N = 5$, which results in a lower complexity. The complexity of the GS-BTN detector is as low as approximately $K^{3.2}$ FLOPS. (When we run the simulations, we also calculated the corresponding complexity.)

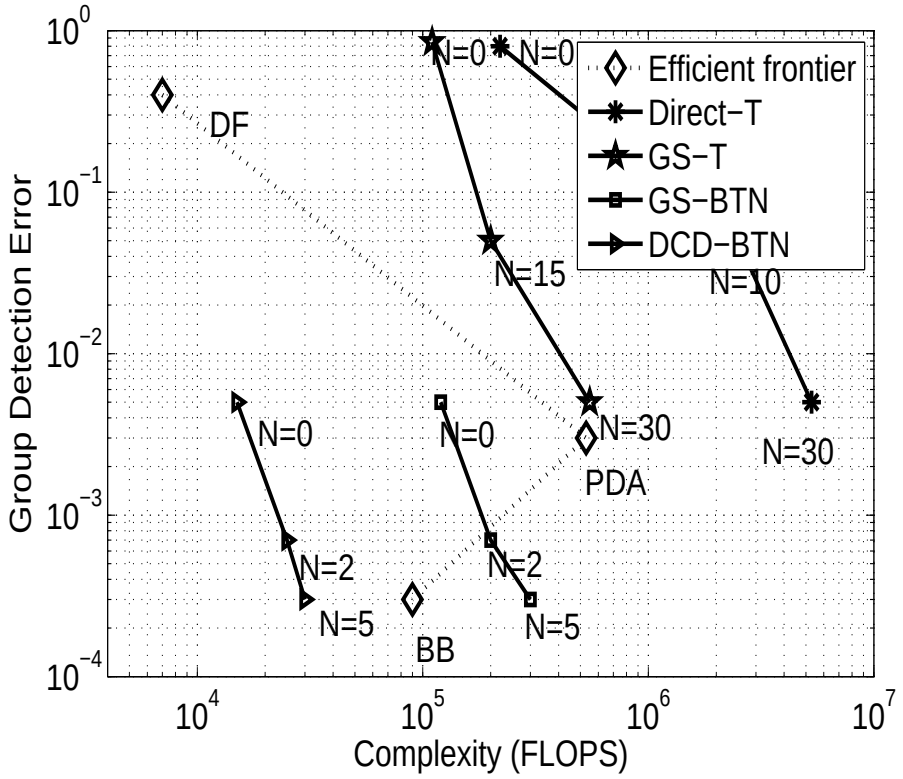


Figure 4.3: Group detection error vs average complexity; $K = 60$, SF $= 63$, SNR $= 10$ dB.

The DCD-BTN detector also achieves the lowest group detection error after $N = 5$ Tikhonov iterations. Its performance is similar to that of the GS-BTN detector. However, the worst-case complexity of the DCD-BTN detector with $N = 5$ Tikhonov iterations is only $K^{2.8}$ FLOPS, which is lower than that of the GS-BTN detector. Thus, the DCD-BTN detector allows the reduction of the worst-case complexity, compared to the GS-BTN detector without degrading the detection performance.

Fig.4.3 shows the group detection error as a function of the average complexity for the number of user $K = 60$, the random binary spreading codes SF $= 63$ and SNR $= 10$ dB. The DCD-BTN detector has a lower average complexity than the other detectors. The only exception is the DF detector, but its detection performance is poor. Moreover, the DF detector requires computing the Cholesky decomposition of the matrix $\mathbf{R}$ (not taken into account in Fig.4.2 and Fig.4.3). For $N = 5$, the DCD-BTN detector achieves the same detection performance as the BB detector, *i.e.* it achieves the performance of the optimal ML detector. In this case, the average complexity of the DCD-BTN detector is approximately $K^{2.5}$ FLOPS, which is lower than that of the BB detector.

Fig.4.4 demonstrates the group detection error of the BB and DCD-BTN detectors as

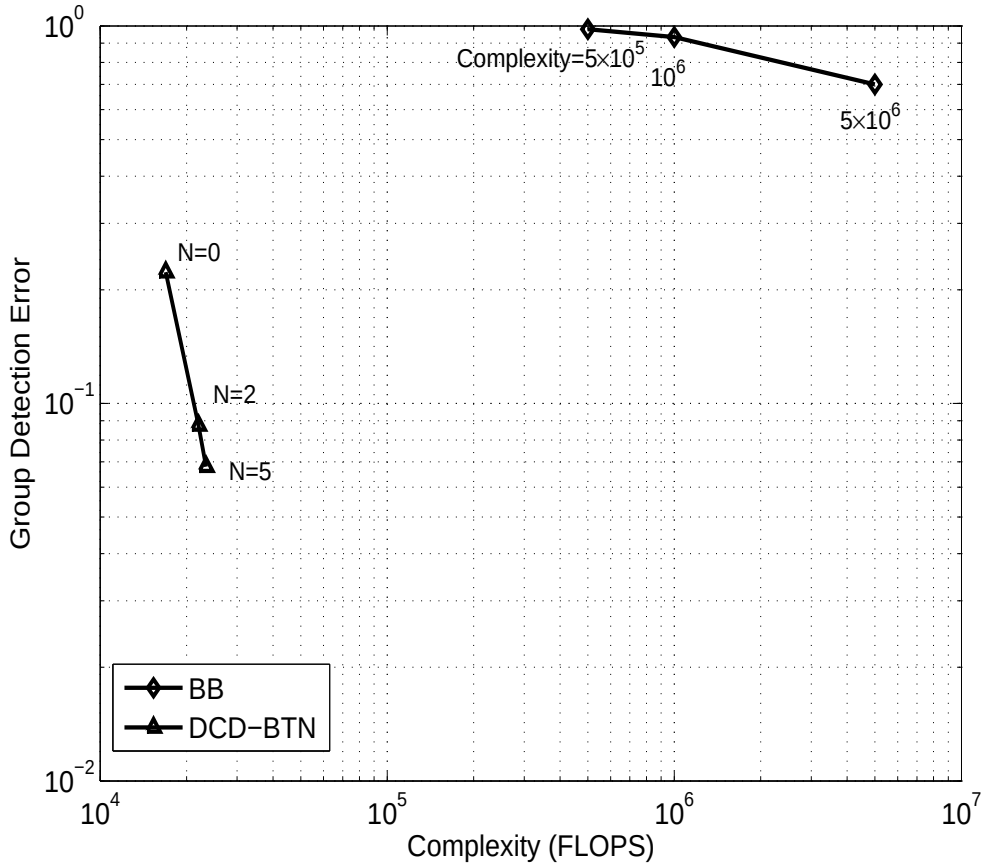


Figure 4.4: Group detection error vs. worst-case complexity complexity; $K = 60$, SF $= 63$, SNR $= 7$ dB.

a function of the worst-case complexity for the number of user $K = 60$, the random binary spreading codes SF $= 63$ and SNR $= 7$ dB. Simulation results are shown for a BB detector with upper bounded complexity for several complexity bounds (5×10^5, 10^6 and 5×10^6). This is due to the high worst-case complexity of the BB detector for low SNRs. The detection performance of the DCD-BTN is significantly better than that of the BB detector. In addition, the complexity of the DCD-BTN detector is significantly lower than that of the BB detector.

Fig.4.5 use the computer simulation to show the BER performance of the detectors DF, PDA and DCD-BTN with $N = 5$ compared to that of the single-user bound as a function of SNR for the number of user $K = 60$ and the random binary spreading codes SF $= 63$. It can be seen that the DCD-BTN detector outperforms the other detectors, especially at high SNRs. In this case, its BER performance is close to that of the single user bound.

The DCD-BTN multiuser detector has the lowest complexity among detectors providing the ML performance. It significantly improves the "efficient frontier" in multiuser

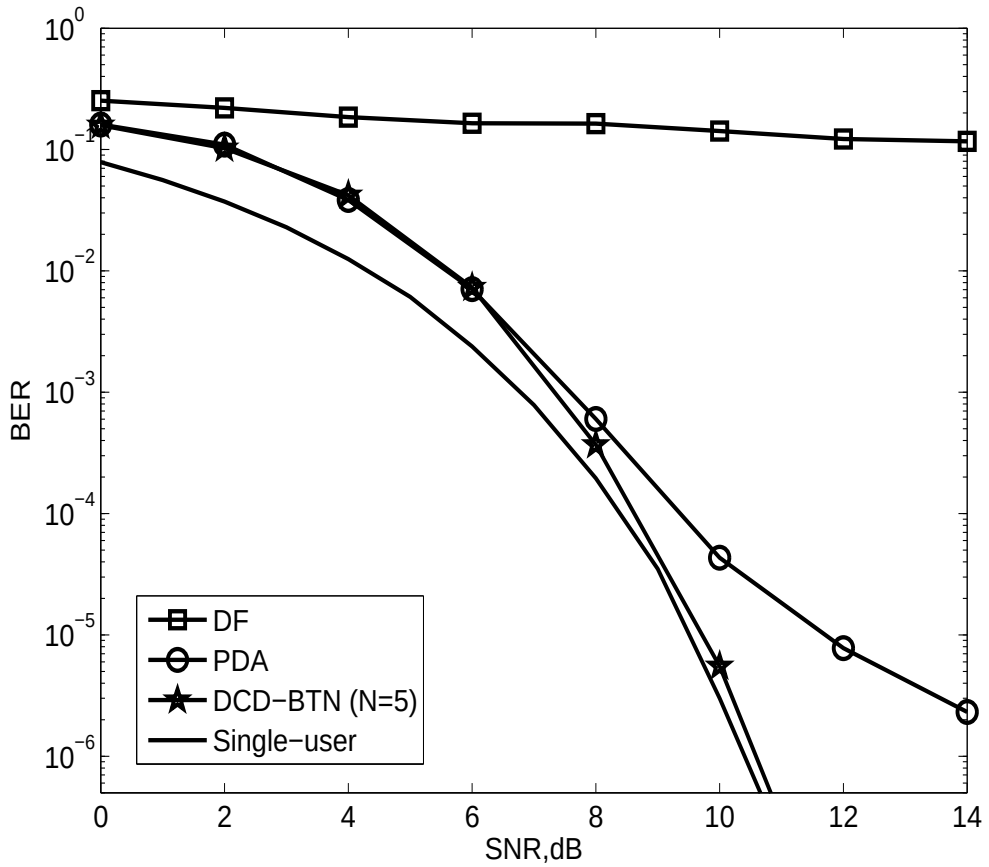


Figure 4.5: BER performance of the DF, PDA, and DCD-BTN detectors against single-user bound; $K = 60$, SF $= 63$.

detection. An extra benefit of the proposed DCD-BTN detector is that it is essentially multiplication-free and division-free. Multiplications are only required for calculation of the vector $\mathbf{r}^{(n)}$ and diagonal elements of the matrix in (4.15). Most operations in the detector are additions and bit-shifts. This makes the proposed technique attractive for fixed-point hardware implementation.

4.8 Hardware Architecture of the DCD-BTN Algorithm

Fig.4.6 depicts the top level block diagram for the DCD-BTN algorithm. This block diagram is a general architecture representing the data dependency and data flow between sub-blocks. The Master State Machine is in charge of initialising DCD-BTN transactions and controls the operation of other modules, *i.e.* the destination of the modules' input stream and the source of its output stream. The Transceiver is responsible for the data flow between the FPGA board and the MATLAB end. Tikhonov Iteration Module computes

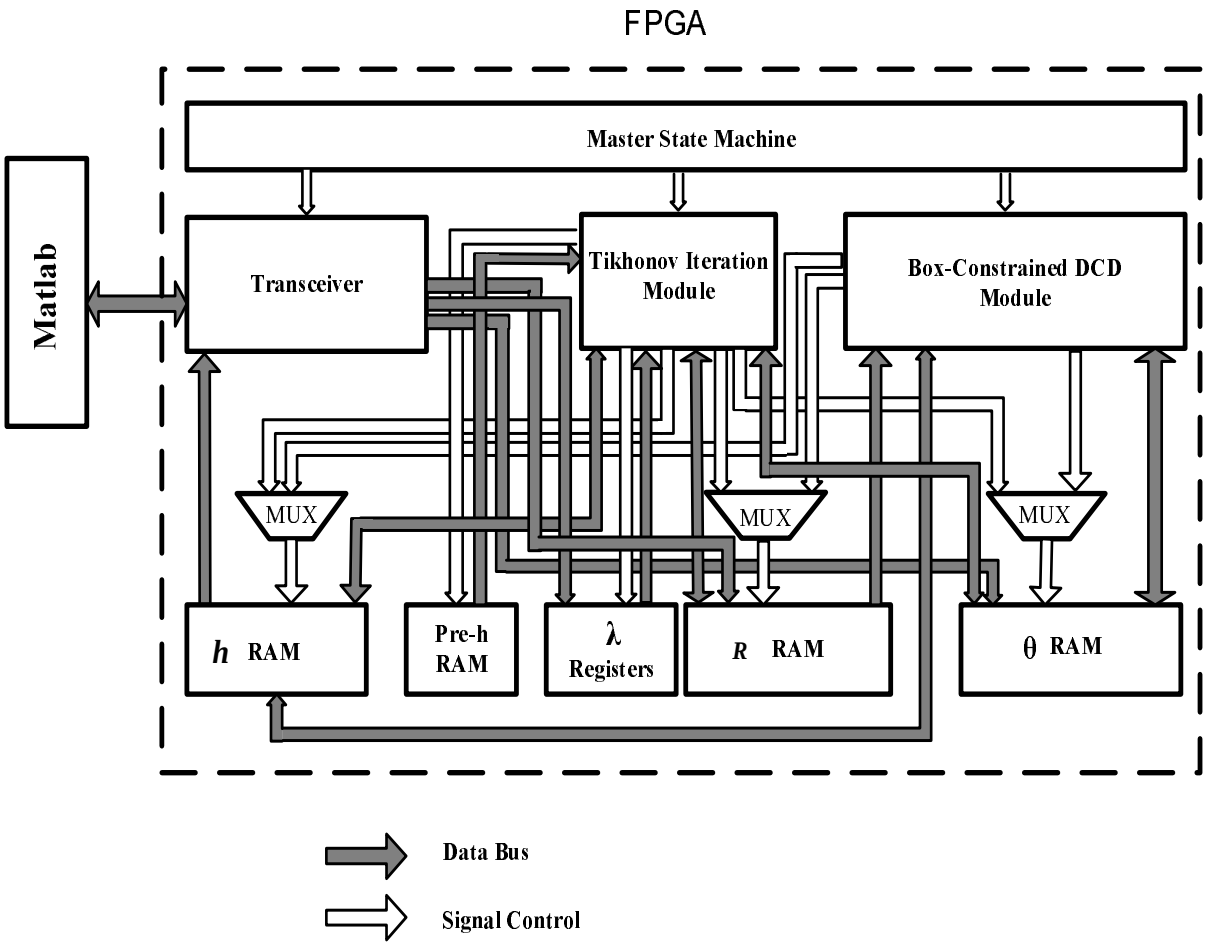


Figure 4.6: Block-diagram of the FPGA design of the DCD-BTN algorithm.

the vector $\mathbf{r}$ in (4.16) and introduces the negative diagonal loading of $\mathbf{R}$ (*i.e.* $\mathbf{R} - \lambda_n \mathbf{I}$). The box-constrained DCD module solves the equation (4.15) with zero initialization of $\mathbf{h}$. All other necessary parameters (*i.e.* $\mathbf{R}$, $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$) are received from a MATLAB software and stored in an Input RAM, which is not shown in Fig.4.6 and is located in the Transceiver. This Transceiver then sends these parameters to the corresponding $\mathbf{R}$, $\boldsymbol{\theta}$ RAMs and $\boldsymbol{\lambda}$ Registers. The multiplexers (MUX) select the module (Tikhonov Iteration or Box-constrained DCD) to which the elements in $\mathbf{h}$, $\boldsymbol{\theta}$ and $\mathbf{R}$ RAMs are connected. After the nth Tikhonov iteration, a solution $\mathbf{h}^{(n)}$ is obtained and copied to a single-port **Pre-h** RAM which is used for the $(n+1)$th Tikhonov iteration. Elements in the matrix $\mathbf{R}$ and vector $\mathbf{h}$ are 16-bit data width, whereas the elements in the vector $\boldsymbol{\theta}$ are 32-bit data width.

4.8.1 Tikhonov Iteration Module

The Tikhonov Iteration Module includes $\mathbf{r}$ Update Module, which performs the residual vector $\mathbf{r}$ updating, and Deregularization Module, which is used for obtaining $(\mathbf{R} - \lambda_n \mathbf{I})$ at the nth Tikhonov iteration.

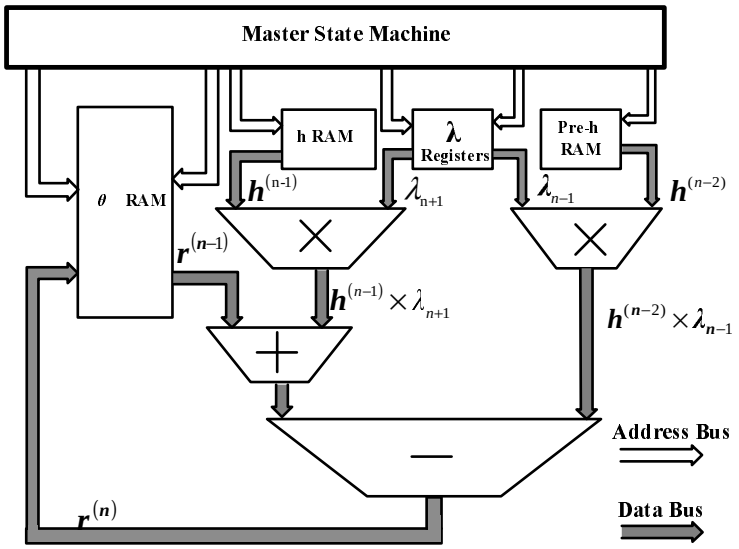


Figure 4.7: Architecture of the $\mathbf{r}$ Update Module.

r Update Module: $\mathbf{r}$ Update Module updates the residual vector $\mathbf{r}$ as given in (4.16) when the number of Tikhonov iterations $n \geq 1$. The architecture block diagram of this

module is shown in Fig.4.7. This module consists of two adders/substractors and two real multipliers. The module deals with the three RAMs ($\boldsymbol{\theta}, \mathbf{h}$, pre-h) and $\boldsymbol{\lambda}$ registers. The Master State Machine controls addressing, operands to adder/subtractor, *i.e.* the Master State Machine selects the element r in $\boldsymbol{\theta}$ RAM, chooses the regularization parameter $\boldsymbol{\lambda}$ and selects element h in $\mathbf{h}$ RAM. The pipelining technique allows the residual vector $\mathbf{r}$ to be updated using K cycles. Besides, 3 cycles of latency for RAM reading and 2 cycles of latency for multiplication are required. Therefore in total, the $\mathbf{r}$ update requires $K + 5$ cycles.

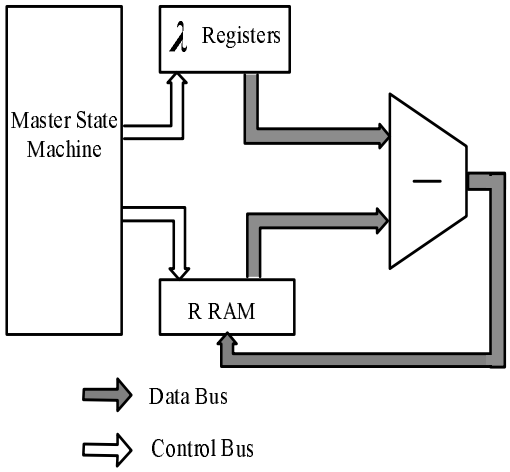


Figure 4.8: Architecture of the Deregularization Module.

Deregularization Module: Deregularization Module updates the diagonal elements in matrix $\mathbf{R}$ as shown in (4.15). The architecture diagram is presented in Fig.4.8. This module contains a subtractor and deals with $\mathbf{R}$ RAM and $\boldsymbol{\lambda}$ registers. The operation of $(\mathbf{R} - \lambda_n \cdot \mathbf{I})$ updates the diagonal elements of matrix $\mathbf{R}$ only by using a subtractor. This module requires K cycles for regularization. Thus, in total $K + 3$ cycles are required including 3 latency clock cycles. Moreover, when the system size is small, *e.g.* 4×4, the diagonal elements of the matrix $\mathbf{R}$ can be stored in registers, and so, the $\mathbf{R}$ deregularization can be realized in one clock cycle.

4.8.2 Box-constrained DCD Module

The FPGA architecture of the box-constrained DCD algorithm has been presented in Chapter 2. The Master State Machine controls the number of times this module is in

use. Before the Tikhonov iterations start, the box-constrained DCD module is used first to obtain the solution $\tilde{\mathbf{h}}^{(0)}$ and the residual vector $\mathbf{r}^{(0)}$ (corresponding to step 1) using the parameters stored in $\mathbf{h}$, $\boldsymbol{\theta}$ and $\mathbf{R}$ RAMs. Initially, $\mathbf{h}$ RAM is set to zero, $\mathbf{R}$ and $\boldsymbol{\theta}$ RAMs get the $\mathbf{R}$ and $\boldsymbol{\theta}$ values from Input RAM in the Transceiver. However, the $\mathbf{R}$ RAM needs to be updated as $\mathbf{R} - \lambda_0\mathbf{I}$, using the Deregularization Module. After that, the box-constrained DCD is used again for N times in cooperation with the Tikhonov Iteration Module.

4.8.3 Numerical Results for the DCD-BTN Detector FPGA Implementation

In this section, we present the detection performance and complexity of the fixed-point DCD-BTN detector implemented on the FPGA (XC2VP30) platform.

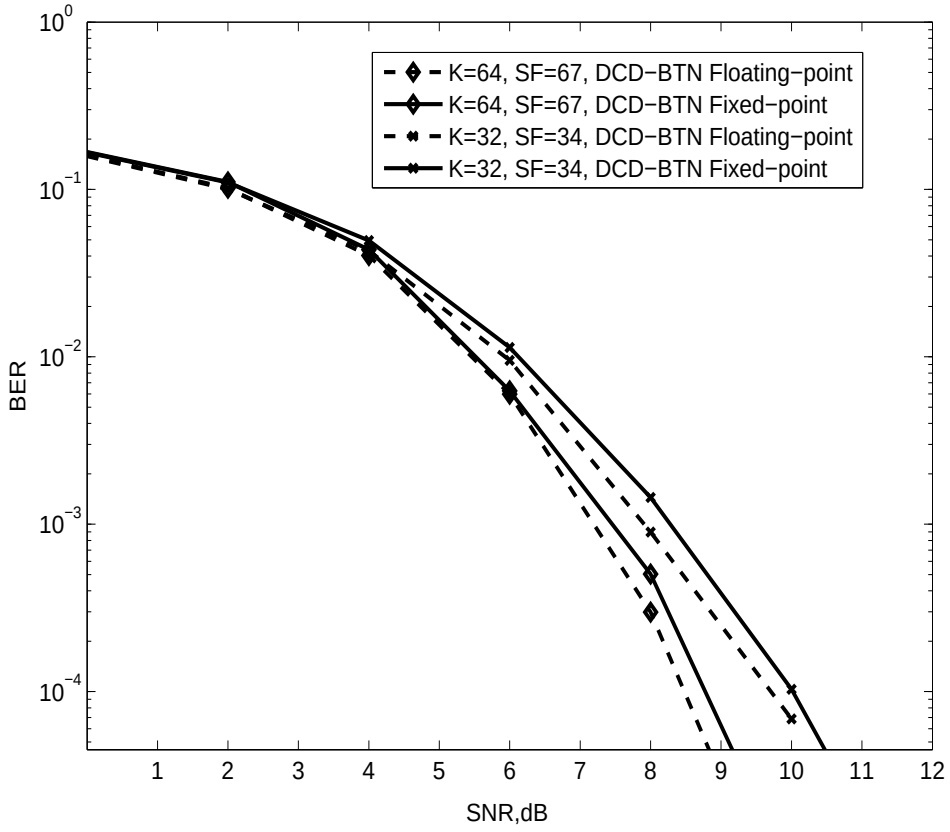
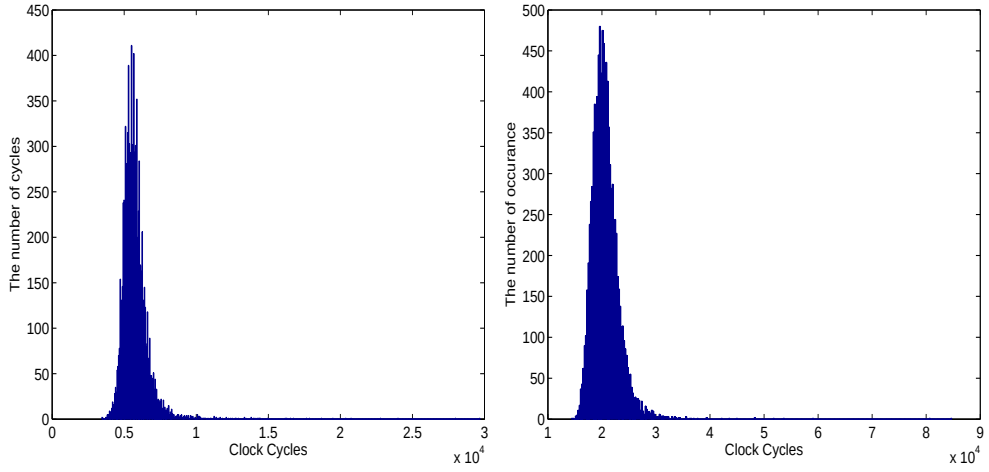


Figure 4.9: BER performance between DCD-BTN floating-point and fixed-point; $K = 64$, SF $= 67$, $N = 5$ and $M_b = 6$.



(a) (b)

Figure 4.10: Clock cycles distribution for fixed-point DCD-BTN detector implementation: (a) $K = 32$, SF $= 34$; (b) $K = 64$, SF $= 67$, $M_b = 6$; SNR $= 7$ dB.

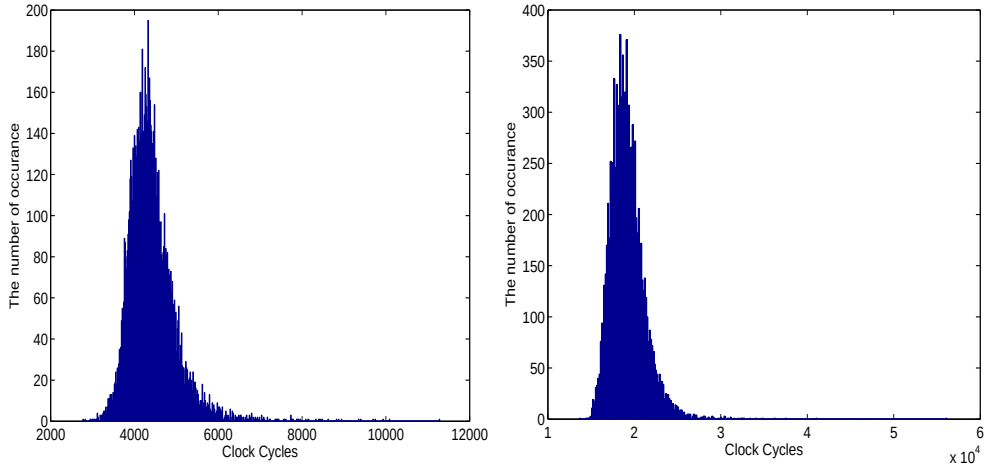
Fig.4.9 compares the BER performance of the fixed-point DCD-BTN implementations against their floating-point implementations, obtained in 10^4 simulation trails. The number of users is $K = 32, 64$, and the spreading factor is SF $= 34, 67$, the number of bits is $M_b = 6$ and the number of Tikhonov iterations is $N = 5$. It shows that the two implementations have similar detection performance.

Fig.4.10(a), (b) displays the clock cycles distribution for fixed-point DCD-BTN detector implementation in the scenarios of $K = 32$, SF $= 34$ and $M_b = 6$, and $K = 64$, SF $= 67$ and $M_b = 6$, respectively. Fig.4.11(a), (b) displays the clock cycles distribution for fixed-point DCD-BTN detector implementation in the scenarios of, $K = 32$, SF $= 34$, $M_b = 4$ and $K = 64$, SF $= 67$, $M_b = 4$, respectively. According to the figures, we can estimate the average and worst-case number of clock cycles for each scenario.

Table 4.1: The clock cycles required for the DCD-BTN detector with different number of bits for worst and average cases; $K = 64$, SF $= 67$, $N = 5$ and SNR $= 7$ dB.

M_b	1bit	2bits	3bits	4bits	5bits	6bits
Worst(α)	14912 (2.31)	23616(2.42)	32448(2.49)	57008(2.63)	73600(2.69)	86208(2.73)
Average(α)	10314(2.22)	14293(2.30)	17404(2.34)	20748(2.39)	24319(2.42)	28032(2.46)

Table 4.1 shows the required clock cycles for the DCD-BTN detector with different number of bits M_b for $K = 64$, SF $= 67$, SNR $= 7$ dB, in the worst-case and average-case scenarios. For simplicity, the required clock cycles are represented as K^{α}, where α is an exponent of K. The results show that the required clock cycles in the detector is maxi-



(a) (b)

Figure 4.11: Clock cycles distribution for fixed-point DCD-BTN detector implementation: (a) $K = 32$, SF $= 34$, $M_b = 4$; (b) $K = 64$, SF $= 67, M_b = 4$; SNR $= 7$ dB.

mally (*i.e.* for $M_b = 6$) $K^{2.46}$ for the average-case and $K^{2.73}$ for the worst-case.

Table 4.2: FPGA resources and update rates for DCD-BTN detector; $M_b = 6$ and $N = 5$

System Size	$K = 4$	$K = 32$	$K = 64$
Slices	973	1162	1233(9%)
Block RAM	8	10	18
Multiplier	3	3	3
Update Rate(kHz)	28	12.5	3

Table 4.2 compares the FPGA resources required for the design and update rates for the DCD-BTN detector for $K = 4$, SF $= 4$, $K = 32$, SF $= 34$ and $K = 64$, SF $= 67$ with $M_b = 6$ and $N = 5$. The area usage of the Transceiver module is not included in these figures. Results show that this design requires at most 9% of the resources available on the FPGA chip. The increase of K has no much effect on the slice count, but affects the size of memory. The FPGA resources increase slightly when the system size increases from $K = 4$ to $K = 64$. At the same time, the update throughput slows down when K is increased. To improve the update throughput, we can implement the box-constrained DCD algorithm in the parallel architecture (as described in chapter 2) instead of the serial architecture.

4.9 Conclusions

A known iterative multiuser detection technique based on box-constrained solution with Gauss-Seidel iterations and nonstationary iterative deregularization (GS-BD) provides a better trade-off between complexity and detection performance than the "efficient frontier", which is formed by the decision-feedback (DF), probabilistic data association (PDA), and the branch and bound (BB) detectors. Using DCD iterations instead of the Gauss-Seidel iterations, we have proposed the DCD-boxed constraint with negative diagonal loading and Tikhonov iterations (DCD-BTN) multiuser detector.

The performance of the DCD-BTN detector, in terms of group detection error and complexity, has been compared with that of the existing advanced multiuser detection techniques, which are the DF, BB and PDA detectors. The DCD-BTN detector shows the lowest complexity among these detectors (except for the DF detector, which has poor detection performance) while providing almost an ML-like performance. Most operations in the DCD-BTN detector can be implemented using additions and bit-shifts. This makes it easy to be implemented in the real-time hardware. The multiplications are only required for the calculation of the residual vector $\mathbf{r}$ update. However, this involves a very small number of multiplications.

This design has been implemented on an FPGA board. This design only requires 1233 slices when $K = 64$, SF $= 67$, $M_b = 6$ and $N = 5$. We have compared the fixed-point FPGA DCD-BTN detection performance with its floating point detection performance. The results have shown that the two BER performance curves are very close to each other. The DCD-BTN detector considered in this chapter allows detection of BPSK symbols. A multiuser detector using the similar approach, but also suitable for detection of M-PSK symbols, is considered in the next chapter.

Chapter 5

The DCD-BTN-M Detector for M-PSK Symbols

Contents

5.1 Introduction

The DCD-BTN detector has shown a detection performance close to the single user bound for BPSK symbols in chapter 4. In this chapter, we extend the DCD-BTN algorithm by making some modifications and propose a DCD-BTN-M detector. The proposed detector is designed for M-PSK symbols with $M > 2$. Numerical results show that the DCD-BTN-M detector provides a good detection performance with a reasonable complexity for M-PSK symbols.

This chapter is organised as follows. In Section 5.2, the channel model for M-PSK multiuser detection is presented. In Section 5.3, the DCD-BTN-M detector is described. The

numerical results for the DCD-BTN-M detector are shown in Section 5.4. Section 5.5 concludes the chapter.

5.2 Problem Formulation

For M-PSK symbols, the optimal ML multiuser detector estimates the vector $\mathbf{h}$ by minimizing the quadratic function

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in \mathcal{A}^K} \left\{ \frac{1}{2} \mathbf{h}^T \mathbf{R} \mathbf{h} - \boldsymbol{\theta}^T \mathbf{h} \right\}, \tag{5.1}$$

where $\mathbf{h}$, $\boldsymbol{\theta}$ are $K \times 1$ vectors, $\mathbf{R}$ is a $K \times K$ symmetric positive definite matrix. Alphabet $\mathcal{A}$ stands for M-PSK constellation. To solve this problem, by using box-constraint and the negative loading, and with the Tikhonov iterations, we can find the solution of the optimization problem

$$\tilde{\mathbf{h}}^{(n)} = \arg \min_{|h_k|<1, k=1,\cdots,K} \left\{ \frac{1}{2} \mathbf{h}^T \mathbf{R} \mathbf{h} - (\boldsymbol{\theta} + \lambda_n \tilde{\mathbf{h}}^{(n-1)})^T \mathbf{h} - \frac{\lambda_n}{2} \mathbf{h}^T \mathbf{h} \right\}, \tag{5.2}$$

which can be considered as a box-constrained solution of the equation

$$(\mathbf{R} - \lambda_n \mathbf{I}) \tilde{\mathbf{h}}^{(n)} = \boldsymbol{\theta} + \lambda_n \tilde{\mathbf{h}}^{(n-1)}, \quad n = 1, \cdots, N. \tag{5.3}$$

where N is the number of the Tikhonov iterations, λ_n is the diagonal loading regularization parameter.

5.3 DCD Box-constrained Detector with Negative Diagonal Loading and Tikhonov Iterations for M-PSK Symbols

The DCD-BTN detector has shown an outstanding detection performance for BPSK symbols. In this section, we further exploit the DCD-BTN detector, and make some refinements to enable it dealing with complex-valued symbols. These refinements are described as follows: the regularization parameter λ_n varies with iterations n, constant c and according to

$$\lambda_n = \begin{cases} \sigma^2 & : \quad n = 0 \\ cn\sigma^2 & : \quad n = 1, \cdots, N \,. \end{cases} \tag{5.4}$$

Table 5.1: Complex-valued box-constrained DCD algorithm

State	Operation
0	Initialization: $\mathbf{h} = \bar{\mathbf{h}}$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\bar{\mathbf{h}}$, $m = M_b$, $p = 0$ $\Delta m = 0$, $s = 1$, $j = 1$, Flag$= 0$
1	if $m = 0$, algorithm stops else, $m = m - 1$, $d = 2^m$, $\Delta m = \Delta m + 1$
2	if $s = 1$, then $r_t = \Re(r(j))$; else, $r_t = \Im(r(j))$ $c = R(j,j)/2 - \lvert r_t \rvert \times 2^{\Delta m}$ $h_t = h(j) + \text{sign}(r_t) sd$ if $\lvert h_t \rvert \leq 1$, then $\alpha = 0$; else, $\alpha = 1$
3	if $c < 0$ and $\alpha = 0$, then goto state 4 else, goto state 5
4	$h(j) = h_t$ $\mathbf{r} = \mathbf{r} \times 2^{\Delta m} - \text{sign}(r_t) s \mathbf{R}^{(j)}$ $\Delta m = 0$, $p = p + 1$, Flag = 1 if $p = N_u$, algorithm stops
5	if $s = 1$, then $s = i$, goto state 2 else, $s = 1$, $j = (j) \text{mod}(M_T) + 1$ if $j = 1$ and Flag = 1, then Flag = 0, goto state 2 elseif $j = 1$ and Flag = 0, then goto state 1 else, goto state 2

Table 5.1 shows the complex-valued box-constrained DCD algorithm (DCD-B-CMPLX) applied for this detector. It is similar to the complex-valued DCD algorithm in chapter 3, but in state 2 $|\Re(h_t)| \leq H$ and $|\Im(h_t)| \leq H$ is replaced by $|h_t| \leq 1$. The DCD-B-CMPLX algorithm provides an initial solution; the initial solution is mapped to the constellation $\mathcal{A}$, and then is used in the Tikhonov iterations of the DCD-B-CMPLX algorithm. The proposed detector is named DCD-BTN-M and described as follows.

1. Solve the equation $(\mathbf{R} - \lambda_0 \mathbf{I})\mathbf{h} = \boldsymbol{\theta}$ by using the box-constrained complex valued DCD algorithm with the zero-initialization of $\mathbf{h}$ and matrix $\mathbf{R}$ replaced by $\mathbf{R} - \lambda_0 \mathbf{I}$, *i.e.* we use DCD-B-CMPLX$(\bar{\mathbf{h}}, \mathbf{R}, \boldsymbol{\theta}, H, N_u, M_b)$ to obtain a solution vector $\bar{\mathbf{h}}^{(0)}$.

2. Mapping $\bar{\mathbf{h}}^{(0)}$ into the M-PSK constellation to obtain $\hat{\mathbf{h}}^{(0)}$.

3. For $n = 1, \cdots, N$, the box-constrained complex valued DCD algorithm uses $\tilde{\mathbf{h}}^{(n-1)}$ as an initialization of the solution and

$$\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta} - \lambda_n \tilde{\mathbf{h}}^{(n-1)} \tag{5.5}$$

as an initialization of the residual vector and matrix $\mathbf{R}$ is replaced by $\mathbf{R} - \lambda_n \mathbf{I}$, *i.e.* we use DCD-B-CMPLX$(\tilde{\mathbf{h}}^{(\mathbf{n-1})}, \mathbf{R}, \boldsymbol{\theta}^{(n)}, H, N_u, M_b)$ to obtain a solution vector $\tilde{\mathbf{h}}^{(n)}$.

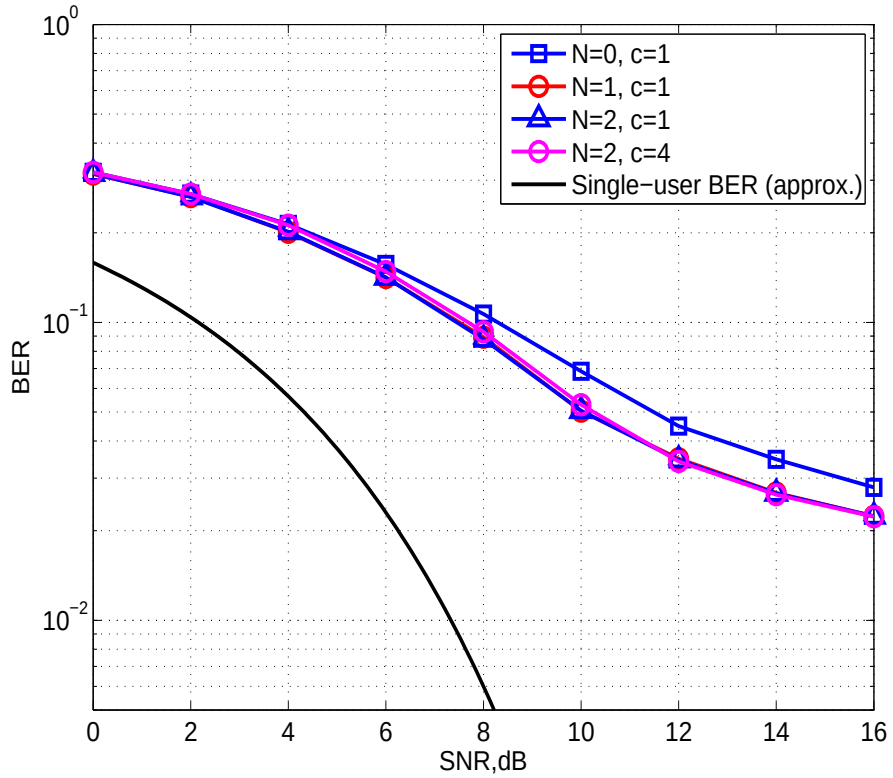


Figure 5.1: BER performance of the DCD-BTN detector in the scenario of $K = 4$, SF $= 4$, $M_b = 15$.

4. Mapping $\tilde{\mathbf{h}}^{(N)}$ into the M-PSK constellation to obtain $\hat{\mathbf{h}}^{(N)}$.

5.4 Simulations for the DCD-BTN-M Detector

In this section, we present computer simulation examples to show the detection performance and the complexity of the proposed multiuser detector for QPSK symbols. We assume the simulation channel is perfect power control with AWGN, where the users employ randomly generated spreading codes. We choose $M_b = 15$ and $N_u = 300$. The Single-user BER performance is obtained by theoretical calculation. Fig. 5.1 shows the BER performance of the DCD-BTN-M detector. The number of users is $K = 4$, the spreading factor is SF $= 4$. It can be seen that the performance is poor. Therefore, the DCD-BTN-M detector is not suited for the small system size.

Fig. 5.2 shows the bit error rate (BER) performance of the proposed multiuser detector in comparison with MMSE detector in the scenario of $K = 40$ and SF $= 63$. It shows that

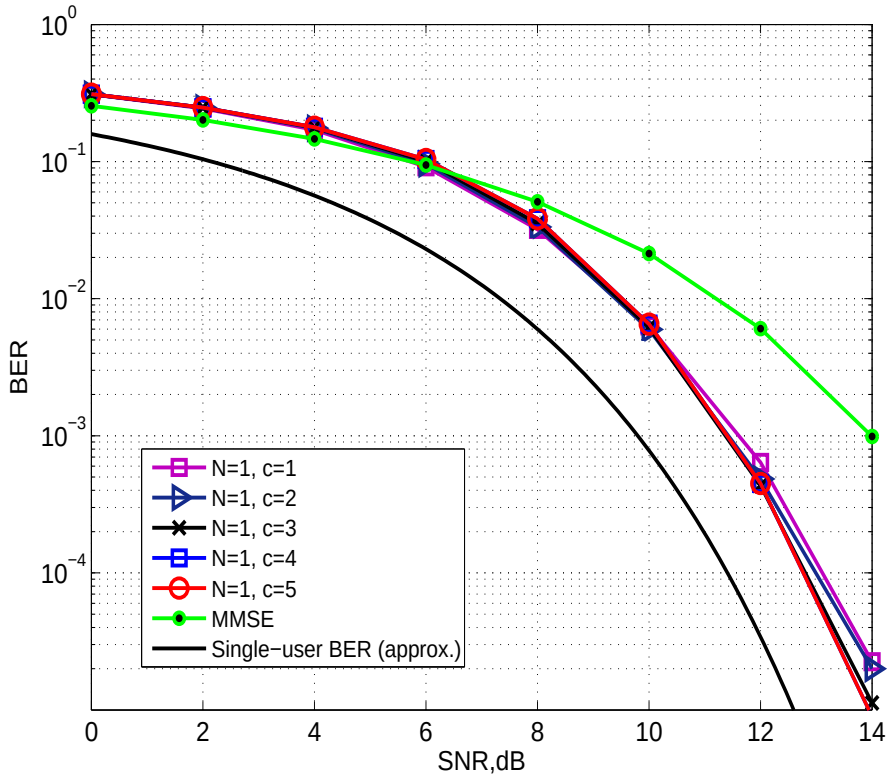


Figure 5.2: The effect of the constant c on BER performance of the DCD-BTN-M detector when $N = 1$ in the scenario of $K = 40$, SF $= 63$, $M_b = 15$.

MMSE detector is inferior to the proposed detector. The simulation results also show the effect of the constant c on the BER performance of the proposed detector. It shows that when $N = 1$, the increase of c can slightly improve the BER performance of this proposed detector at high SNRs. The performance cannot be further improved when $c > 4$. When $c = 4$, the difference in the performance compared to the single user bound is 1.5 dB at BER of 10^{-4}.

Fig. 5.3 shows the effect of the number of Tikhonov iterations N and the constant c on the BER performance of the proposed detector in the scenario of $K = 40$, SF $= 63$. When c varies from 1 to 4, increase in N can slightly improve the BER performance of the proposed detector at high SNRs. The symbol error rate (SER) performance in Fig. 5.4 and the group detection error (GDE) performance in Fig. 5.5 also show this.

Fig. 5.6 shows the complexity of the proposed detector in the scenario of $K = 40$ and SF $= 63$. It can be seen that when $N = 4$, this detector with $c = 4$ significantly reduces the complexity compared to that with $c = 1$. When $N = 1$, the detector with $c = 4$ has lower average complexity than that with $c = 1$, and also has a lower worst-case complexity at

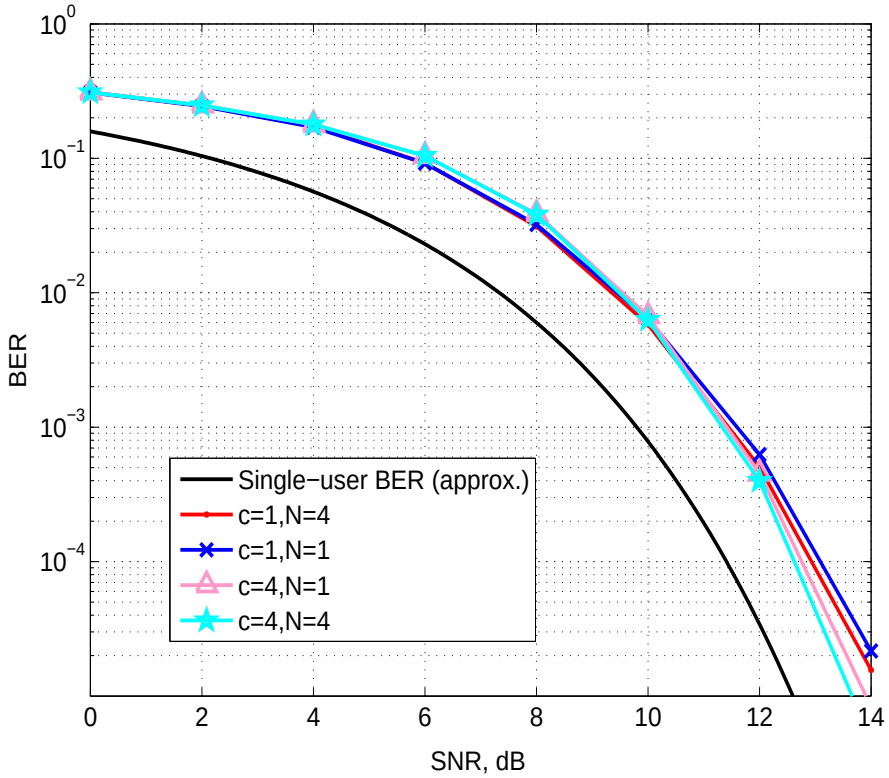


Figure 5.3: BER performance of the DCD-BTN-M detector in the scenario of $K = 40$, SF $= 63$, $M_b = 15$.

low SNRs.

Fig. 5.7 shows the BER performance of the proposed multiuser detector in comparison with MMSE detector in the highly loaded scenario of $K = 60$ and SF $= 63$. It shows that MMSE detector is inferior to the proposed detector. Increase in N cannot significantly improve the performance of this detector. Fig. 5.8 and Fig. 5.9 show the SER and GDE performance of the proposed detector, respectively. At SER $= 10^{-4}$, this proposed detector has approximately 2 dB difference to the single user bound. Increase in N cannot significantly improve the detection performance of proposed detector. Fig. 5.10 shows that the complexity of this proposed detector linearly increases with the increase of N. We also can see that this proposed detector with $c = 4$, and $N = 1$ has almost constant complexity when SNR > 8 dB.

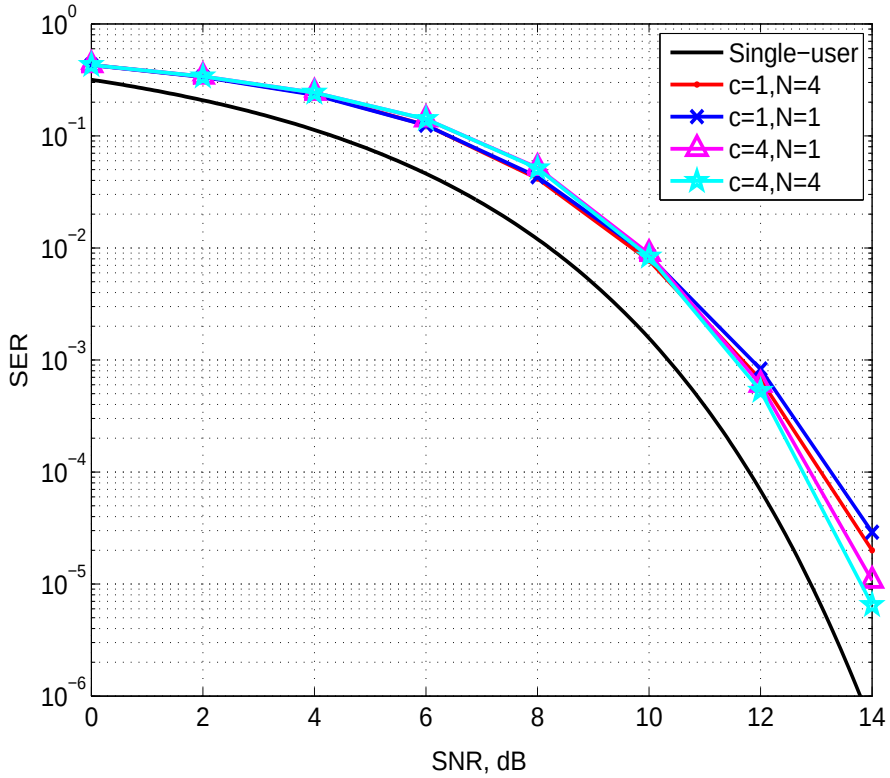


Figure 5.4: SER performance of the DCD-BTN-M detector in the scenario of $K = 40$, SF $= 63$, $M_b = 15$.

5.5 Conclusions

In this chapter, we further exploit the DCD-BTN detector and propose the DCD-BTN-M detector for M-PSK symbols. Numerical results show that the DCD-BTN-M detector is suitable for detection of the complex-valued symbols and provides better detection performance in comparison with the MMSE detector. The type of constrained used in the DCD-BTN-M detector is most suitable for M-PSK constellation. It can also be used for QAM symbols. However, for QAM symbols, this constrained is not tight enough. Therefore, for QAM symbols, the detection performance can be worse than that for M-PSK symbols.

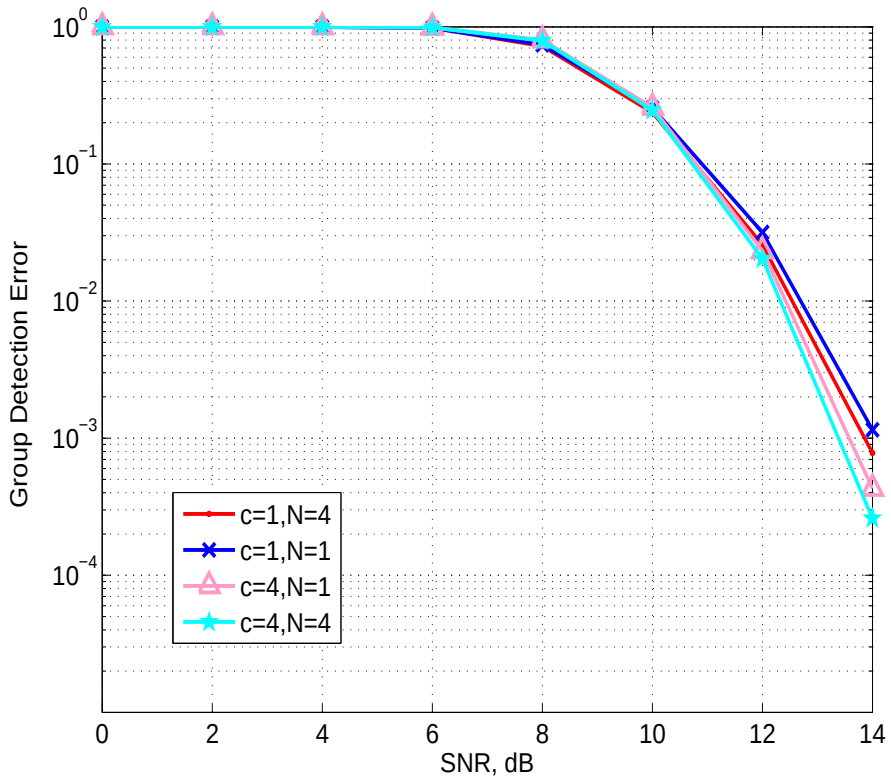


Figure 5.5: GDE performance of the DCD-BTN-M detector in the scenario of $K = 40$, SF $= 63$, $M_b = 15$.

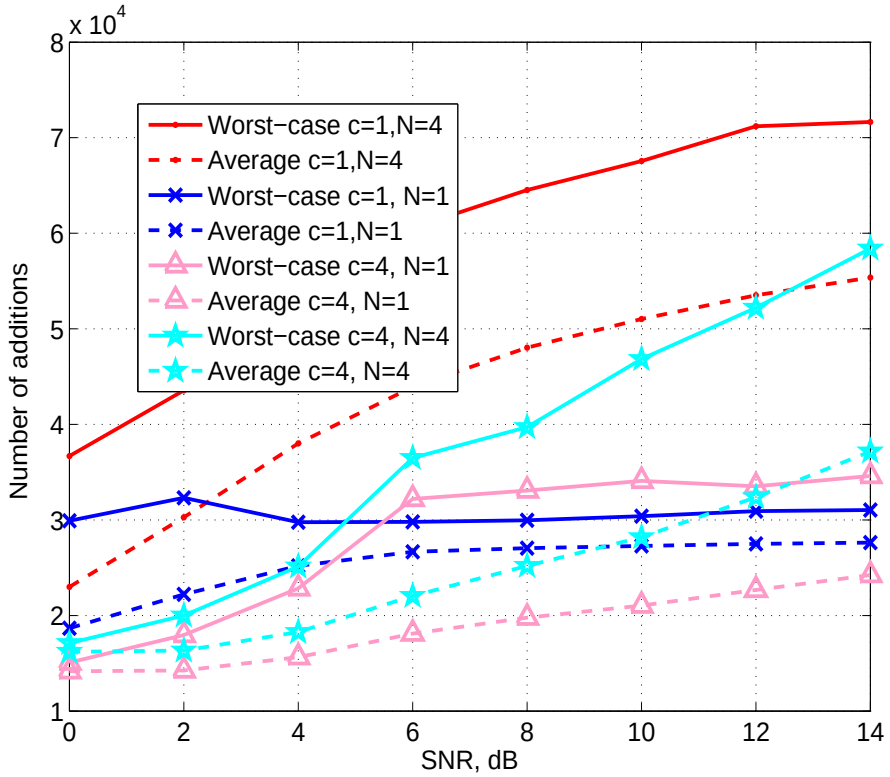


Figure 5.6: The complexity of the DCD-BTN-M detector in the scenario of $K = 40$, SF $= 63$, $M_b = 15$.

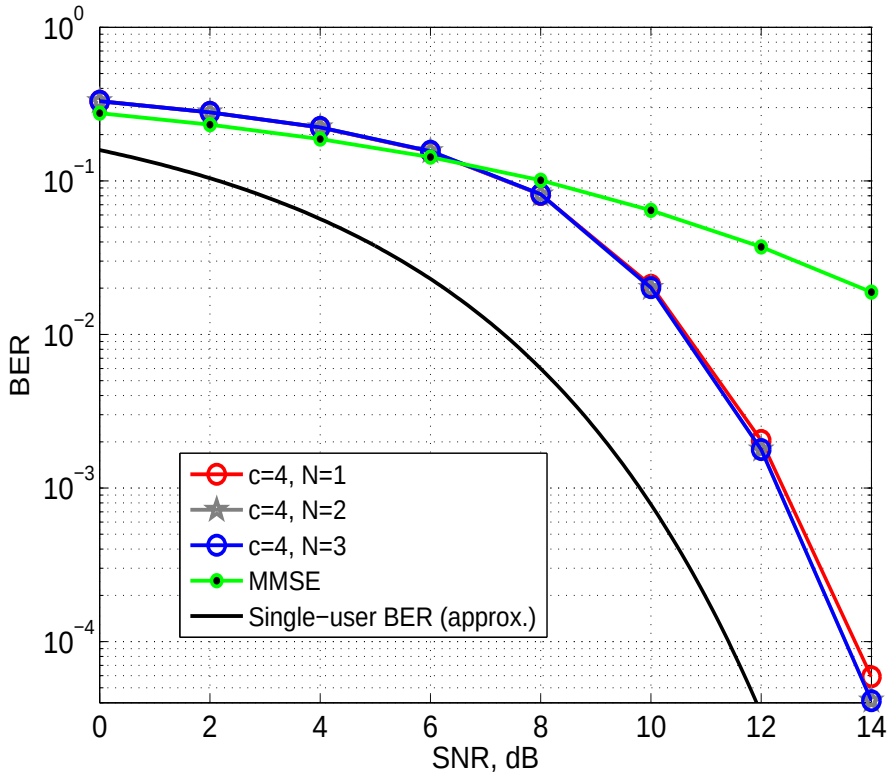


Figure 5.7: BER performance of the DCD-BTN-M detector in the scenario of $K = 60$, SF $= 63$, $M_b = 15$.

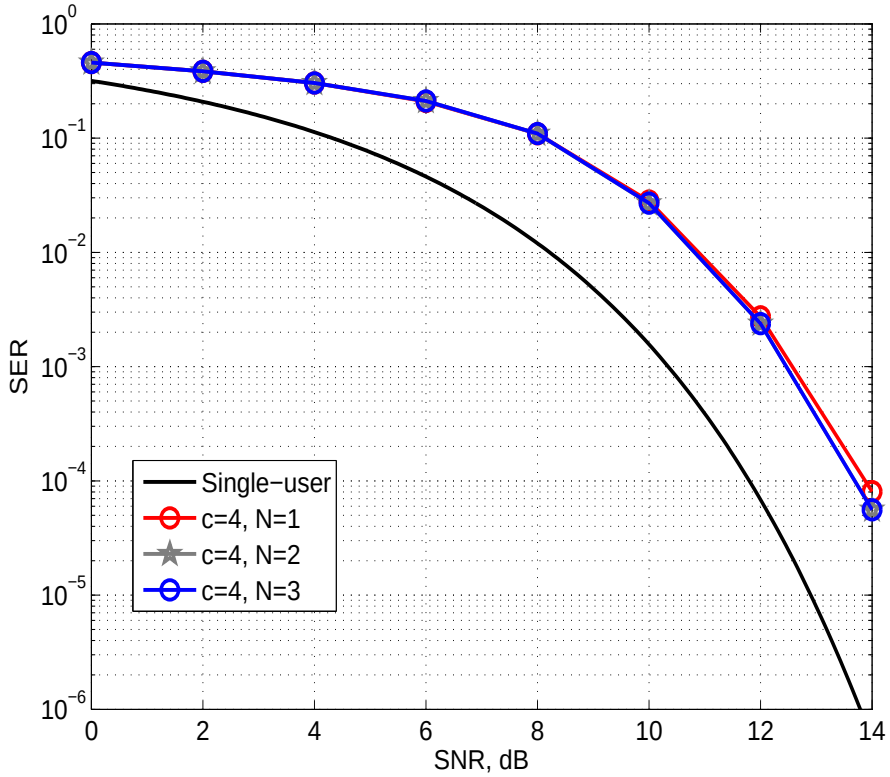


Figure 5.8: SER performance of the DCD-BTN-M detector in the scenario of $K = 60$, SF $= 63$, $M_b = 15$.

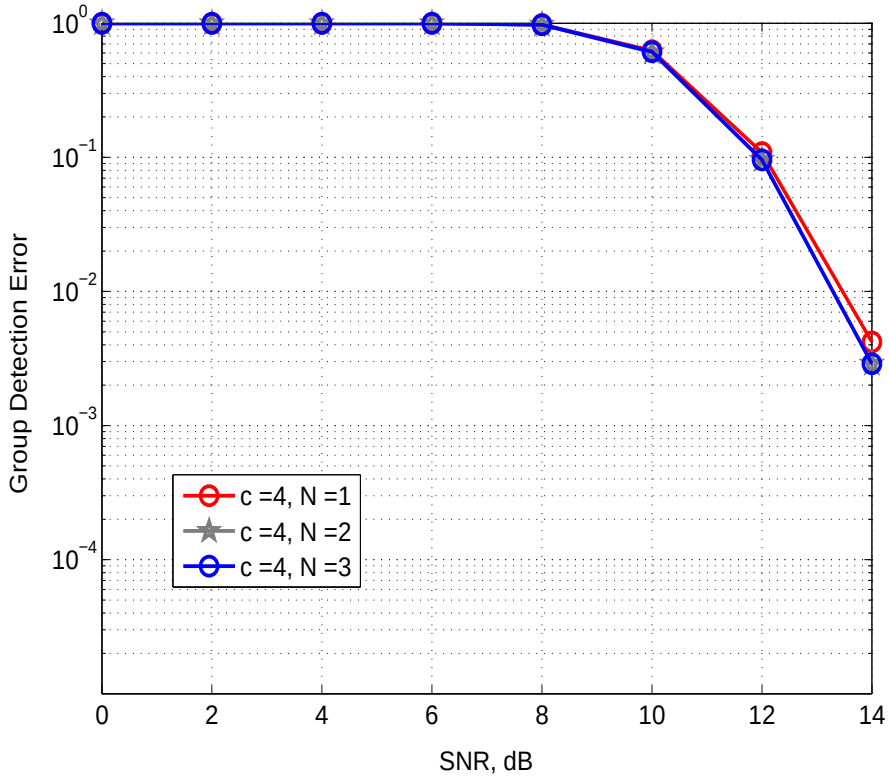


Figure 5.9: GDE performance of the DCD-BTN-M detector in the scenario of $K = 60$, SF $= 63$, $M_b = 15$.

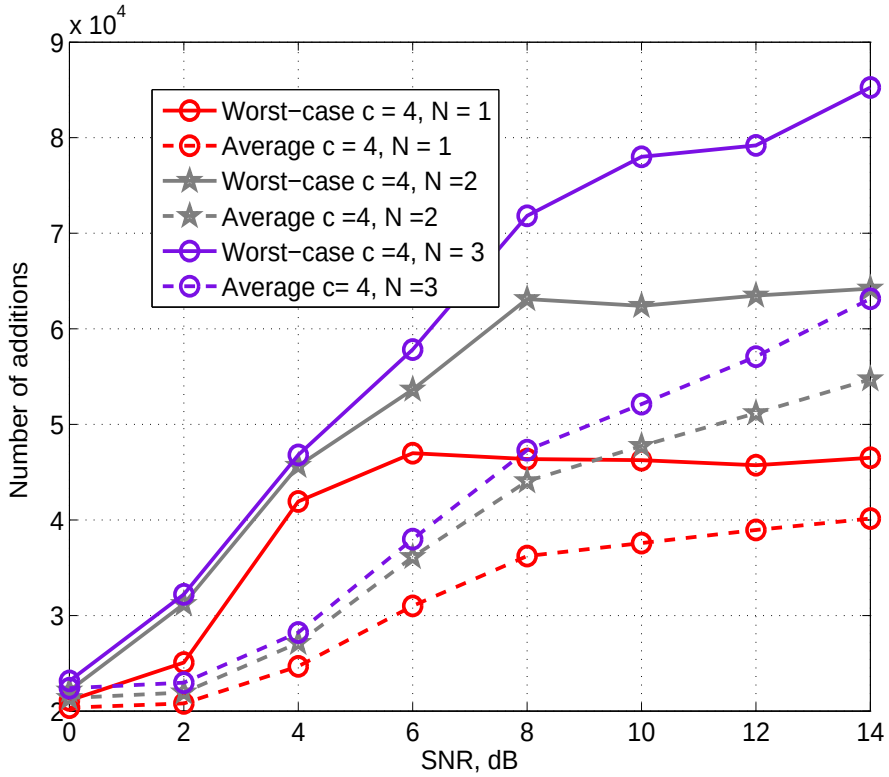


Figure 5.10: Complexty of the DCD-BTN-M detector in the scenario of $K = 60$, SF $= 63$, $M_b = 15$.

Chapter 6

Multiple Phase Detection of M-PSK Symbols

Contents

6.1 Introduction

In multiple-access CDMA systems, multiuser detection is capable of providing high detection performance [9]. However, the complexity of multiuser detectors that are capable of approaching the optimal performance is still a very important issue. For a small-size problem, sphere decoding achieves a nearly optimal performance [97], but becomes complicated when the size of the problem increases [98]. Semi-definite relaxation (SDR) has also been proposed and investigated for joint detection of a number of symbols with

M-PSK modulation [12]. Although, promising for multiuser detection, SDR is still complicated for practical implementation [99].

In the MIMO wireless systems, ML decoder offers performance advantages over sophisticated sub-optimal detectors [100, 101]. The sphere decoders using depth-first tree search can provide the optimal ML decoding performance, while retaining a lower complexity [102]. Branch and bound based ML decoders have also been proposed for implementation of ML detectors [39]. However, the computational complexity of depth-first sphere decoders and branch and bound decoders rely on a careful choice of initial sphere radius; if the initial radius is too small, new search has to be restarted and redundant computations occur [103].

In this chapter, we propose a new technique, multiple phase decoder (MPD), for solving the quadratic optimization problem. The MPD is based on a phase descent search (PDS) algorithm. The PDS algorithm is based on coordinate descent iterations, where coordinates are unknown symbol phases, while constraints the symbols to have a unit magnitude. The MPD is investigated for detection of M-PSK symbols in multiuser and MIMO systems. In the multiuser detection, the MPD can be applied to highly loaded scenarios and the numerical results show that the MPD can provide the near-optimal performance and allow low complexity. In the MIMO detection, the MPD exhibits more favorable performance/complexity characteristics and can be considered as a promising alternative to the sphere decoder for decoding in MIMO systems.

This chapter is organized as follows. In Section 6.2, the problem formulation is presented. In Section 6.3 proposes the phase descent search algorithm. Section 6.4 proposes the multiple phase detection for M-PSK symbols. Section 6.5 presents the simulation results of the multiple phase detection in multiuser, and the simulation results of the multiple phase detection in MIMO system are shown in section 6.6. Section 6.7 proposes the hardware implementation issues for the multiple phase detection. Section 6.8 concludes the chapter.

6.2 Problem Formulation

The multiuser and MIMO detection deal with the channel model can be considered to

$$\mathbf{z} = \mathbf{G}\mathbf{h} + \mathbf{n}\,, \tag{6.1}$$

where $\mathbf{h}$, $\mathbf{z}$ are $K \times 1$ vectors, $\mathbf{G}$ is a $K \times K$ symmetric positive definite matrix, $\mathbf{n}$ is a $K \times 1$ zero mean Gaussian random vector with variance σ^2. The vector $\mathbf{h}$ can be found by

$$\begin{aligned} \hat{\mathbf{h}} &= \arg \min_{\mathbf{h} \in \mathcal{A}^K} \|\mathbf{z} - \mathbf{G}\mathbf{h}\|^2 \\ &= \arg \min_{\mathbf{h} \in \mathcal{A}^K} \left(\mathbf{h}^H \mathbf{R}\mathbf{h} - 2\Re\{\boldsymbol{\theta}^H \mathbf{h}\}\right). \end{aligned} \tag{6.2}$$

We denote

$$J(\mathbf{h}) = \mathbf{h}^H \mathbf{R}\mathbf{h} - 2\Re\{\boldsymbol{\theta}^H \mathbf{h}\}\,, \tag{6.3}$$

the quadratic cost function, $\mathbf{R} = \mathbf{G}^H\mathbf{G}$, and $\boldsymbol{\theta} = \mathbf{G}^H\mathbf{z}$. The ML detector provides the optimal detection performance. However the complexity of the ML detector is exponential in the constellation set $\mathcal{A}$ and the number of users K (in multiuser detection) or transmit antennas (in MIMO communications).

6.3 Phase Descent Search Algorithm

The phase descent search (PDS) algorithm is based on coordinate descent iterations with respect to the unknown symbol phases and a constraint that forces the symbols to have a unit magnitude. Specifically, elements of the solution are given by

$$h_k = e^{j\phi_k}\,, \quad k = 0, \ldots, K-1, \quad \phi_k \in [-\pi, \pi]\,. \tag{6.4}$$

The coordinate descent iterations are applied to the phases ϕ_k. The derivation below is based on a general coordinate descent method described in [104] (see also [8]). Here, we apply this method to the cost function (6.3) with elements h_k from (6.4).

Let $\mathbf{h}(i-1)$ be a solution obtained at the $(i-1)$th iteration; elements of $\mathbf{h}(i-1)$ are denoted as $h_k(i-1)$, $k = 0, \ldots, K-1$. At the ith iteration, the solution $\mathbf{h}(i)$ may differ from $\mathbf{h}(i-1)$ in the pth element only. This element may be updated as

$$h_p(i) = h_p(i-1)e^{\pm jd}\,, \tag{6.5}$$

where $d \in [0, 2\pi]$ is a step-size parameter (we call such an iteration *successful*), or it may stay unchanged (we call such an iteration *unsuccessful*). Depending on a sequence of unsuccessful iterations, the step-size parameter can be reduced; this will be explained below. We can express the update (6.5) as

$$\mathbf{h}(i) = \mathbf{h}(i-1) + \mathbf{\Delta}\,,$$

where $\mathbf{\Delta} = uh_p(i-1)\mathbf{e}_p$, $\mathbf{e}_p$ is a vector whose elements are zeros except the pth element which is equal to 1, and

$$u \in \{u_1, u_2\},\ \ u_1 = e^{-jd} - 1,\ \ u_2 = e^{jd} - 1.$$

The update is applied (*i.e.* the iteration is successful) if the cost function (6.3) is reduced, *i.e.*

$$\Delta J(i) = J[\mathbf{h}(i)] - J[\mathbf{h}(i-1)] < 0\,.$$

The decrement $\Delta J(i)$ of the cost function can be written as

$$\begin{aligned}\Delta J(i) \;\; = \;\; & [\mathbf{h}(i-1) + \mathbf{\Delta}]^H \mathbf{R}[\mathbf{h}(i-1) + \mathbf{\Delta}] - \mathbf{h}^H(i-1)\mathbf{R}\mathbf{h}(i-1) \\ & -2\Re\{\boldsymbol{\theta}^H[\mathbf{h}(i-1) + \mathbf{\Delta}]\} + 2\Re\{\boldsymbol{\theta}^H\mathbf{h}(i-1)\}\,. \end{aligned} \tag{6.6}$$

The first two terms in (6.6) can be represented as

$$\mathbf{\Delta}^H\mathbf{R}\mathbf{\Delta} + 2\Re\{\mathbf{\Delta}^H\mathbf{R}\mathbf{h}(i-1)\}.$$

It is easy to see that $\mathbf{\Delta}^H\mathbf{R}\mathbf{\Delta} = R_{p,p}|u|^2$ and

$$\Re\{\mathbf{\Delta}^H\mathbf{R}\mathbf{h}(i-1)\} = \Re\left\{u^* h_p^*(i-1)\sum_{k=0}^{K-1} R_{p,k}h_k(i-1)\right\},$$

where $R_{p,k}$ are elements of the matrix $\mathbf{R}$. As the last two terms in (6.6) can be represented as $-2\Re\{\mathbf{\Delta}^H\boldsymbol{\theta}\} = -2\Re\{u^*h_p^*(i-1)\theta_p\}$, we finally obtain

$$\Delta J(i) = R_{p,p}|u|^2 - 2\Re\left\{u^* h_p^*(i-1)r_p(i-1)\right\}, \tag{6.7}$$

where $r_p(i-1) = \theta_p - \sum_{k=0}^{K-1} R_{p,k}h_k(i-1)$ is the pth element of the residual vector $\mathbf{r}(i-1) = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}(i-1)$ and θ_p is the pth element of $\boldsymbol{\theta}$. It shows that the residual vector can be recursively updated as

$$\mathbf{r}(i) = \mathbf{r}(i-1) - uh_p(i-1)\mathbf{R}(:,p) \tag{6.8}$$

with an initialization $\mathbf{r}(0) = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}_0$, where $\mathbf{h}_0$ is an initialization for the solution vector, and $\mathbf{R}(:,p)$ is the pth column of $\mathbf{R}$.

From (6.7), we conclude that the ith iteration is successful if

$$R_{p,p}|u|^2 < 2\Re\left\{u^* h_p^*(i-1) r_p(i-1)\right\}. \tag{6.9}$$

Now we summarize the iterative Phase Descent Search (PDS) algorithm as shown in Table 6.1, where p is chosen in a circle order, *i.e.* $p = i \bmod K,\ \ p = 0, \ldots, K-1$, and we take into account that $|u|^2 = 2[1 - \cos(d)]$.

If one of the inequalities at steps 8 or 14 is satisfied, the iteration is successful. The element h_p and the residual vector $\mathbf{r}$ are updated (steps 10-11 or 16-17, respectively); otherwise, they are not changed. The index n denotes the number of successful updates; it is used for introducing the stopping criterion at step 19, where N_u is a predefined parameter that limits the maximum number of successful updates. In a pass for achieving mth bit, including K iterations with $p = 0, \ldots, K-1$, if there is no successful update, the step-size is reduced at step 2: $d = d_0\lambda^m$, $0 < \lambda < 1$, where d_0 is an initial value of d. It is natural to choose $d_0 = 2\pi$. The choice of λ may depend on the modulation scheme used; this will be addressed below. The parameter M_b indicates the number of reductions of the step-size d and, thus, the final phase resolution $d_0\lambda^{M_b}$; *e.g.* in the case of $\lambda = 1/2$ and $M_b = 8$, the final phase resolution is $2\pi/2^{M_b} = \pi/128$.

Table 6.1 also shows the complexity of different steps of the PDS algorithm in terms of real multiplications and additions, as well as the maximum complexity. The computational load of the algorithm mainly depends on the system size, and the condition number of the system matrix, as well as N_u and M_b.

A successful iteration requires 4 real multiplications and 8 additions for comparison, $4K$ multiplications and $4K$ additions for updating $\mathbf{r}$. For an unsuccessful iteration, only 4 real multiplications and 8 additions are used for the comparison. The worst case complexity corresponds to an unlikely situation, which occurs when only the last bit has N_u successful iterations. This means that the calculation of the first $(M_b - 1)$ bits do not contain any successful iteration, and so, require $4K(M_b - 1)$ real multiplications, and $8K(M_b - 1)$ additions. The worst-case complexity for calculation of the last bit (corresponds to $m = M_b$) occurs when only one successful iteration happens among the K iterations ($p = 0, 1, \cdots, K-1$). This requires $4K$ real multiplications and $8K$ real additions for the comparison, $4K$ real multiplications and $4K$ real additions to update $\mathbf{r}$. In addition, for all M_b bits, M_bK real multiplications are required for calculating $\boldsymbol{\psi}$ at step 3. In total, N_u successful iterations require $N_u(4K + 4K) + M_bK$ real multiplication, $N_u(8K + 4K)$ additions.

Table 6.1: Phase Descent Search algorithm

Step	Equation	$\times$	$+$
Init.	$\mathbf{h} = \mathbf{h}_0$, $\boldsymbol{\phi} = \boldsymbol{\phi}_0$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}_0$, $d = d_0$, $n = 0$		
1	for $m = 1 : M_b$		
2	$d = \lambda d$	$-$	$-$
3	$\boldsymbol{\psi} = \text{diag}\{\mathbf{R}\}[1 - \cos(d)]$	K	$-$
4	Flag$= 0$	$-$	$-$
5	for $p = 0 : (K - 1)$	$-$	$-$
6	$\phi_{p,1} = \phi_p + d$, $h_{p,1} = e^{j\phi_{p,1}}$	$-$	1
7	$\Delta_1 = h_{p,1} - h_p$, $T_1 = \Re\{\Delta_1^* r_p\}$	2	2
8	if $\psi_p < T_1$	$-$	1
9	$n = n + 1$, Flag$= 1$	$-$	$-$
10	$\mathbf{r} = \mathbf{r} - \Delta_1 \mathbf{R}(:, p)$	$4K$	$4K$
11	$\phi_p = \phi_{p,1}$, $h_p = h_{p,1}$	$-$	$-$
12	$\phi_{p,2} = \phi_p - d$, $h_{p,2} = e^{j\phi_{p,2}}$	$-$	1
13	$\Delta_2 = h_{p,2} - h_p$, $T_2 = \Re\{\Delta_2^* r_p\}$	2	2
14	if $\psi_p < T_2$	$-$	1
15	$n = n + 1$, Flag$= 1$	$-$	$-$
16	$\mathbf{r} = \mathbf{r} - \Delta_2 \mathbf{R}(:, p)$	$4K$	$4K$
17	$\phi_p = \phi_{p,2}$, $h_p = h_{p,2}$	$-$	$-$
18	end the loop over p	$-$	$-$
19	if $n > N_u$ the algorithm stops	$-$	$-$
20	if Flag$= 1$ go to step 4	$-$	$-$
21	end the loop over m	$-$	$-$
	Total complexity: $\leq 8KN_u + 5KM_b$ real multiplications and $\leq 12KN_u + 8KM_b$ real additions		

Table 6.2: Multiple Phase Algorithm

Step	equation
	for $q = 1 : Q$
1	Initialize the PDS algorithm with
	$M_b > 1$, $\lambda = 1/2$, $\boldsymbol{\phi} = \boldsymbol{\phi}_q$, $\mathbf{h} = \mathbf{h}_q$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}_q$
	Apply the PDS algorithm to obtain a solution $\mathbf{h}$
2	Map $\mathbf{h}$ into the M-PSK constellation to obtain $\tilde{\boldsymbol{\phi}}$ and $\tilde{\mathbf{h}}$
3	Initialize the PDS algorithm with
	$M_b = 1$, $\lambda = 1/M$, $\boldsymbol{\phi} = \tilde{\boldsymbol{\phi}}$, $\mathbf{h} = \tilde{\mathbf{h}}$, $\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\tilde{\mathbf{h}}$
	Apply the PDS algorithm to obtain a solution $\mathbf{h}$
4	Calculate the cost function for the solution $\mathbf{h}$
	end
	Choose the solution with the minimum cost

Therefore, the complexity of the PDS algorithm is upper bounded by $8KN_u + 5KM_b - 4K \approx 8KN_u + 5KM_b$ real multiplications, $12KN_u + (8K+1)M_b - 8K \approx 12KN_u + 8KM_b$ additions. However in a typical situation, there should be several successful iterations in each pass $(m = 1, \cdots, M_b)$, and the average complexity will be close to $8KN_u$ real multiplications and $12KN_u$ additions.

6.4 Multiple Phase Decoder

The PDS algorithm with a consequent mapping of the solution to the constellation $\mathcal{A}$ is shown to provide a good detection performance for highly loaded BPSK systems. However, for M-PSK systems with $M > 2$, the performance can be improved by multiple use of the PDS algorithm in the multiple phase algorithm as shown in Table 6.2.

The PDS algorithm is used Q times with different initializations of the solution vector. To obtain a solution for the qth initialization $\mathbf{h}_q$, we apply the PDS algorithm twice. Among the Q solutions, the one that has the smallest cost function $J(\mathbf{h})$ is selected. In the first PDS, we use a high M_b and $\lambda = 1/2$. With the consequent mapping (Step 2) to the constellation $\mathcal{A}$, the first PDS algorithm provides an initial solution for the second PDS algorithm. The second PDS algorithm performs a descent local search moving from one ML feasible solution to another in the neighborhood of the initial solution [99]. We use $M_b = 1$ and $\lambda = 1/M$, $\tilde{\mathbf{h}} = \exp(j\tilde{\boldsymbol{\phi}})$, and the PDS algorithm performs M-PSK *symbol-*

flipping. For initialization of the solution vector, we use the following vectors

$$\mathbf{h}_q = \exp\left(j\frac{\pi q}{Q}\right)\mathbf{1} = \exp\left(j\boldsymbol{\phi}_q\right), \tag{6.10}$$

where $\mathbf{1}$ is a K-length vectors of ones and $\boldsymbol{\phi}_q = (\pi q/Q)\mathbf{1}$.

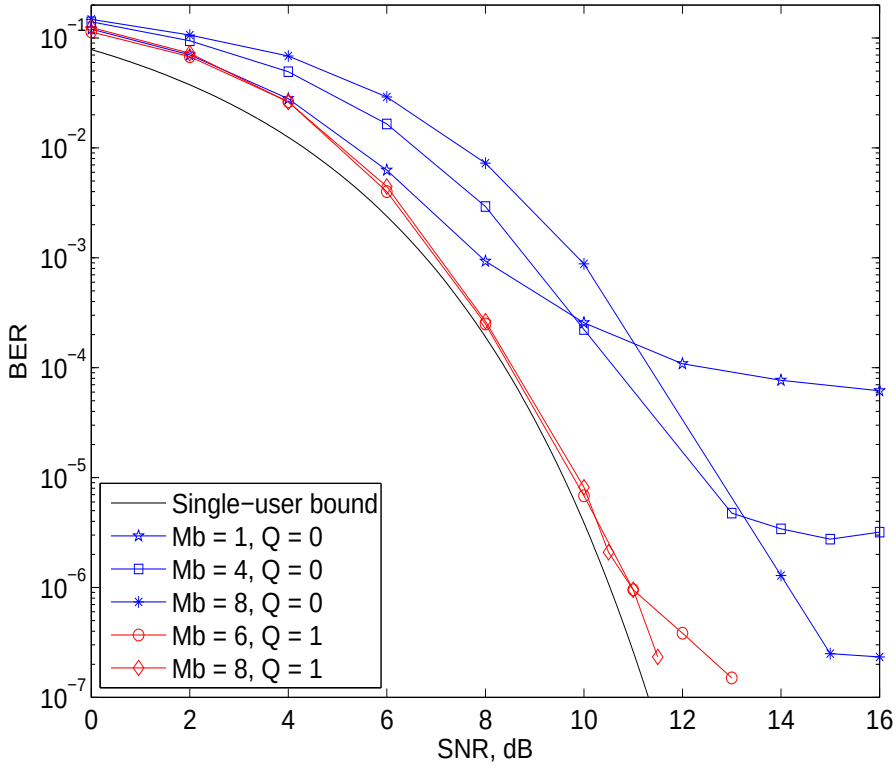


Figure 6.1: BER performance of MPD-based multiuser detector; BPSK modulation, $K = 60$, SF $= 63$.

6.5 Simulation Results for MPD in Multiuser Detection

Fig.6.1 shows the BER performance of the proposed detector for BPSK modulated signals against the single-user bound, obtained in 10^6 simulation trials. The user signature waveforms have equal energies. They are binary and chosen randomly in each simulation trial. The number of users is $K = 60$ and the spreading factor is SF $= 63$. The case $Q = 0$ corresponds to the PDS algorithm with a consequent mapping of the solution to the set $\mathcal{A}$ (steps 1 and 2 in Table 6.2). The case $M_b = 1$ with $\lambda = 1/2$ corresponds to bit-flipping, *i.e.* changing the pth coordinate of the solution vector between $+1$ and -1. The bit-flipping provides a good performance at low SNRs, but at high SNRs, there is a

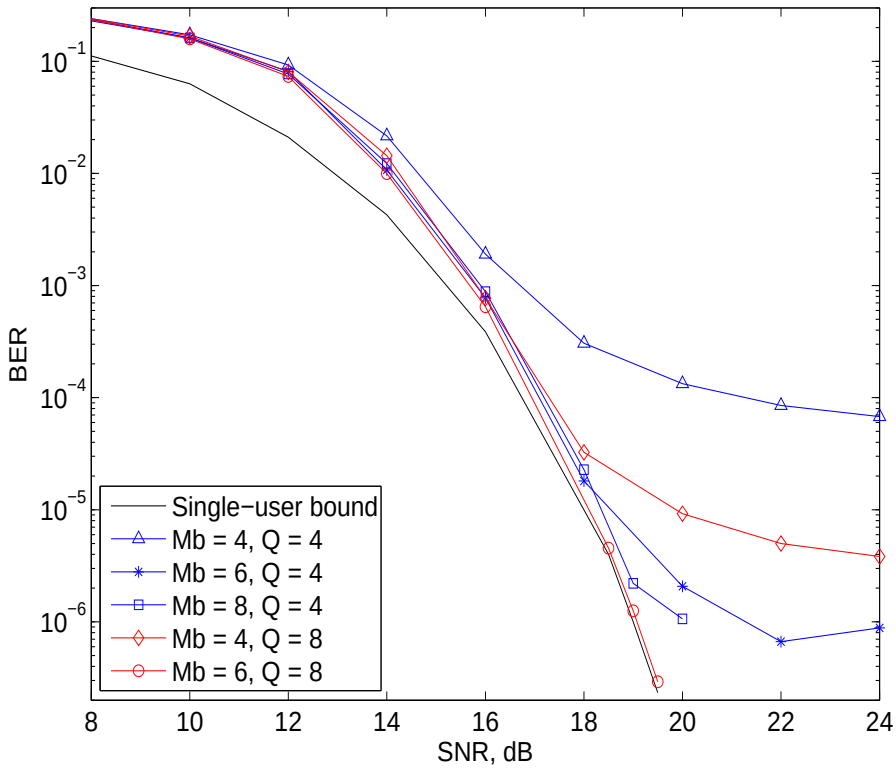


Figure 6.2: BER performance of MPD-based multiuser detector; 8-PSK modulation, $K = 60$, SF $= 63$.

BER floor. As M_b increases, the BER floor level is reduced; however, the performance at low SNRs becomes worse. In the case of $Q = 1$, the PDS algorithm is used again at step 3, now with $M_b = 1$, *i.e.* the second PDS algorithm provides the bit-flipping, which results in significant improvement in the BER performance. For a highly loaded scenario, the performance becomes very close to the single-user bound. We compare the performance with the single-user bound (instead of the ML performance) because simulation of the ML detector with $K = 60$ users would be impractical.

Fig.6.2 shows the BER performance for a scenario with 8-PSK modulation. For this scenario, the use of $Q = 0$ or $Q = 1$ does not allow the detection performance to approach the single-user bound. For $Q = 4$, as M_b increases, the BER floor level is reduced, whereas the BER curves slightly depart from the single-user bound. However, for $Q = 8$ and $M_b = 6$ at high SNRs, the BER performance of the MPD is very close to the single-user bound.

Fig.6.3 compares the symbol-error-rate (SER) performance of the MPD against the SDR detector for 8-PSK modulation. The results for the SDR detector are taken from [105],

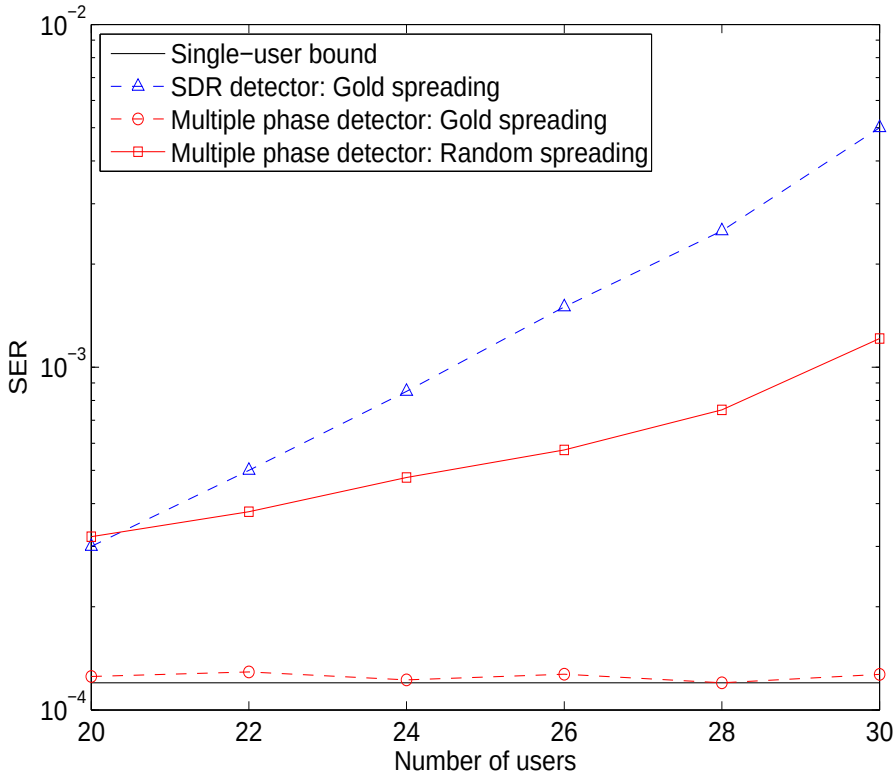


Figure 6.3: SER performance of the multiple phase detector against the SDR detector; 8-PSK modulation, $Q = 4$, $M_b = 8$, SF $= 31$, SNR $= 17$ dB.

where the simulation has been done with Gold signature waveforms of length SF $= 31$. We also use the Gold signature waveforms in the same scenarios, where the number of users varies from $K = 20$ to $K = 30$. Fig.6.3 shows that the MPD significantly outperforms the SDR detector and the performance of the proposed detector is very close to the single-user bound. The Gold signature waveforms have good correlation properties, however, the good correlation properties are easily distorted due to the complex environment in practical transmission. The random signature waveform can proximately imitate the sequences in real systems. Therefore, we have also repeated the simulation for binary signature waveforms randomly chosen in each simulation trial; it can be seen that the performance is similar or better than that of the SDR detector with Gold signature waveforms.

Fig.6.4 shows the BER performance of the MPD in overloaded multiuser scenarios with BPSK modulation where the number of users exceeds the spreading factor. The signature waveforms are random binary with equal energies. It is seen that the proposed detector allows reliable detection even in these difficult situations. However, this requires an increase in the number of initializations Q.

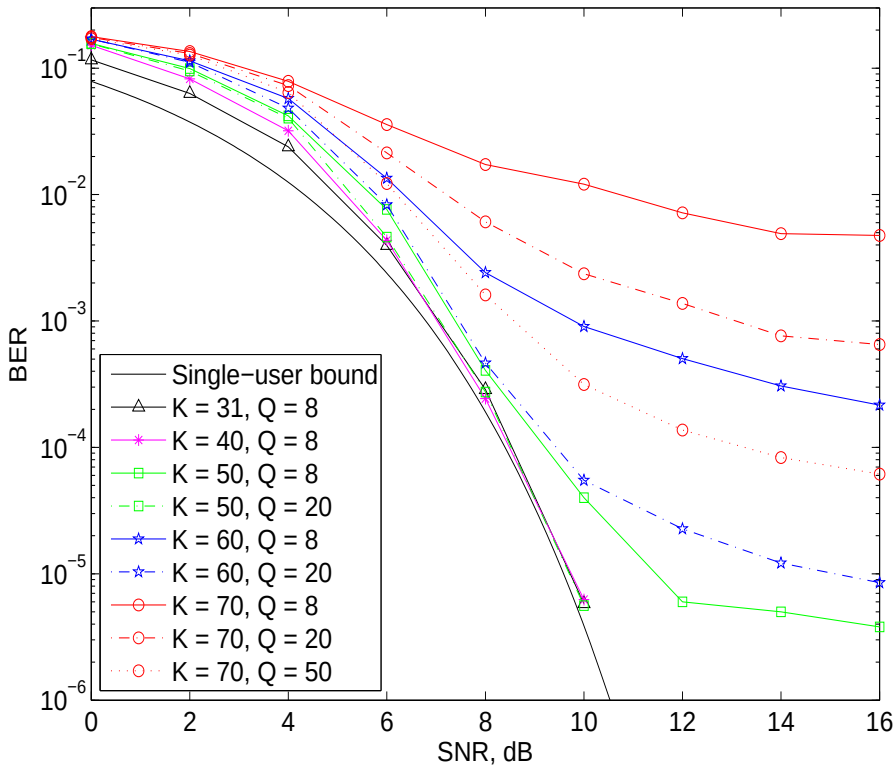


Figure 6.4: BER performance of the multiple phase detector in overloaded scenarios; BPSK modulation, SF $= 31$.

Finally, we provide simulation results that demonstrate near-far resistance of the proposed detector. We consider scenarios where all but the first user has the same SNR [99]. In Fig.6.5, the BER performance of the first user is shown against the ratio of the strength of the interfering user signals to the first user signal for BPSK modulation. In the case of $K = 10$ and SF $= 31$ (this case and the case $K = 24$ and SF $= 31$ are similar to that considered in [99]), the near-far resistance of the proposed detector is close to that of the ML detector. In this case, increase in Q does not improve the BER performance. In the case of $K = 24$, increase in Q results in a slight improvement. Note that the direct simulation of the ML detector for $K = 24$ is impractical, therefore we cannot compare our results with the ML performance. In [99], the BER performance of an ML detector implemented using a branch and bound algorithm based on SDR is given for the same scenario. However, our results show slightly better near-far resistance compared to the ML performance presented in [99]. It is seen that with further increase of the load (the case of $K = 60$ and SF $= 63$), the proposed detector demonstrates small variations in the near-far resistance. Fig.6.6 shows that the proposed detector has also good near-far resistance for 8-PSK modulation.

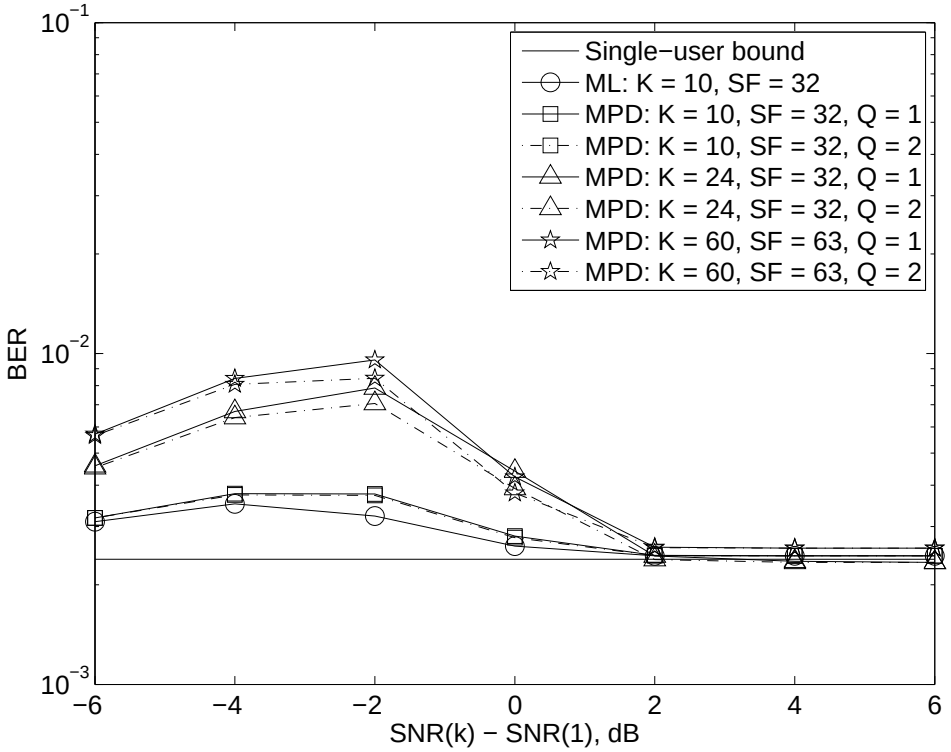


Figure 6.5: BER performance of the multiple phase detector in near-far scenarios; BPSK modulation, SNR(1) = 6 dB.

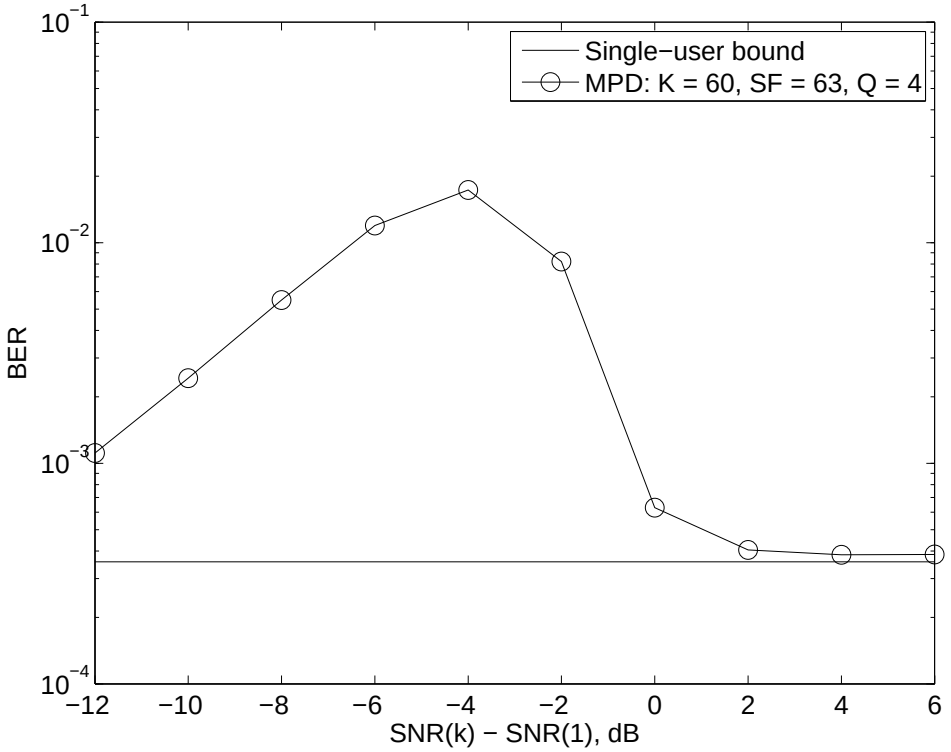


Figure 6.6: BER performance of the multiple phase detector in near-far scenarios; 8-PSK modulation, SNR(1) = 16 dB.

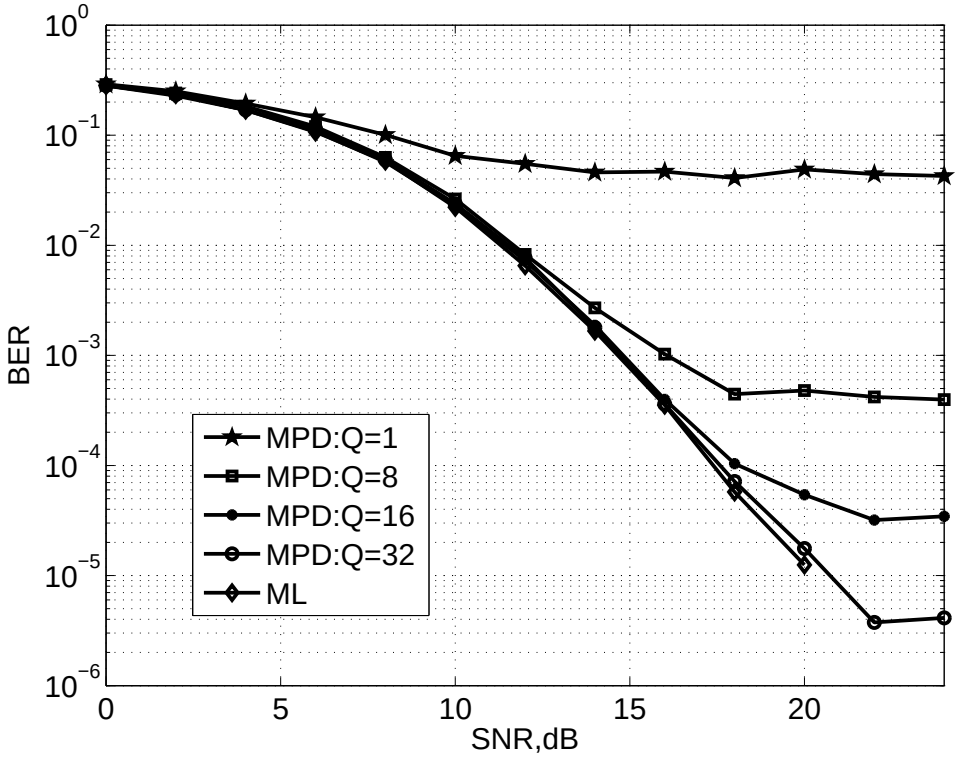


Figure 6.7: BER performance of the multiple phase decoder using different Q against the ML decoder in a 4×4 MIMO system with QPSK modulation; $M_b = 6$.

In the simulation, we computed the average number of multiplications in the proposed detector with the best performance in Fig.6.2, *i.e.* $Q = 8$ and $M_b = 6$. On average, the detector requires approximately $5 \cdot 10^5$ multiplications to detect all $K = 60$ user symbols. Note that the complexity of the decorrelator (one of the simplest detectors) is approximately $K^3 \approx 2 \cdot 10^5$ multiplications, *i.e.* the complexity of the proposed multiple phase detector with $Q = 8$ branches in such a highly loaded scenario is close to that of the decorrelator. In [105], complexity results for the SDR detector are presented; specifically, for $K = 10$, the SDR detector requires approximately $2 \cdot 10^6$ multiplications. Our simulation results have shown that, in the case of $K = 10$, $Q = 8$, and $M_b = 6$, the proposed detector requires on average $1.2 \cdot 10^4$ multiplications, *i.e.* two orders of magnitude lower than that of the SDR detector.

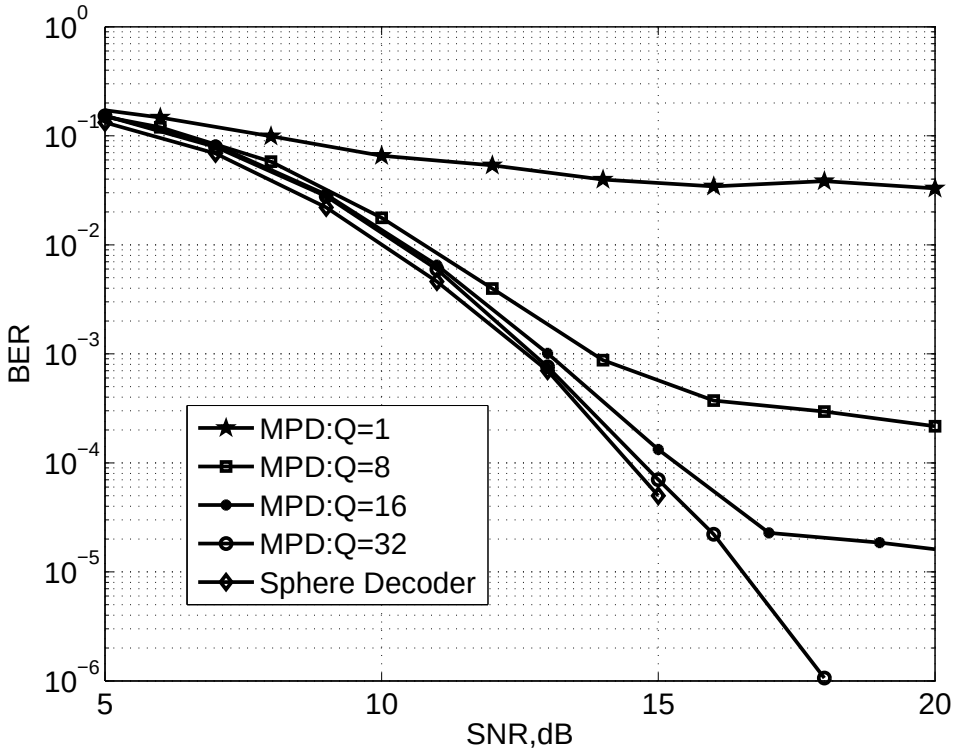


Figure 6.8: BER performance of the multiple phase decoder using different Q against the sphere decoder in a 8×8 MIMO system with QPSK modulation; $M_b = 6$.

6.6 Simulation Results for MPD in MIMO Detection

In this section, MPD is applied to solve the detection problem in MIMO systems, with near optimal ML performance and low complexity.

Fig.6.7 and Fig.6.8 show the detection performance of the multiple phase decoder for 4×4 and 8×8 MIMO systems with QPSK modulation. The performance of the MPD with various Q is compared to the performance obtained from ML decoder and the sphere decoder. The simulation trials is 10^6. Fig.6.7 and Fig.6.8 show that the MPD with $Q = 1$ does not provide a good performance. When $Q = 8$, the MPD makes a significant performance improvement in comparison with the case of $Q = 1$, but there is a BER floor at high SNRs. As Q increases, the BER floor level is reduced. When $Q = 32$, the MPD in the 4×4 MIMO system shown in Fig.6.7 performs near the ML detector boundary. Fig.6.8 shows that the BER performance of the MPD in the 8×8 MIMO system is close to that of the sphere decoder; the BER performance of the sphere decoder is taken from [106].

Fig.6.9 shows the average complexity and the worst case complexity of the multiple phase

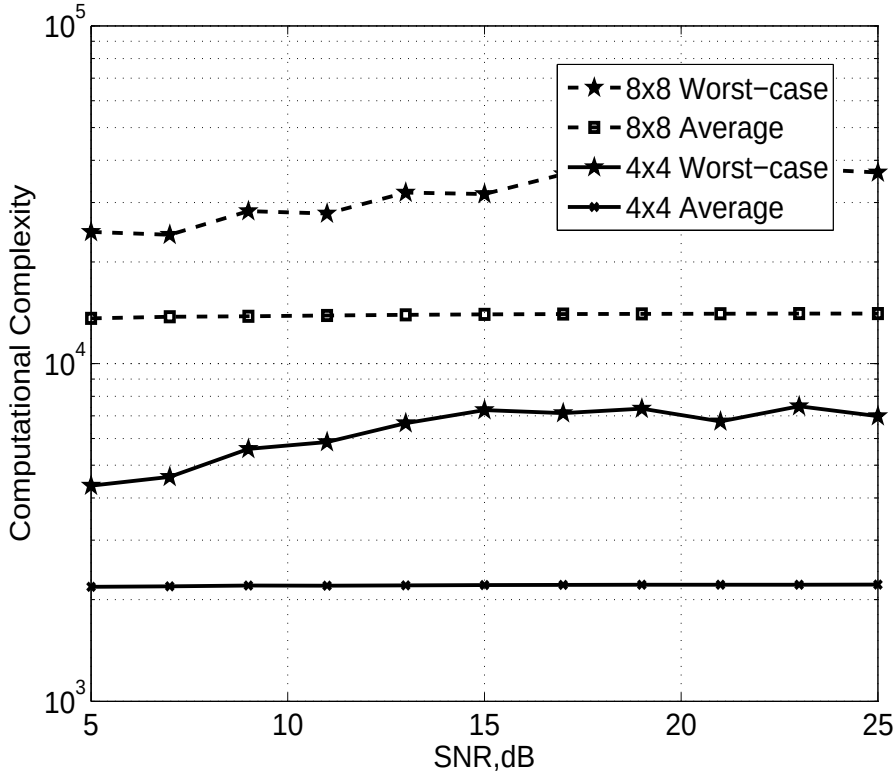


Figure 6.9: Computational complexity versus the SNR of the multiple phase decoder for 4×4 and 8×8 MIMO systems with QPSK modulation; $Q = 16$.

decoder in the 4×4 and 8×8 MIMO systems with $Q = 16$. It can be seen that the average complexity keeps stable for the whole region of SNRs. The average complexity of the MPD in the 8×8 MIMO system is about six times higher than that of MPD in the 4×4 MIMO system. The worst-case complexity varies over the SNR range, however the variations are insignificant. It can be seen that the worst-case complexity of the MPD in the 8×8 MIMO system is about four times higher than that of the MPD in the 4×4 MIMO system. The worst-case complexity of the sphere decoder is known to be an exponential function of system size [107].

Fig.6.10 shows the decoding complexity of the proposed decoder and the sphere decoder in MIMO communication. The complexity of the sphere decoder is taken from [108]. Fig.6.10(a) shows that the complexity of the sphere decoder is exponentially proportional to M_T at low SNRs [109]. The proposed decoder with $Q = 4$ offers a much lower complexity than the sphere decoder by providing a similar decoding performance of $BER \approx 2 \times 10^{-2}$. Fig.6.10(b) demonstrates that in the higher SNR region corresponding to $3 \times 10^{-5} < BER < 9 \times 10^{-5}$, the complexity of the sphere decoder is similar to that of the proposed decoder with $Q = 32$ when the number of transmit an-

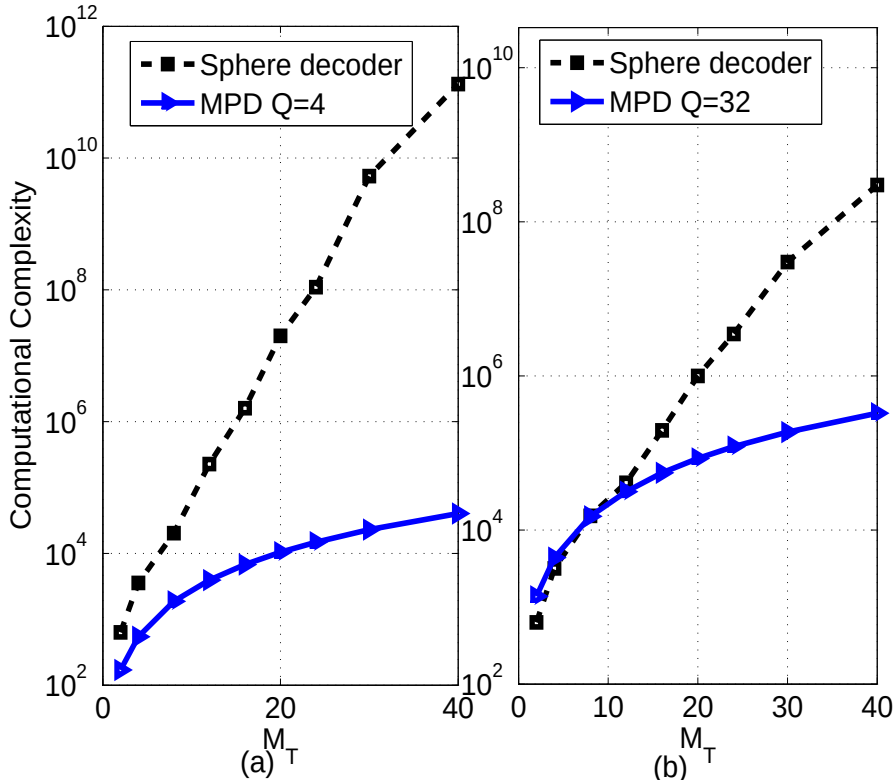


Figure 6.10: Decoding complexity verses the number of transmit antennas with QPSK modulation for sphere decoder and multiple phase decoder; a) BER $= 2 \times 10^{-2}$ and for MPD with $Q = 4$, b) $3 \times 10^{-5} <$ BER $< 9 \times 10^{-5}$ and for MPD with $Q = 32$.

tennas is less than 20. However, the sphere decoder still shows higher complexity than the proposed decoder when the number of transmit antennas increases. Overall, the average computational complexity of the proposed decoder is lower than that of the sphere decoder.

6.7 MPD Implementation Issues

In this section, we discuss the implementation of the proposed multiple phase detector in hardware. This detector can be implemented as Q identical parallel branches as in Fig. 6.11, each containing two PDS blocks (corresponding to steps 1 and 3 in Table 6.2), a mapping block (step 2 in Table 6.2) and a block for computation of the cost function (step 4 in Table 6.2). Table 6.3 shows the PDS algorithm in a form suitable for FPGA implementation.

Table 6.3: PDS in a form suitable for an FPGA implementation

Step	Operation	$\times$	$+$
0	$\boldsymbol{\phi} = \boldsymbol{\phi}_q$, $\mathbf{h}_q = \exp(j\boldsymbol{\phi}_q)$, $m = 0$, $n = 0$, $p = 1$	–	–
1	$\mathbf{r} = \boldsymbol{\theta} - \mathbf{R}\mathbf{h}_q$,	K	$K(K-1)$
2	if $m = M_b$, algorithm stops	–	–
	else, $m = m + 1$, $d = 2^{-m}$,	–	1
3	$\rho = 1 - \Re(\exp(jd))$	–	–
	$\boldsymbol{\psi} = \rho \cdot \text{diag}\{\mathbf{R}\}$	K	–
4	$\phi_{q,p1} = \phi + d$	–	1
	$\triangle\mathbf{h}_{q,p1} = \exp(j\boldsymbol{\phi}_{q,p1}) - \exp(j\boldsymbol{\phi}_{q,p})$	–	1
	$T_1 = \Re(\triangle\mathbf{h}_{q,p1} \cdot \mathbf{r}_{q,p})$	1	–
	$c_1 = \psi_p + T_1$	–	1
5	if $c_1 < 0$, then goto state 6	–	–
	else, goto state 7	–	–
6	$\mathbf{h}_{q,p1} = \exp(j\boldsymbol{\phi}_{q,p1})$	–	–
	$\mathbf{r} = \mathbf{r} + \triangle\mathbf{h}_{q,p1} \cdot \mathbf{R}(:, p)$	K	K
	$n = n + 1$, Flag = 1	–	–
	if $n = N_u$, algorithm stops	–	–
	else goto state 9	–	–
7	$\phi_{q,p2} = \phi_{q,p} - d$	–	1
	$\triangle\mathbf{h}_{q,p2} = \exp(j\boldsymbol{\phi}_{q,p2}) - \exp(j\boldsymbol{\phi}_{q,p})$	1	1
	$T_2 = \Re(\triangle\mathbf{h}_{q,p2} \cdot \mathbf{r}_{q,p})$	1	–
	$c_2 = \psi_p + T_2$	–	1
8	if $c_2 < 0$, then goto state 9	–	–
	else, goto state 10	–	–
9	$\mathbf{h}_{q,p2} = \exp(j\boldsymbol{\phi}_{q,p2})$	–	–
	$\mathbf{r} = \mathbf{r} + \triangle\mathbf{h}_{q,p2} \cdot \mathbf{R}(:, p)$	K	K
	$n = n + 1$, Flag = 1	–	–
	if $n = N_u$, algorithm stops	–	–
	else goto state 10	–	–
10	$p = p \text{mod} K + 1$	–	1
	if $p = 1$ and Flag = 1, then Flag = 0, goto state 3	–	–
	elsif $p = 1$ and Flag = 0, then goto state 2	–	–
	else, goto state 3	–	–

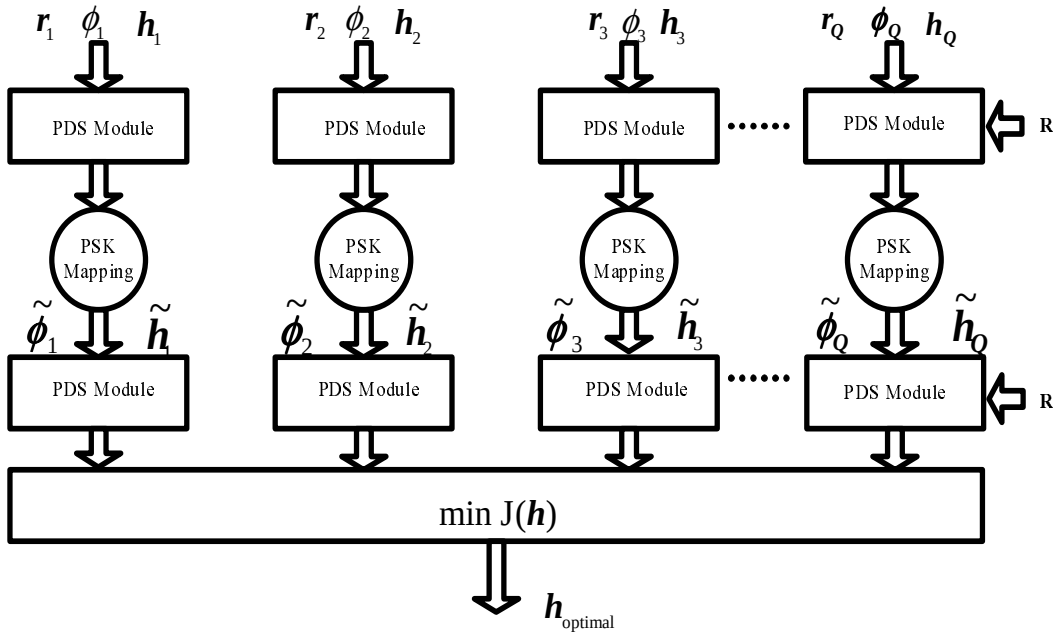


Figure 6.11: Block diagram of the multiple phase decoder.

The first PDS blocks in the branches use different initialization of the phase vector $\boldsymbol{\phi}$ and, accordingly, different initialization of the solution vector $\mathbf{h}$ and the residual vector $\mathbf{r}$. The initial vector $\mathbf{h}_q$ in the qth branch should satisfy the constraint (6.4) and therefore it can be represented as $\mathbf{h}_q = \exp(j\boldsymbol{\phi}_q)$. For a fixed vector $\boldsymbol{\phi}_q$, the initial vector $\mathbf{h}_q$ can be obtained using a look-up table, which contains samples of the complex. The size of the look-up table is as small as 2^{M_b}; *e.g.* for $M_b = 6$, it contains 64 complex numbers. To initialize the residual vector (step 1 in Table 6.3), one has to calculate the matrix-vector product $\mathbf{R}\mathbf{h}_q$, which, in the general case, would require K^2 complex-valued multiplications and $K(K-1)$ complex-valued additions in each branch. This computation is significantly simplified if all elements of the phase vector $\boldsymbol{\phi}_q$ are the same, *i.e.* $\mathbf{R}\mathbf{h}_q = c_q\mathbf{R}\mathbf{1}$, where c_q is a complex-valued constant and $\mathbf{1}$ is a K-length vector of ones. In this case, the vector $\mathbf{R}\mathbf{1}$ is calculated once for all Q branches and this calculation does not require multiplications. Then, for initialization of the residual vector for each branch is as small as K complex-valued multiplications and $K(K-1)$ complex-valued additions. Multiple simulations with different initialization have shown that the use of $c_q = \exp(j\frac{\pi q}{Q})$ provides good detection performance.

Since the PDS algorithm provides phases of the solution, the mapping at step 2 in Table 6.2 becomes a simple operation. It includes quantization of M_b-bit elements of the phase vector $\boldsymbol{\phi}_q$ into $\log_2(M)$-bit elements of the vector $\tilde{\boldsymbol{\phi}}$ and the use of a look-up table of size M to obtain $\tilde{\mathbf{h}}$. These two vectors are used for initialization of the second PDS

blocks in the branches. Note that the proposed choice of the parameters $M_b = 1$ and $\lambda = 1/M$ makes the second PDS simple for implementation and, also, it removes the need for symbol mapping after the second PDS. Our simulation results have shown that, in the second PDS blocks, the parameter N_u limiting the number of updates can be significantly lower than that in the first PDS block.

In the PDS algorithm presented in Table 6.1, there are two stopping criteria: 1) at step 19 upon performing N_u updates; and 2) at step 21 upon achieving a predefined phase resolution $d_0\lambda^{M_b}$. When implementing in hardware (*e.g.* on an FPGA platform), other stopping criteria can be used; *e.g.* the PDS can stop after a predefined number of clock cycles (or execution time). Table 6.1 shows the complexity of different steps of the PDS algorithm in terms of real multiplications and additions, as well as the maximum complexity. At step 3, the values $1 - \cos(d)$ for M_b values of d can be precomputed. Thus, this step only requires K complex-valued multiplications (corresponding to the step 3 in Table 6.3). For transforming phases $\phi_{p,1}$ and $\phi_{p,2}$ into complex numbers $h_{p,1}$ and $h_{p,2}$ (at steps 6 and 12, respectively), a look-up table of size 2^{M_b} can be used as explained above. The maximum complexity corresponds to a scenario where, for every pass, only one successful iteration happens and the PDS algorithm stops at step 19, *i.e.* due to reaching a predefined maximum number of successful updates N_u. If, in a pass, there are several successful iterations and/or the PDS algorithm stops at step 21, *i.e.* due to reaching the predefined phase resolution, the complexity will be lower. In the simulation above, we used N_u high enough to guarantee that the PDS algorithm stops at step 21.

6.8 Conclusions

In this chapter, a novel iterative technique, the phase descent search algorithm, for joint detection of M-PSK symbols has been proposed. The technique provides a solution to the quadratic optimization problem with the constraint that forces elements of the solution to have unit magnitudes. The technique is used multiple times in the proposed multiple phase detector.

The multiple phase detector has been applied to the multiuser detection and MIMO communication. In multiuser detection, the multiple phase detector has been investigated in highly loaded and overloaded scenarios. The numerical results have shown that in highly loaded scenarios, the multiple phase detector has offered a detection performance which

was close to the single-user bound. The multiple phase detector has also shown a better performance and a lower complexity than those of the semi-definite relaxation detector. In the MIMO communication, the multiple phase detector has been shown to offer a performance which was similar to that of the sphere decoder in QPSK modulation. Furthermore, it has been shown that the worst-case complexity of the multiple phase detector slightly varies with SNR, but the variation is insignificant. In addition, its complexity linearly increases with the number of transmit antennas. While at low SNRs, the complexity of the sphere decoder is predicted as exponential. Therefore the multiple phase detector exhibits more favorable performance/complexity characteristics and can be considered as a promising alternative to the sphere decoder for ML decoding in MIMO detection.

This proposed detector is especially designed for the M-PSK symbols detection. We have not run the simulation to show if the proposed detector can provide good detection performance for QAM symbols. This detector might provide good detection performance for QAM symbols after adjusting the searching structure for the QAM constellation.

Chapter 7

Combined multi-user detection with improved "complexity-detection" performance

Contents

7.1 Introduction

In CDMA systems, since multiple users share the same bandwidth to transmit data, user signal may interfere with each other if orthogonality is not maintained and causes Multiple Access Interference (MAI). MAI degrades the performance of the system. Multiuser detection has received a considerable attention as a technique used to solve the problem of MAI. However, the optimal multiuser detection is generally NP (nondeterministic poly-

nomial time)-hard. Since the computational complexity of the optimal multiuser detector grows exponentially as the number of users K increases [110, 111], it is impossible for the optimal receiver to be implemented in a practical system. Consequently, many sub-optimal receivers with a relatively lower computational complexity have been proposed.

The box-constrained dichotomous coordinate descent (DCD) algorithm, which is proposed in chapter 2, provides a good detection performance with a low complexity over all the SNRs, even in highly loaded systems. In addition, this algorithm is multiplication and division free, and so, is efficient for real-time implementation.

The fast optimal algorithm based on branch and bound (BB) method (*i.e.* fast BB) has shown optimal ML detection performance and low average complexity at high SNRs [41]. In multiuser detection, the BB method has two searching approaches which are depth-first search and width-first search. In [41], the popular sphere decoding algorithm has been shown actually a type of the depth-first BB algorithm. However, its worst-case computational complexity is identical to that of the ML optimal multiuser detector, which grows exponentially as K increases. In practice, the worst-case computational complexity is taken into account when the receiver is designed. In order to be implemented in real-time systems, the worst-case computational complexity of a receiver needs to be within the polynomial complexity [112]. Therefore, the worst-case complexity of the BB algorithm prevents it from being implemented in practical systems.

In this chapter, we propose a novel method for multiuser detection, based on combining the BB algorithm and the box-constrained DCD algorithm, to overcome the complexity barrier at low SNRs of the BB detector. The results show that the combined detector significantly reduces the worst-case computational complexity in comparison with that of the BB detector and outperforms the box-constrained DCD algorithm. As a result, the "complexity-detection" performance is improved.

This chapter is organized as follows. In Section 7.2, the problem formulation of the multiuser detection is described. The depth-first branch and bound algorithm, sphere decoding algorithm and the fast branch and bound algorithm are introduced in Section 7.3. In Section 7.4, the detection performance and complexity of the box-constrained DCD detector and the BB detector are presented. In Section 7.5, the combined BB-DCD algorithm is presented. Simulation results and performance analysis are also given in this section. Section 7.6 concludes the chapter.

7.2 Problem Formulation

We assume a K-user synchronous CDMA system using BPSK modulation in an AWGN channel. The matched filter output in the receiver is given by

$$\boldsymbol{\theta} = \mathbf{R}\mathbf{h} + \mathbf{n}\,, \tag{7.1}$$

where $\mathbf{h} \in \{-1, +1\}^K$ denotes the K-length vector of bits transmitted by K users, $\mathbf{R}$ is a $K \times K$ symmetric correlation matrix, $\mathbf{n}$ is a real-valued zero-mean Gaussian random vector with a covariance matrix $\sigma^2\mathbf{R}$. Using Cholesky decomposition, we assume that $\mathbf{R} = \mathbf{L}^T\mathbf{L}$, where $\mathbf{L}$ is a lower triangular matrix (*i.e.* $L_{ij} = 0$ if $i < j$). The system can also be represented by a white noise model

$$\tilde{\boldsymbol{\theta}} = \mathbf{L}^{-T}\boldsymbol{\theta} = \mathbf{L}\mathbf{h} + \mathbf{n}_0\,, \tag{7.2}$$

where $\mathbf{n}_0 = \mathbf{L}^{-T}\mathbf{n}$ is a white Gaussian noise with zero mean and covariance matrix $\sigma^2\mathbf{I}$. The optimal solution of (7.1) is given by

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \{\mathbf{h}^T\mathbf{R}\mathbf{h} - 2\boldsymbol{\theta}^T\mathbf{h}\}. \tag{7.3}$$

The solution of the decorrelating detector [113] is obtained in two steps. First, the unconstrained solution $\tilde{\mathbf{h}} = \mathbf{R}^{-1}\boldsymbol{\theta}$ is computed. Then, it is projected onto the binary constraint set via $\hat{h}_i = \text{sign}(\tilde{h}_i)$.

The decorrelating decision feedback (DF) method is described in [21]. If we denote the ith component of a vector $\boldsymbol{\theta}$ by θ_i, and denote the (i, j)th component of a matrix $\mathbf{A}$ by a_{ij}, the decorrelating DF detector can be characterized by

$$\hat{\mathbf{h}} = \mathbf{P}\tilde{\mathbf{h}}\,; \qquad \tilde{h}_i = \text{sign}\left(\sum_{j=1}^{K} f_{ij}[\mathbf{P}\boldsymbol{\theta}]_j - \sum_{j=1}^{i-1} a_{ij}\tilde{h}_j\right) \tag{7.4}$$

where $\mathbf{F} = U\left([\mathbf{P}\mathbf{R}\mathbf{P}^T]^{-1}\right)$, $\mathbf{A} = L(\mathbf{F}\mathbf{P}\mathbf{R}\mathbf{P})$, $U(\cdot)$ represents the upper triangular part of a matrix, $L(\cdot)$ represents the strictly lower triangular part of a matrix, and $\mathbf{P}$ is a permutation matrix. The choice of $\mathbf{P}$ has been discussed in [21].

7.3 Existing Method

7.3.1 Depth-first Branch and Bound Algorithm

In [41], the relationship between the decorrelating DF, the sphere decoding algorithms and the depth-first BB algorithm has been clearly drawn. It shows that the decorrelating DF detector corresponds to a "one-pass" depth-first BB detector, and the sphere decoding algorithm actually is a type of the depth-first BB detector [41]. For the convenience of understanding the fast BB algorithm, we first present the BB algorithm based on the depth-first search in this section.

Since $\mathbf{R}^{-1} = \mathbf{L}^{-1}(\mathbf{L}^{-1})^T$, equation (7.3) can be written as

$$\begin{aligned} \hat{\mathbf{h}} &= \arg \min_{\mathbf{h}\in\{-1,+1\}^K} (\mathbf{h} - \mathbf{R}^{-1}\boldsymbol{\theta})^T \mathbf{R} (\mathbf{h} - \mathbf{R}^{-1}\boldsymbol{\theta}) \\ &= \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \|\mathbf{L}(\mathbf{h} - \mathbf{R}^{-1}\boldsymbol{\theta})\|_2^2 \\ &= \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \|\mathbf{L}\mathbf{h} - (\mathbf{L}^{-1})^T\boldsymbol{\theta}\|_2^2. \end{aligned} \tag{7.5}$$

We define $\tilde{\boldsymbol{\theta}} = (\mathbf{L}^{-1})^T\boldsymbol{\theta}$ and $\mathbf{T} = \mathbf{L}\mathbf{h}$. We consequently get

$$\begin{aligned} \hat{\mathbf{h}} &= \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \|\mathbf{T} - \tilde{\boldsymbol{\theta}}\|_2^2 \\ &= \arg \min_{\mathbf{h}\in\{-1,+1\}^K} \sum_{k=1}^{K} \left(T_k - \tilde{\theta}_k\right)^2, \end{aligned} \tag{7.6}$$

where T_k and $\tilde{\theta}_k$ are the kth component of $\mathbf{T}$ and $\tilde{\boldsymbol{\theta}}$, respectively. Since $\mathbf{L}$ is a lower triangular matrix, T_k depends only on $(h_1, h_2, \cdots, h_k)$. When the decisions for the first k users are fixed, the term

$$\ell_k = \sum_{j=1}^{k} (T_j - \tilde{\theta}_j)^2 \tag{7.7}$$

can serve as a lower bound of (7.6). When the binary constraints on $(h_{k+1}, h_{k+2}, \cdots, h_K)$ are disregarded, the lower bound can be obtained easily. In the depth-first branch and bound algorithm, the following terms are defined. A node stack is named as OPEN, and a scalar named as UPPER that is equal to the lower bound of the temporary solution. The level of a node is labeled as k, where the virtual root node has the level 0. The branch which connects the two nodes $(h_1, \cdots, h_{k-1})$ and $(h_1, \cdots, h_k)$ is termed as $T_k(h_1, h_2, \cdots, h_k)$. The node $(h_1, \cdots, h_k)$ is termed as the lower bound ℓ_k. The vector $\mathbf{z}_k$

is defined as $\mathbf{z}_k = \tilde{\boldsymbol{\theta}} - \sum_{j=1}^{k} h_j \mathbf{l}_j$, where $\mathbf{l}_j$ is the jth column of $\mathbf{L}$. The jth component of the vector $\mathbf{z}_k$ is denoted as $[z_k]_j$. The depth-first BB algorithm proceeds as follows [39].

1. Order users according to theorem 1 of [21]. Pre-compute $\boldsymbol{\theta}$, $\mathbf{R}$ and $\mathbf{L}$.
2. Precompute $\tilde{\boldsymbol{\theta}} = \mathbf{L}^{-T}\boldsymbol{\theta}$.
3. Initialise $k = 0$, $\mathbf{z}_k = \tilde{\boldsymbol{\theta}}$, $\ell_k = 0$, UPPER$= +\infty$ and OPEN$= NULL$.
4. Set $k = k + 1$. For both nodes, let $\mathbf{z}_k = \mathbf{z}_{k-1}$ and $\ell_k = \ell_{k-1}$. Choose the node in level k such that $h_k = \text{sign}([z_k]_k)$ and set flag $f = 1$.
5. Compute $[z_k]_k = [z_k]_k - h_k l_{kk}$.
6. Compute $\ell_k = \ell_k + (T_k - \tilde{\theta}_k)^2 = \ell_k + [z_k]_k^2$.
7. If $\ell_k \geq$ UPPER, and the OPEN list is not empty, drop this node. Pick the node from the end of the OPEN list, set k, ℓ_k and $\mathbf{z}_k$ equal to the stored values corresponding to this node. Set $f = 0$ and go to step 5.
8. If $\ell_k <$ UPPER and $k < K$, $\forall j > k$ precompute $[z_k]_j = [z_k]_j - h_k l_{jk}$. If $f = 1$, append the other node with $h_k = -\text{sign}([z_k]_k)$ to the end of the OPEN list, and store the associated k, ℓ_k, and $\mathbf{z}_k$ together with this node, go to step 4.
9. If $\ell_k <$ UPPER, $k = K$, and the OPEN list is not empty, update the provisional solution and UPPER$= \ell_k$. Pick the node from the end of the OPEN list, set k, ℓ_k and $\mathbf{z}_k$ equal to the stored values corresponding to this node. Set $f = 0$ and go to step 5.
10. If $\ell_k <$ UPPER, $k = K$, and the OPEN list is empty, update the provisional solution and UPPER$= \ell_k$.
11. Stop and report the provisional solution.

7.3.2 Sphere Decoder

The sphere decoder [50] is a well known efficient algorithm that can achieve the optimal ML performance in the multiuser detection. This decoder is considered as a type of the depth-first BB algorithm and is described as follows [39].

1. Precompute the Cholesky decomposition $\mathbf{R} = \mathbf{L}^T\mathbf{L}$.

2. Precompute $\tilde{\boldsymbol{\theta}}$, $C = \alpha K\sigma^2$, where α is chosen, so that an expected lattice point can be found with a high probability.

3. Initialize $k = 0$, $\mathbf{z}_k = \tilde{\boldsymbol{\theta}}$, $\ell_k = 0$, UPPER $= C$ and OPEN $= NULL$.

4. Set $k = k + 1$. For both nodes, let $\mathbf{z}_k = \mathbf{z}_{k-1}$, $\ell_k = \ell_{k-1}$. Choose the node in level k such that $h_k = -1$. Append the nodes with $h_k = +1$ to the end of the OPEN list, and store the corresponding k, ℓ_k and $\mathbf{z}_k$ together with this node.

5. Compute $[z_k]_k = [z_k]_k - h_k l_{kk}$.

6. Compute $\ell_k = \ell_k + (T_k - \tilde{\theta}_k)^2 = \ell_k + [z_k]_k^2$.

7. If $\ell_k \geq$ UPPER, and the OPEN list is not empty, drop this node. Pick the node from the end of the OPEN list, set k, ℓ_k and $\mathbf{z}_k$ equal to the stored values corresponding to this node and go to step 5.

8. If $\ell_k <$ UPPER and $k < K$, for $j = k + 1$, precompute $[z_k]_j = [z_k]_j - \sum_{i=1}^{k} h_i l_{ji}$. Got to step 4.

9. If $\ell_k <$ UPPER and $k = K$, and the OPEN list is not empty, update the provisional solution and UPPER$= \ell_k$. Pick the node from the end of the OPEN list, set k, ℓ_k, $\mathbf{z}_k$ equal to the stored valued associated with this node and go to step 10.

10. If $\ell_k <$ UPPER, $k = K$, and the OPEN list is empty, update the provisional solution and UPPER$= \ell_k$.

11. If no solution is available yet, let $C = 2C$ and go to step 3. Otherwise, stop and report the provisional solution.

Although, described in a different way, we can see that the sphere decoding algorithm is similar to the depth-first BB algorithm. The main difference between the sphere decoding and the depth-first BB algorithms are in steps 1, 3, 4 and 8. In the depth-first BB algorithm, step1 orders the users. Step 3 initializes the upper bound. The choice of h_k in step 4 of the depth-first BB algorithm is actually the solution of the DF detector. When the system is noise-free, this solution guarantees a minimum computational complexity. Step 8 calculates the lower bound.

7.3.3 Fast Branch and Bound Algorithm

The fast BB algorithm is based on the depth-first BB algorithm, and makes some refinements to speed up the proecessing. These refinements are described as follows.

1. The first feasible solution in the fast BB algorithm is the decision feedback (DF) solution. The decorrelation based DF detector lowers the probability of detection error by applying a successive cancelation technique on users. The choice of user ordering discussed in [21] shows that the easiest or the most powerful user is detected first [15]. In the DF detector, $P_e(k)$ denotes the probability of error on user k and all the decisions on users $1, \cdots, k-1$ are correct. $P_e(k)$ is defined as [21]

$$P_e(k) = Q\left(\frac{l_{kk}}{\sigma}\right), \tag{7.8}$$

where $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$ [114].

In high SNRs, the probability of error of the DF detector solution is dominated by the user corresponding to the minimum diagonal element of $\mathbf{L}$.

2. The user ordering maximizes the probability that the first solution is optimal, however there is still a probability that it is not optimal. In the high SNR region, we denote $P_e(1^{st})$ as the probability of error of the first feasible solution that is represented as

$$P_e(1^{st}) = \sum_{k=1}^{K} Q\left(\frac{l_{kk}}{\sigma}\right). \tag{7.9}$$

We define the indices of the minimum and subsequent (ith) minimums diagonal elements of $\mathbf{L}$, respectively, as

$$\begin{aligned} n_1 &= \arg\min_j l_{jj} \\ n_i &= \arg\min_{j \neq n_1, \cdots, n_{i-1}} l_{jj}\,. \end{aligned} \tag{7.10}$$

If the first given feasible solution is not optimal, user n_1 has a high probability to be an erroneous user because $Q\left(\frac{l_{n_1 n_1}}{\sigma}\right)$ dominates $P_e(1^{st})$. Accordingly, the best option is to change the decision on user n_1 and use the DF detection to obtain the second feasible solution. The probability that neither the first nor the second feasible solution is optimal is given by $P_e(2^{nd}) = \sum_{i \neq n_1} Q(\frac{l_{ii}}{\sigma})$. As a result, we should next search for the n_2th user, and then for the n_3th, n_4th, etc.

3. If a branch on the level k is accepted, then the corresponding sub-branches on levels $k+1, \cdots, k+n$ may also be accepted with a high probability as long as $\forall k < j \leq k+n$, $l_{jj} < l_{kk}$. For user k, we define

$$u_k = \begin{cases} \arg\min_{0<i<k}(\forall i \leq j < k, l_{jj} < l_{kk}) \\ k \qquad \text{if no solution can be found from above} \end{cases} \tag{7.11}$$

$$d_k = \begin{cases} \arg\max_{k<i\leq K}(\forall k < j \leq i\,, l_{jj} \leq l_{kk}) \\ k \qquad \text{if no solution can be found from above} \end{cases} \tag{7.12}$$

Therefore when computing the lower bound ℓ_k, we need to precompute for users $k+1, \cdots, d_k$

$$\forall k < j \leq d_k\,, [z_k]_j = [z_{k-1}]_j - \sum_{i=u_k}^{k} b_i l_{ji} \tag{7.13}$$

i.e. the precomputing involves only the block in $\mathbf{L}$ with rows $[k+1, d_k]$ and columns $[u_k, k]$.

The fast BB algorithm is similar to the depth-first BB algorithm, the user ordering is pre-computed, and all matrices are properly precomputed for the ordered system. In order to implement the new search strategy, instead of using the OPEN stack, the fast BB algorithm has K queues, termed $\mathbf{q}_1, \cdots, \mathbf{q}_K$. Queue $\mathbf{q}_k$ is associated with user k. The nodes in a queue follow the "first-in-first-out" rule, *i.e.* nodes enter from the tail and are taken from the head of the queue. In addition, the queues are ordered according to the values of the diagonal elements of $\mathbf{L}$, *i.e.* in the BB search, the nodes are taken from the queues in the order $[\mathbf{q}_{n_1}, \cdots, \mathbf{q}_{n_K}]$, where $n_1, \cdots, n_K$ are defined in (7.10). The user ordering is corresponding to step 1, the lower bound computations is corresponding to step 8, the upper bound initialization is corresponding to step 3. The choice of h_k in step 4 corresponds to the solution of the DF detector. The full version of the fast optimal BB algorithm is described as following [39, 112].

1. Order users in the ascending order. Compute $\boldsymbol{\theta}$, $\mathbf{R}$ and $\mathbf{L}$; pre-compute the vectors $\mathbf{d}$ and $\mathbf{u}$.
2. Pre-compute $\tilde{\boldsymbol{\theta}} = \mathbf{L}^{-T}\boldsymbol{\theta}$.
3. Initialize $k = 0$, $\mathbf{z}_k = \tilde{\boldsymbol{\theta}}, \ell_k = 0$, UPPER$= +\infty$, and initialize K queues by $\forall k, \mathbf{q}_k = NULL$.

4. Set $k = k + 1$. For both nodes, let $\mathbf{z}_k = \mathbf{z}_{k-1}$, $\ell_k = \ell_{k-1}$, choose the node in level k such that $h_k = \text{sign}([z_k]_k)$. Set flag $f = 1$.

5. Compute $[z_k]_k = [z_k]_k - h_k l_{kk}$.

6. Compute $\ell_k = \ell_k + (T_k - \tilde{\theta}_k)^2 = \ell_k + [z_k]_k^2$.

7. If $\ell_k \geq$ UPPER and not all the queues are empty, drop this node. Goto step 11.

8. If $\ell_k <$ UPPER and $k < K$, do

   - if $f = 1$ for both nodes in level k,
     - If $d_k > k$, precompute $\forall k < j \leq d_k$, $[z_k]_j = [z_k]_j - \sum_{i=u_k}^{k-1} h_i l_{ji}$.
     - If $d_k = k$, precompute $j = k + 1$, $[z_k]_j = [z_k]_j - \sum_{i=u_{k+1}}^{k-1} h_i l_{ji}$.
     - Append the node $h_k = -\text{sign}([z_k]_k)$ to the tail of the queue $\mathbf{q}_{n_k}$, and store the associated k, ℓ, and $\mathbf{z}_k$ together with this node;
   - if $d_k > k$, precompute $\forall k < j \leq d_k$, $[z_k]_j = [z_k]_j - h_k l_{jk}$.
   - if $d_k = k$, pre-compute $j = k + 1$, $[z_k]_j = [z_k]_j - h_k l_{jk}$.
   - Go to step 4.

9. If $\ell_k <$ UPPER, $k = K$ and not all the queues are empty, update the provisional solution and UPPER$= \ell_k$. Go to step 11.

10. If $\ell_k <$ UPPER, $k = K$ and all the queues are empty, update the provisional solution and UPPER $= \ell_k$. Go to step 12.

11. Pick one node from the queues (note that we should check queues in the order of $\mathbf{q}_{n_1} \cdots \mathbf{q}_{n_K}$). Set k, ℓ_k and $\mathbf{z}_k$ equal to the stored values associated with this node. Set $f = 0$, goto step 5.

12. Stop and report the provisional solution.

7.4 Numerical Results for Box-constrained DCD and Fast BB Algorithms

In this section, we present computer simulation results to show the detection performance and the complexity for the box-constrained DCD and fast BB algorithms. We assume

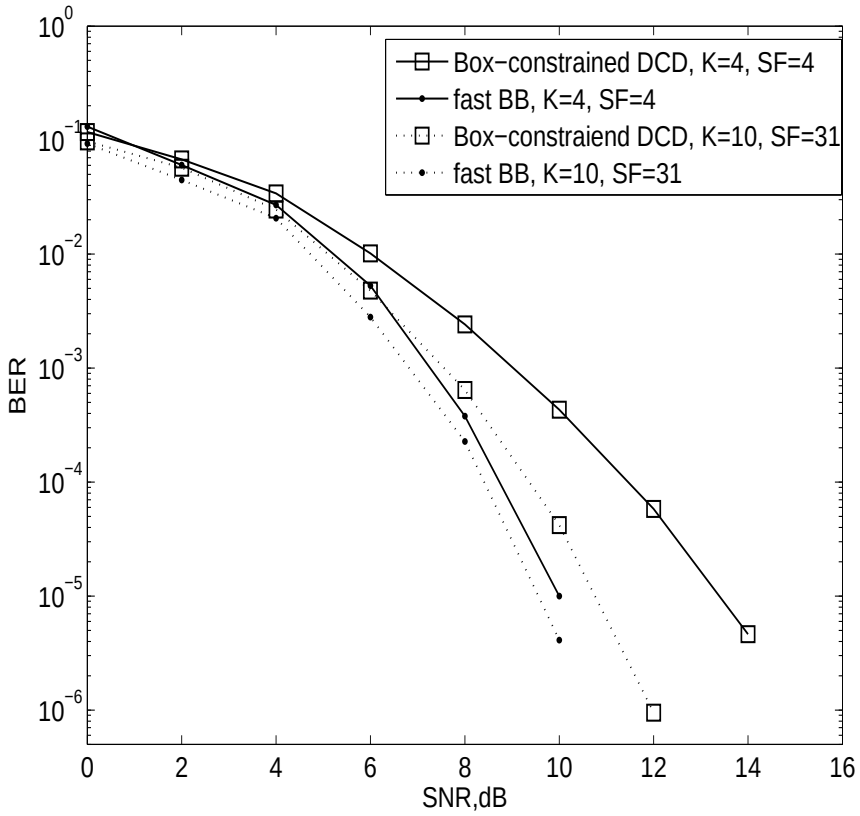


Figure 7.1: BER performance of the box-constrained DCD and BB algorithms in the scenarios of $K = 4$, SF $= 4$ and $K = 10$, SF $= 31$.

the simulation channel is perfect power control with AWGN, where the users employ randomly generated spreading codes. Fig.7.1 shows the BER performance of the box-constrained DCD detector and the fast BB detector. When $K = 4$ and SF $= 4$, the detection performance of the BB detector is better than that of the box-constrained DCD detector. When $K = 10$ and SF $= 31$, the fast BB detector also outperforms the box-constrained DCD detector but does not show much improvement.

Fig.7.2 shows the worst-case complexity of the box-constrained DCD and the fast BB algorithms in the scenarios of $K = 4$, SF $= 4$ and $K = 10$, SF $= 31$. The worst complexity of the box-constrained DCD detector is less than that of the fast BB detector. Furthermore, its worst-case complexity is approximately invariant over the SNRs. Fig.7.2 also presents the fast BB algorithm complexity with the matrix inversion complexity $\mathcal{O}(K^3)$. The results show that the worst complexity of the fast BB detector at low SNRs is almost 10 times of that of the box-constrained DCD detector. In a practical system, transmission is continuous, and the channel is not static, which leads to the difficulty of channel matrix decomposition.

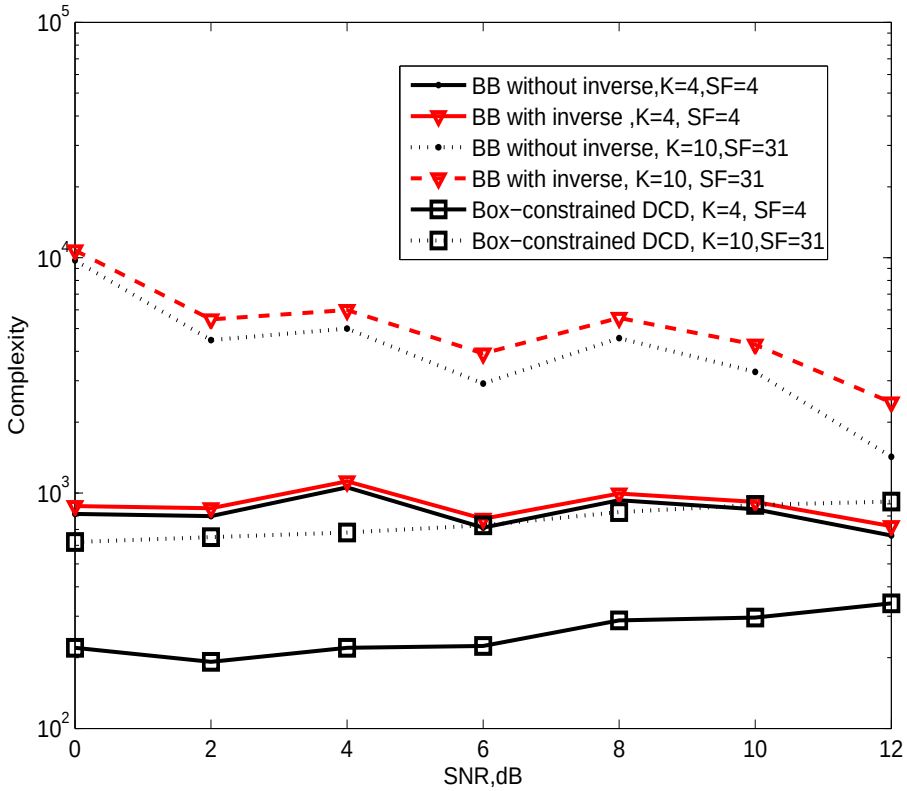


Figure 7.2: Worst complexity of box-constrained DCD algorithm vs. BB algorithm with & without inverse calculation of matrix inverse: (1) $K = 4$, SF $= 4$; (2) $K = 10$, SF $= 31$.

Fig.7.3 shows the complexity (in terms of the number of multiplications and additions) distribution for the fast BB algorithm and the box-constrained DCD algorithm in a highly loaded system with $K = 28$ and SF $= 31$, where X-axis represents complexity and Y-axis represents the number of occurrence. The Fig.7.3$(a\,, c)$ show that when SNR is low *i.e.* SNR $= 7$ dB, the fast BB algorithm has a heavy-tailed complexity distribution, while the complexity distribution of the box-constrained DCD algorithm is concentrated around its mean. Fig.7.3$(b\,, d)$ show that when SNR is high *i.e.* SNR $= 13$ dB, the complexity distribution of the box-constrained DCD algorithm does not significantly change, while the fast BB algorithm has a much more concentrated complexity distribution than the box-constrained DCD algorithm. Therefore, a combination of the fast BB algorithm and the box-constrained DCD algorithm might efficiently solve the problem of the high complexity of the fast BB detector at low SNRs while still provide a performance superior to that of other detectors.

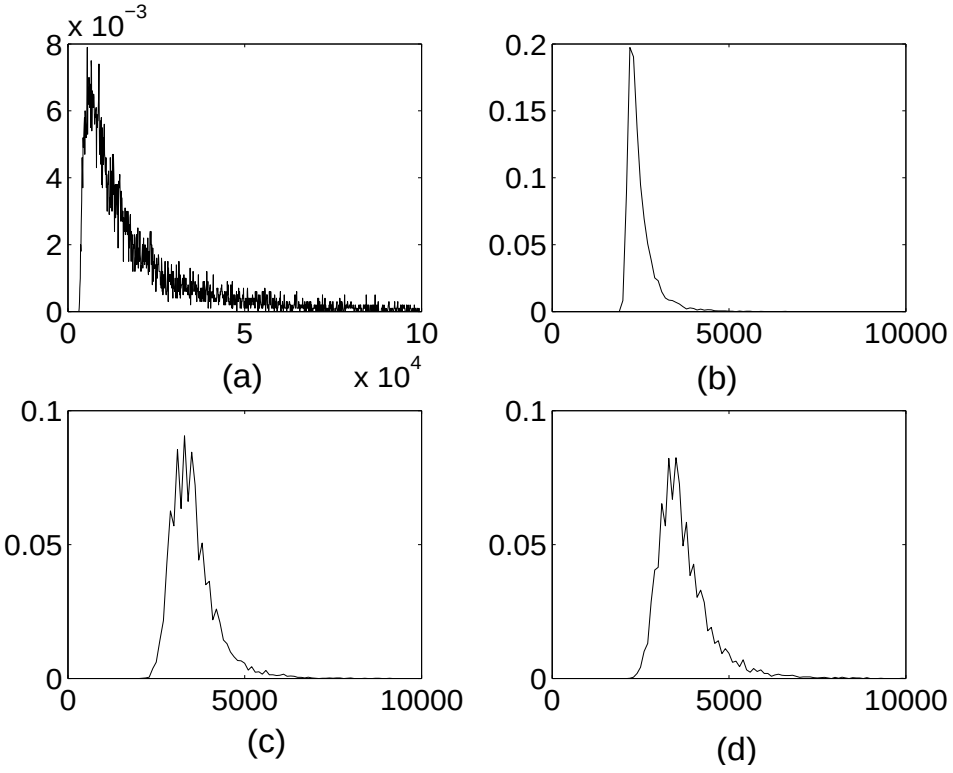


Figure 7.3: Complexity distribution for fast BB and box-constrained DCD detectors:
(a) fast BB detector at SNR $= 7$ dB;
(b) fast BB detector at SNR $= 13$ dB;
(c) box-constrained DCD detector at SNR $= 7$ dB;
(d) box-constrained DCD detector at SNR $= 13$ dB.

7.5 A Combined Detector Based On the Fast BB and Box-constrained DCD Algorithms

Fig.7.4 shows the proposed detector that combines the fast BB algorithm and the box-constrained DCD algorithm using a complexity threshold control. Firstly, the fast BB detector processes the output of the matched filters. The number of the operations (multiplications and additions) is computed. During the whole process, if the number of operations obtained is less than a pre-defined threshold Tr, the solution will be obtained from the fast BB detector. If the number of operations obtained from the fast BB detector is beyond the threshold Tr, then the fast BB detector stops the current detection operation and the box-constrained DCD detector continues until obtaining the box-constrained data estimate $\hat{\mathbf{h}}_{DCD}$. The output data estimate is $\tilde{\mathbf{h}}_{BB-DCD} = \tilde{\mathbf{h}}_{BB}$ in the first case, and $\tilde{\mathbf{h}}_{BB-DCD} = \tilde{\mathbf{h}}_{DCD}$ in the second case.

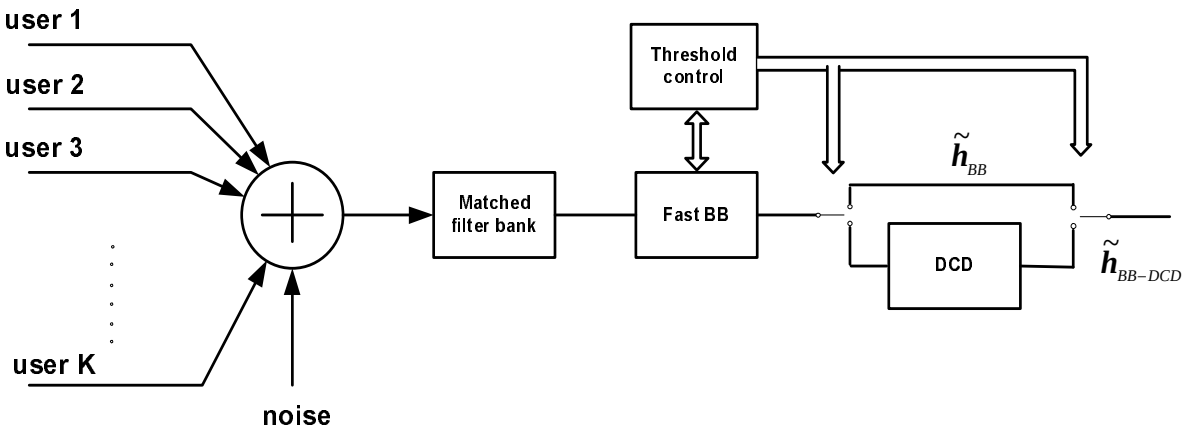


Figure 7.4: Combined fast BB and box-constrained DCD detector.

In addition, we suggest another combined approach which is described as follow. After the matched filter, the first feasible solution of the fast BB detector can be obtained by the box-constrained DCD algorithm instead of using the decorrelating decision feedback detector, thus reducing the complexity and improving the precision of the initial solution. In this chapter, we only investigate the first combination approach.

7.5.1 Simulation Results for The Combined Fast BB and The Box-constrained DCD Algorithms

We consider a high-loaded scenario with $K = 28$ and SF $= 31$ in an AWGN channel. The spreading sequences are randomly generated in every simulation trial. Fig.7.5 compares the performance of the combined BB-DCD detector with various complexity thresholds, with that of the box-constrained DCD detector, and the fast BB detector in terms of group detection error (GDE), which represents the probability that at least one user symbol is incorrect. As the DF detector is one of the simplest receivers, the detection performance of the DF detector is also shown under identical conditions for the purpose of comparison. It can be seen that the DF detector provides a poor performance in the highly-loaded scenario. The box-constrained DCD detector gives a better performance but not as good as that of the fast BB detector. The BB-DCD detector with the threshold Tr $= 10^4$ offers a better performance than the box-constrained DCD detector. With the increase of the complexity threshold by Tr $= 2 \times 10^4$, Tr $= 5 \times 10^4$ and Tr $= 10 \times 10^4$, the GDE performance of the BB-DCD detector is improved and approaches the optimal performance. For the complexity threshold of Tr $= 5 \times 10^4$ FLOPS, the difference compared to the fast BB detector in the performance is 0.1 dB at GDE $= 10^{-5}$.

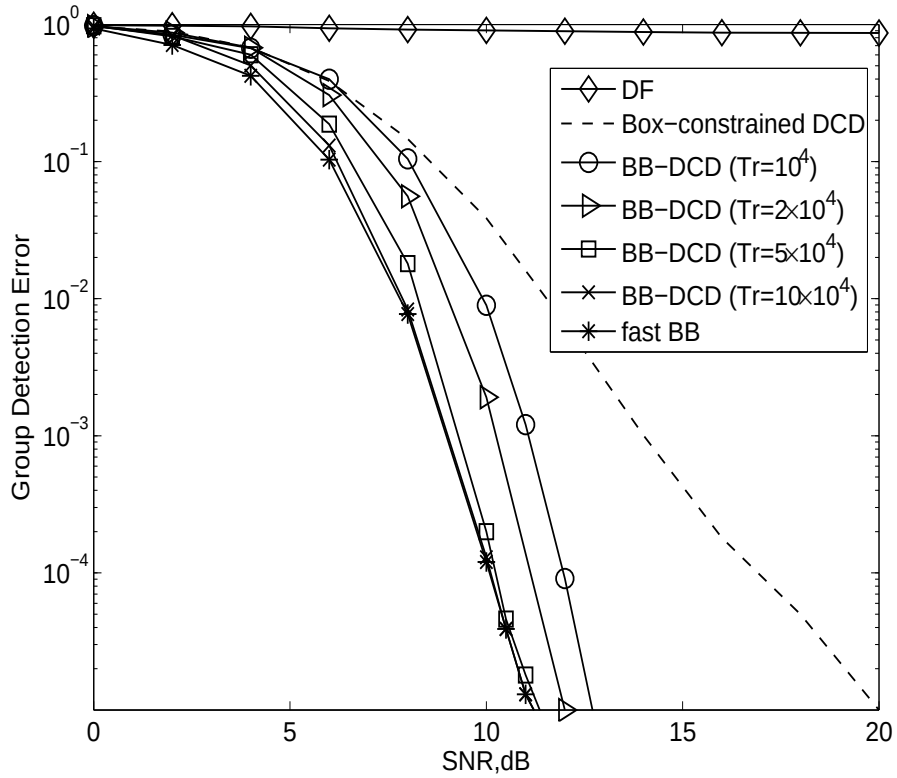


Figure 7.5: Group detection error of the BBD-DCD vs. box-constrained DCD, fast BB and DF; $K = 28$, SF $= 31$.

Fig.7.6 shows the worst-case complexity of the combined detector for this scenario. Though the DF detector offers the lowest complexity compared with other detectors, it performs poorly for all SNRs. The box-constrained DCD detector shows a nearly constant complexity for all SNRs. The worst-case complexity of the fast BB detector increases when the SNR decreases. At SNR $= 0$ dB, its worst-case complexity is 1000 times of complexity of the box-constrained DCD detector. It can be seen that the worst-case complexity of the BB-DCD detector increases when the threshold increases by Tr $= 10^4$, Tr $= 2 \times 10^4$, Tr $= 5 \times 10^4$, Tr $= 10 \times 10^4$ for SNR ≤ 13dB and is close to the complexity of the fast BB detector for SNR ≥ 13 dB.

7.6 Conclusions

The fast BB detector provides the optimal detection performance. However, the worst-case computational complexity of the fast BB detector is prohibitive, which makes the fast BB algorithm difficult for real-time application. The box-constrained DCD algorithm

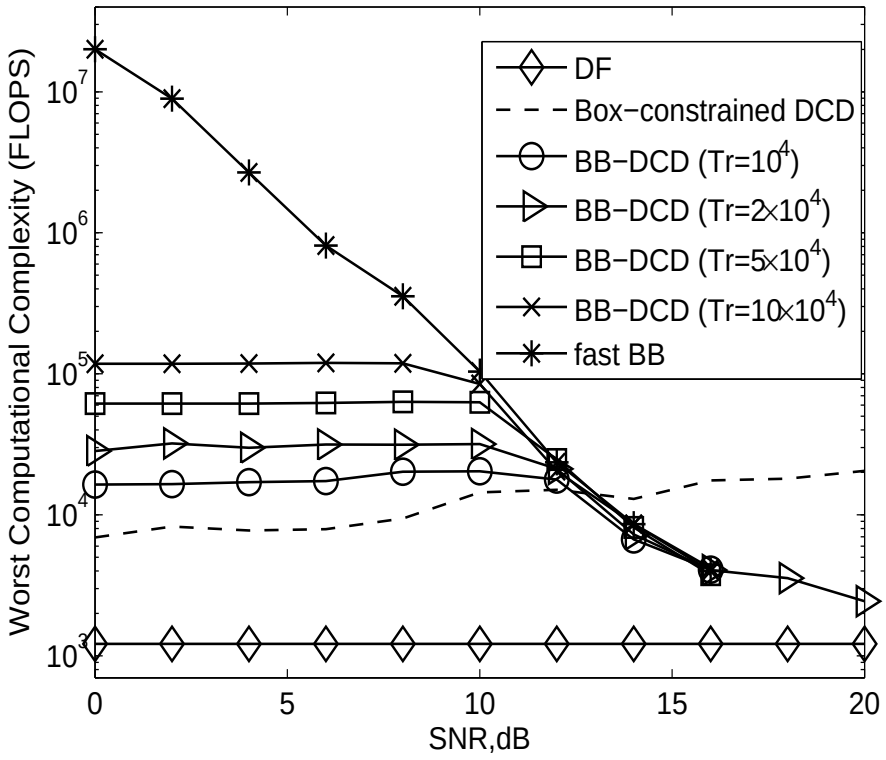


Figure 7.6: Worst-case complexity of the BB-DCD detector vs. the box-constrained DCD, fast BB and DF detectors; $K = 28$, SF $= 31$.

shows a low computational complexity at any value of SNR. However, its detection performance is inferior to that of the fast BB detector. In order to give a good trade-off between the performance and the complexity, we proposed a combination of the fast BB detector and the box-constrained DCD detector. This approach has shown that in a highly loaded scenario, the complexity of the BB-DCD detector is reduced significantly with only a small loss in detection performance with respect to the ML detector, while still better than that of the box-constrained DCD detector.

Chapter 8

The DCD Based Simplified Matrix Inversion for OFDM symbols

Contents

8.1 Introduction

Multiple Input Multiple Output (MIMO) systems have received much attention in recent years as a promising method for next-generation communication systems, because the use of multiple transmit and receive antennas significantly increases the system capacity and diversity [115]. The use of orthogonal frequency-division multiplexing (OFDM) drastically simplifies receiver design in MIMO wireless systems when the channel is frequency selective [116]. In such systems, the linear MIMO receivers including zero-forcing (ZF) and minimum mean square error (MMSE) receivers need to perform channel correlation

inversion. The MMSE detector, however, requires a high computational complexity. The main complexity of the MMSE-based detector is for the inversion calculation. In this chapter, we present an approach based on the DCD algorithm to simplify the inversion operations in MIMO-OFDM systems. The idea of the approach is that the DCD algorithm obtains separately the individual columns of the inverse of the matrix. Due to the low complexity of hardware implementation of the DCD algorithm, a block of DCD processors can be used to obtain the columns of the inverse of the channel correlation matrix in parallel.

In slow fading channels, the DCD-based inverse of the channel correlation matrix needs only to be performed in one OFDM symbol, and then the same solution can be used for other symbols over an OFDM frame. However, in fast fading channels, the DCD-based inverse needs to be performed separately for each symbol in the OFDM frame.

In the frequency-selective channels, the channel is decomposed into parallel flat channels in the frequency domain. The change of the channel frequency response can be slow between neighbouring subcarriers especially if the multipath delay spread is significantly smaller than the duration of an OFDM symbol. The DCD-based inverse of the channel correlation matrix obtained from the first-subcarrier can be used as an initialization for that of the second subcarrier, and so on, up to that of the last sub-carrier. This reduces the complexity compared to the case where the inverse of the channel correlation matrix is initialized to zero for each subcarrier.

This chapter is organised as follows. In Section 8.2, the MIMO-OFDM model is presented. In Section 8.3, two detectors which require matrix inversion, are introduced. In Section 8.4, the DCD-based matrix inversion for MIMO-OFDM symbol is proposed. Various situation results of this inversion approach are presented in Section 8.5. In Section 8.6, the DCD-based inversion is applied to the underwater acoustic communication. Section 8.7 concludes the chapter.

8.2 MIMO-OFDM Model

We consider a MIMO-OFDM system with M_T transmit antennas, M_R receive antennas and K subcarriers. We assume that the frequency selective fading channels between each pair of transmit and receive antennas have L independent delay paths and the same

power delay profile. In frequency-selective channels, OFDM is effective in mitigating the intersymbol interference (ISI) caused by multipath. This is achieved by converting the frequency selective channel into a set of parallel narrow-band flat fading channels, or subcarriers. Therefore, the frequency-response of each narrow-band channel can be assumed to be constant. In the MIMO systems, the received signal in the kth subcarrier is presented as follows:

$$y_k^u = \sum_{v=1}^{M_T} G_k^{u,v} h_k^v + n_k^u \,, \tag{8.1}$$

where h_k^v denotes the transmitted symbols at the kth subcarrier of the vth transmit antenna, $G_k^{u,v}$ denotes the channel response of the kth subcarrier between the uth receive antenna and the vth transmit antenna. The channel frequency response is given by

$$G_k^{u,v} = \sum_{l=0}^{L-1} g_l^{u,v} e^{-j2\pi\tau_l^{u,v}k/T} \,, \tag{8.2}$$

where $1/T$ is subcarrier spacing and T is the OFDM symbol duration without a cyclic prefix, $g_l^{u,v}$ is the zero-mean complex Gaussian random variables with a power-delay profile ϑ_k. In this work, two power-delay profiles are considered: uniform and exponential power delay profiles. The exponential power delay profile is given by $\vartheta_k = e^{-\tau_k/\tau_{rms}}$, where τ_{rms} is the root-mean square width of ϑ_k. The time delay at lth multipath $\tau_l^{u,v}$ is uniformly and independently distributed. n_k^u denotes the additive complex-valued white Gaussian noise (AWGN) with zero mean and variance σ^2. Hereafter, the received signal can simply be written as:

$$\mathbf{y}_k = \mathbf{G}_k \mathbf{h}_k + \mathbf{n}_k \,, \tag{8.3}$$

where $\mathbf{y}_k = [y_k^1, \cdots, y_k^{M_R}]^T$, the (u,v)th element of $\mathbf{G}_k$ is $G_k^{u,v}$, $\mathbf{h}_k = [h_k^1, \cdots, h_k^{M_T}]^T$ and $\mathbf{n}_k = [n_k^1, \cdots, n_k^{M_R}]^T$.

8.3 MIMO OFDM Detection Scheme

Linear detection methods invert the channel correlation matrix when using a zero forcing (ZF) or minimum mean squared error (MMSE) criterion. The matrix representation of the detection scheme at the kth subcarrier is denoted as $\mathbf{J}_k = \{J_k^{u,v}\}$.

In a ZF linear detector, the output of the matched filter is multiplied with the matrix

$$\mathbf{J}_k = {\mathbf{R}_k}^{-1} = ({\mathbf{G}_k}^H \mathbf{G}_k)^{-1} \,, \tag{8.4}$$

which is an inverse of the channel correlation matrix for the kth subcarrier.

The MMSE detector minimizes the mean square error between the actually transmitted symbols and the output of the linear detector that is implemented by using the matrix

$$\mathbf{J_k} = \mathbf{R}_k{}^{-1} = (\mathbf{G}_k{}^H \mathbf{G}_k + \sigma^2 \mathbf{I})^{-1}, \tag{8.5}$$

which is an inverse of the diagonally loaded channel correlation matrix the kth subcarrier, where σ^2 indicates the noise variance and $\mathbf{I}$ is an $M_T \times M_T$ identity matrix.

We can estimate the data in a subcarrier of an OFDM transmitted symbol by linear detection as

$$\mathbf{h}_k = D(\mathbf{J}_k \cdot \boldsymbol{\theta}_k), \tag{8.6}$$

where $D(\cdot)$ is the detection operation for the constellation and $\boldsymbol{\theta}_k = \mathbf{G}_k{}^H \mathbf{y}_k$. As a result, the detection problem can be converted into the problem of finding the inverse of the matrix $\mathbf{R}_k$.

8.4 DCD-based Inversion for MIMO-OFDM Systems

In frequency selective MIMO channels, the channel can be decomposed into parallel flat channels in the frequency domain. The change of the channel frequency response is usually slow between neighbouring subcarriers. The matrix $\mathbf{J}_k$, $(k = 1, \cdots, K)$ obtained from the kth subcarrier can be used as an initialization for that of the $(k+1)$th subcarrier, and so on, up to that of the last subcarrier. The DCD algorithm can obtain separately the individual columns of the inverse of the matrix $\mathbf{R}_k$. Due to the low complexity of the DCD algorithm, a block of DCD processors can be used to obtain the columns of the inverse of the $\mathbf{R}_k$ simultaneously.

Table 8.1 shows the DCD algorithm for solving the inverse of $\mathbf{R}_k$ in one OFDM symbol, where $\mathbf{J}_k^{(j)}$ and $\mathbf{I}^{(j)}$ are the jth columns of the matrices $\mathbf{J}_k$ and $\mathbf{I}$, respectively. In the first sub-carrier $(k = 1)$, the initialization vector $\bar{\mathbf{J}}_k^{(j)}$ is set to $\mathbf{0}$. N_{u_1} is the number of the successful iterations for the first sub-carrier. This number is set to a relatively large number (*i.e.* $N_u = 600$). The $M_T \times M_T$ matrix $\mathbf{R}$ is set to $\mathbf{R}_k$. Then, the DCD algorithm obtains separately the individual columns of the matrix $\hat{\mathbf{J}}_k$, which is an estimate of the inverse matrix $\mathbf{R}_k$ by using $\bar{\mathbf{J}}_k^{(j)}$, N_u, $\mathbf{R}$ and $\mathbf{I}^{(j)}$. For the other subcarriers $(k > 1)$, $\mathbf{J}_k^{(j)}$

Table 8.1: DCD-based inversion algorithm for an OFDM symbol

Step	operation
1	$k = 1$ for $j = 1 : M_T$ $\bar{\mathbf{J}}_k^{(j)} = \mathbf{0}$, $N_u = N_{u_1}$, $\mathbf{R} = \mathbf{R}_k$ $\hat{\mathbf{J}}_k^{(j)} = \text{DCD}(\bar{\mathbf{J}}_k^{(j)}, \mathbf{R}, \mathbf{I}^{(j)}, N_u, M_b)$ j-loop end
2	for $k = 2 : K$ for $j = 1 : M_T$ $\bar{\mathbf{J}}_k^{(j)} = \hat{\mathbf{J}}_{k-1}^{(j)}$, $N_u = N_{u_2}$, $\mathbf{R} = \mathbf{R}_k$ $\hat{\mathbf{J}}_k^{(j)} = \text{DCD}(\bar{\mathbf{J}}_k^{(j)}, \mathbf{R}, \mathbf{I}^{(j)}, N_u, M_b)$ j-loop end k-loop end

is initialized to the solution for the previous subcarrier (*i.e.* $\bar{\mathbf{J}}_k^{(j)} = \mathbf{J}_{k-1}^{(j)}$). In this case, N_{u_2} can be chosen as a relatively small value; *i.e.* smaller than N_{u_1}. The matrix $\mathbf{R}$ is set to the corresponding $\mathbf{R}_k$. By using the $\bar{\mathbf{J}}_k^{(j)}$, N_u, $\mathbf{R}$ and $\mathbf{I}^{(j)}$, the DCD algorithm obtains separately individual columns of the matrix $\hat{\mathbf{J}}_k$.

The inverse matrix $\mathbf{J}_k$ corresponding to each subcarrier can be obtained by using one DCD processor. However, in order to speed up the operation, we can also use a block of M_T DCD processors to obtain the M_T columns of $\mathbf{J}_k$ simultaneously. Fig.8.1 presents a block of M_T DCD processors performing matrix inversion for an OFDM symbol. The inverse of the matrix $\mathbf{R}_k$ is divided into M_T columns. For the first subcarrier, $\mathbf{J}_1^{(j)} = \mathbf{0}$, $\mathbf{I}^{(j)}$, $\mathbf{R}_1$ and N_{u_1} are sent to the jth DCD processor, where $j = 1, 2, \cdots, M_T$. All the M_T DCD processors are used for computing the first solution $\mathbf{J}_1$ corresponding to the first subcarrier. For the successive subcarriers, the same M_T DCD processors are used consecutively for obtaining the inverse matrix $\mathbf{J}_k$ corresponding to the kth subcarrier, where $k = 2, 3, \cdots, K$. For each subcarrier k, $\mathbf{J}_{k-1}$ (*i.e.* initialization from the previous solution), $\mathbf{I}$, $\mathbf{R}_k$ and N_{u_2} are sent to the block of DCD processors. In comparison with the implementation using one DCD processor, a block of M_T DCD processors performs the inversion operation M_T times faster.

The estimated number of slices needed for the FPGA implementation of the DCD-based ($M_b = 15$) inversion algorithm in each subcarrier for 4×4 MIMO systems, are presented in Table 8.2. The estimation is according to the resources required for the FPGA implementation of a single complex-valued DCD algorithm, which has been presented in [117]. The FPGA implementation of the complex-valued DCD algorithm [117] re-

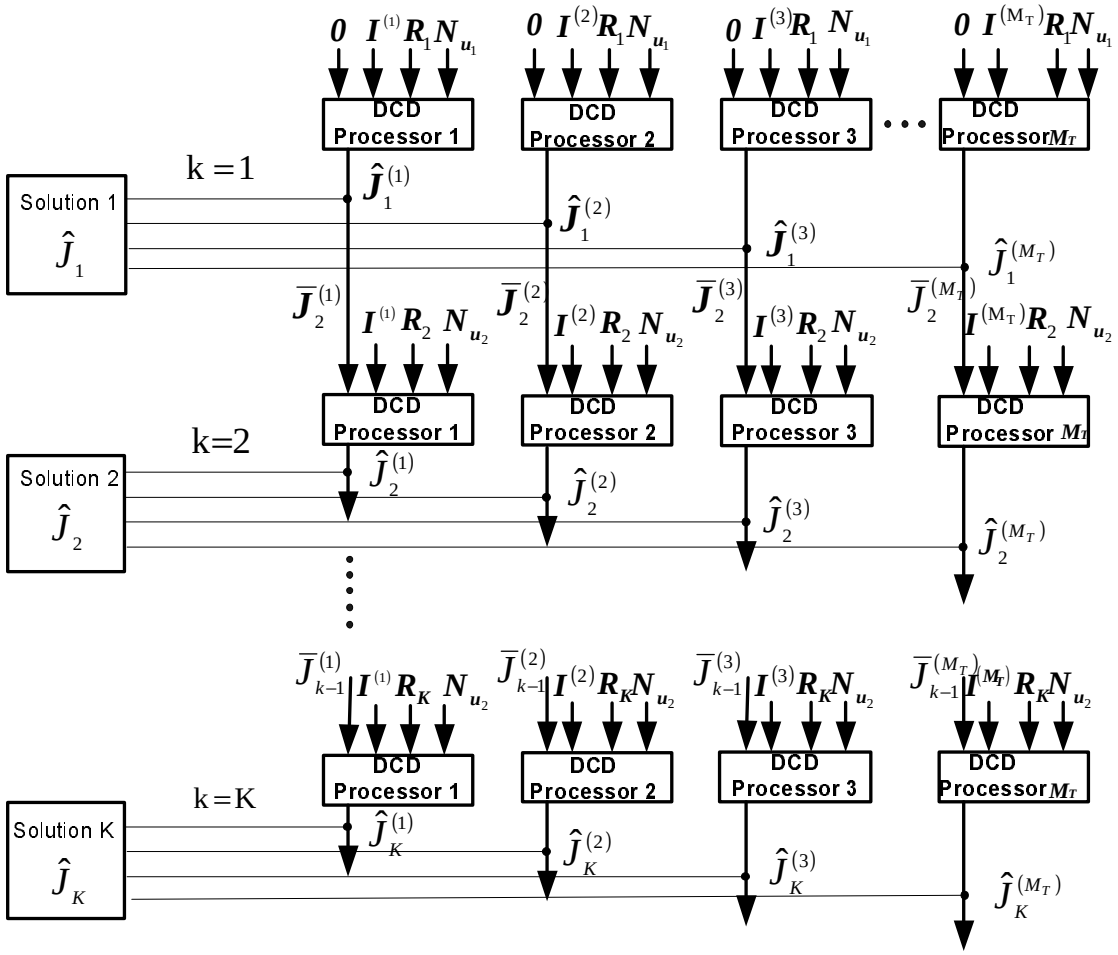


Figure 8.1: A block of M_T DCD processors performing matrix inversion for an OFDM symbol.

Table 8.2: FPGA resources required for each subcarrier for parallel DCD-based inversion algorithms

Resources	$M_T = 4$
Slices	2344 (17%)
D-FFs	193 (2.8%)
LUT4s	2388(8.8%)
Block RAMs	3 (2.2%)

quires 586 slices of the FPGA chip (Xilinx XC2VP30). Therefore, we estimate that for the 4×4 MIMO system, a block of 4 DCD processors requires $586 \times 4 = 2344$ slices. In comparison with the matrix inversion for 4×4 MIMO systems reported in [85, 118], and requiring 9117 and 9474 slices, respectively, the proposed approach has a significantly lower slice usage.

8.5 Simulation Results

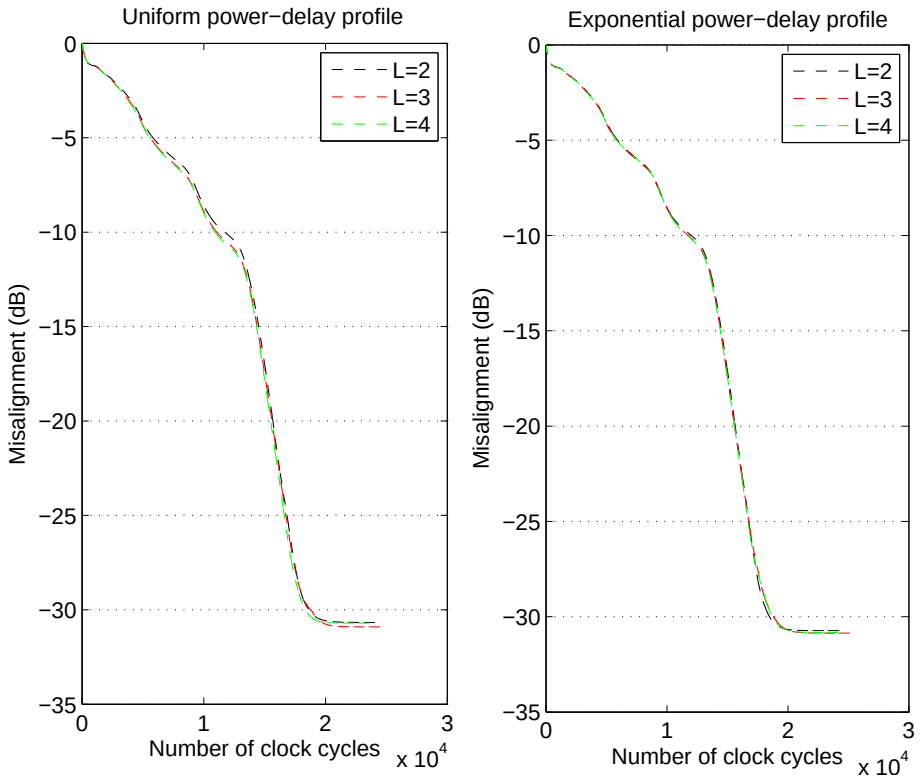


Figure 8.2: Effect of the number of multipath L: the misalignment vs. the number of clock cycles in the 1st sub-carrier; $M_T = 4$, $K = 128$, $N_{u_1} = 400$.

By solving (8.5), the DCD algorithm obtains a solution $\hat{\mathbf{J}}$. The misalignment between matrices $\hat{\mathbf{J}}_k$ and $\mathbf{J}_k$ is calculated as

$$\xi_k = \frac{||\hat{\mathbf{J}}_k - \mathbf{J}_k||^2}{||\mathbf{J}_k||^2}. \tag{8.7}$$

For many simulation trials, the misalignment is averaged over a number of $T = 10^4$ simulation trials and is given in decibels by

$$\bar{\xi}_k = 10 \lg \left\{ \frac{1}{T} \sum_{t=1}^{T} \frac{\sum_{j=1}^{M_T} \sum_{n=1}^{M_T} |\hat{J}_k(n,j) - J_k(n,j)|^2}{\sum_{j=1}^{M_T} \sum_{n=1}^{M_T} |J_k(n,j)|^2} \right\}. \tag{8.8}$$

In a 4×4 MIMO-OFDM system with frequency selective channel, we assume that the DCD-based inversion is evaluated for a uniform power-delay profile and an exponential power-delay profile at SNR $= 20$ dB.

First we investigate the effect of the number of paths L on the misalignment in Fig.8.2, Fig.8.3, Fig.8.4 and Fig.8.5. We set $M_T = 4$, $K = 128$, $N_{u_1} = 400$, $N_{u_2} = 10$.

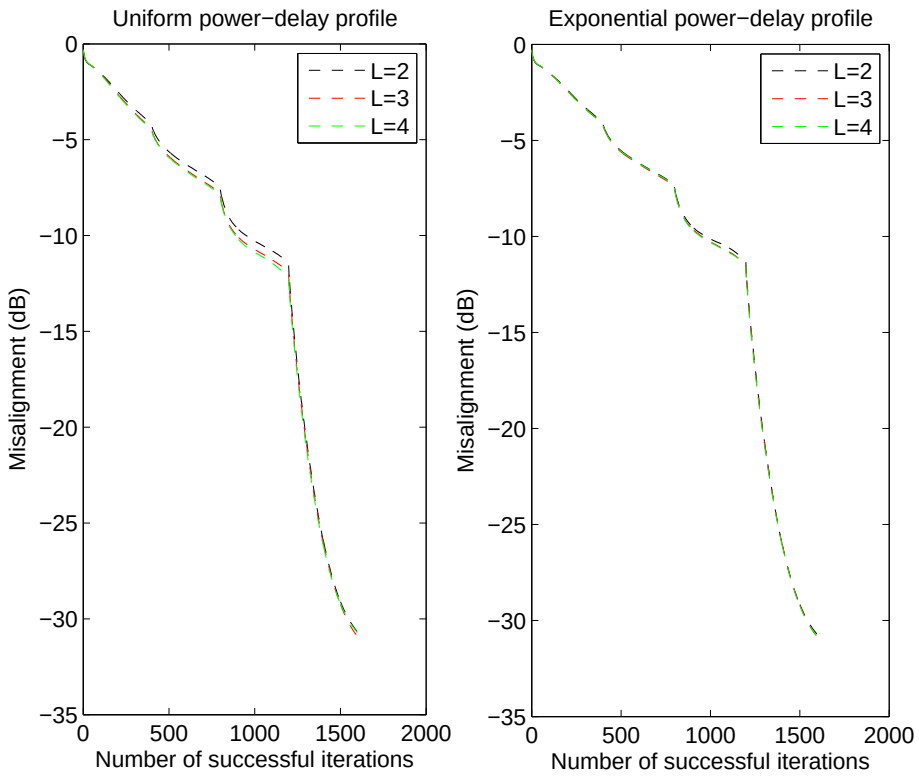


Figure 8.3: Effect of the number of multipath L: the misalignment vs. the number of successful iterations for the 1st sub-carrier; $M_T = 4$, $K = 128$, $N_{u_1} = 400$.

Fig.8.2 shows the misalignment against the number of clock cycles for the 1st sub-carrier. The results show that the various multipath L do not significantly affect the inversion

misalignment in the 1st sub-carrier. The inversion requires approximately $20K$ clock cycles to achieve the misalignment of -32 dB.

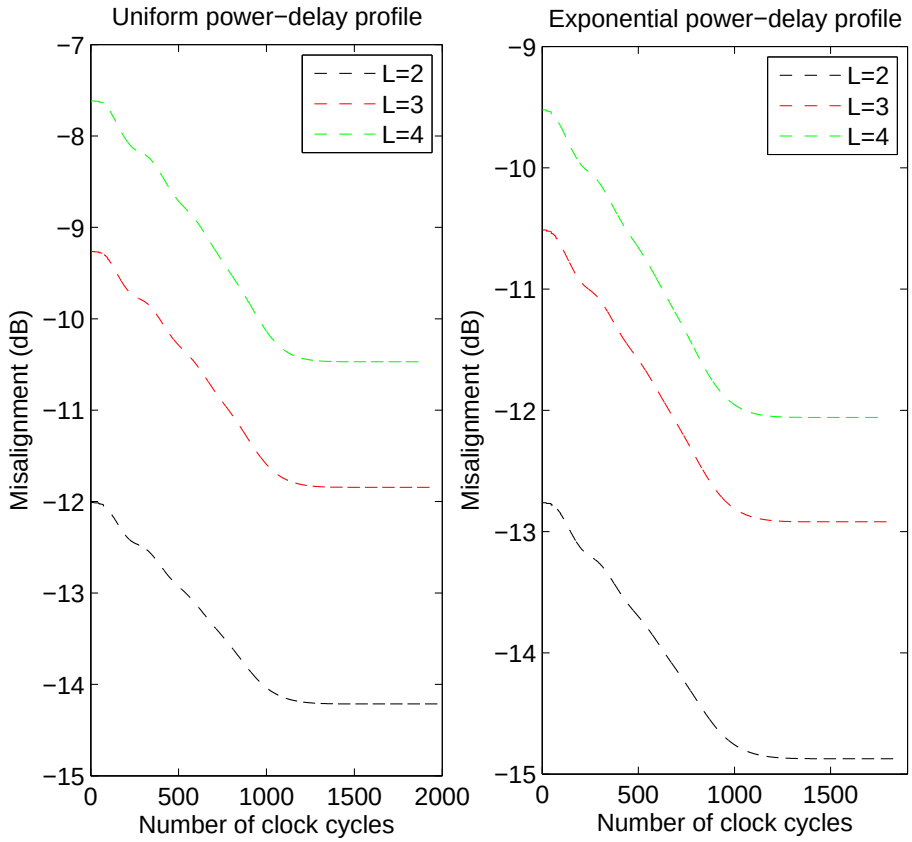


Figure 8.4: Effect of the number of multipath L: the misalignment vs. the number of clock cycles for the $k > 1$ sub-carriers ; $M_T = 4$, $K = 128$, $N_{u_1} = 400$, $N_{u_2} = 10$.

Fig.8.3 shows the misalignment against the total number of successful iterations for the 1st sub-carrier. We assume $N_{u_1} = 400$ for the DCD processor, a 4×4 matrix inversion requires at most 400×4 successful iterations, to achieve the misalignment of -32 dB. In a channel with an exponential power-delay profile or uniform power-delay profile, the misalignments are not significantly affected by the number of L.

Fig.8.4 shows the misalignment against the number of clock cycles for every other ($k > 1$) sub-carrier. The results show that when L increases (*i.e.* $L = 2, 3, 4$), the misalignment is increased. In a channel with a uniform power-delay profile, when $L = 2$ the proposed inversion approach obtains the misalignment of -14 dB using approximately 1000 clock cycles. In a channel with an exponential power-delay profile, when $L = 2$ it obtains the misalignment of -15 dB using approximately 1000 clock cycles.

Fig.8.5 shows the misalignment against the number of iterations for every other ($k > 1$)

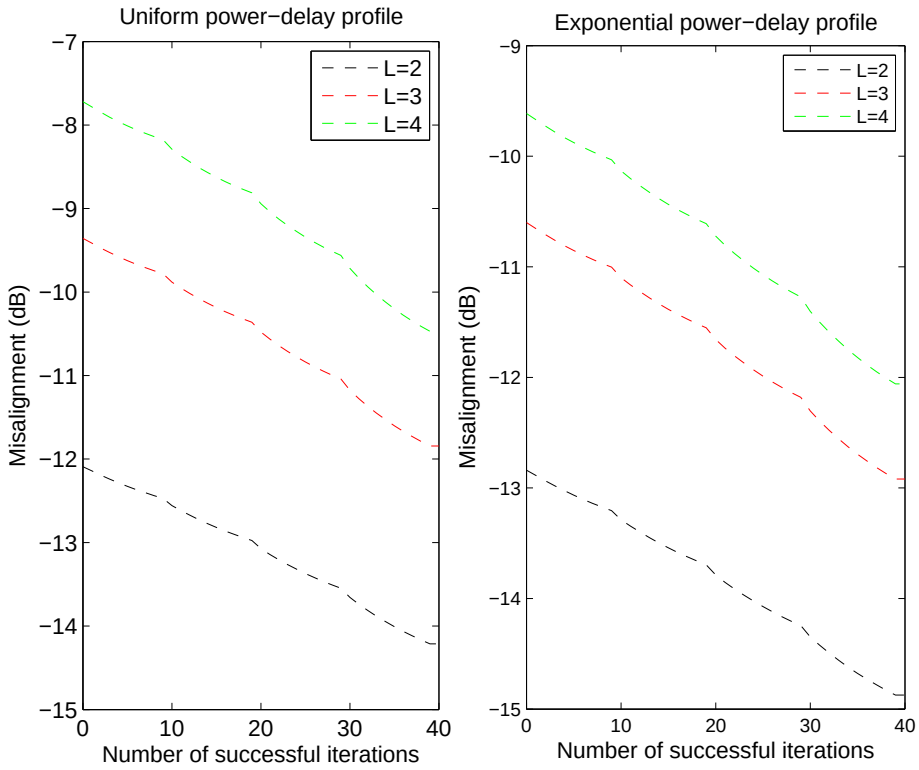


Figure 8.5: Effect of the number of multipath L: the misalignment vs. the number of successful iterations for every other ($k > 1$) sub-carrier; $M_T = 4$, $K = 128$, $N_{u_1} = 400$, $N_{u_2} = 10$.

sub-carrier. The results show that when L increases, the misalignment of the DCD-based inversion is increased. In a channel with a uniform power-delay profile, when $L = 2$ the proposed inversion obtains the misalignment of -14 dB using 40 iterations. In a channel with an exponential power-delay profile, when $L = 2$ the proposed inversion obtains the misalignment of -15 dB.

Secondly, we investigate the effect of the number of sub-carriers K on the misalignment, which are presented in Fig.8.6, Fig.8.7, Fig.8.8 and Fig.8.9. We set $M_T = 4$, $L = 4$, $N_{u_1} = 400$ and $N_{u_2} = 10$.

Fig.8.6 shows the misalignment against the number of clock cycles in the 1st sub-carrier. The results show that the misalignments of the DCD-based inversion in the 1st sub-carrier are not affected by the different number of K. Using approximately 2×10^4 clock cycles, the proposed inversion approach in the channel with a uniform power-delay profile or an exponential power-delay profile, obtains the misalignment of -32 dB.

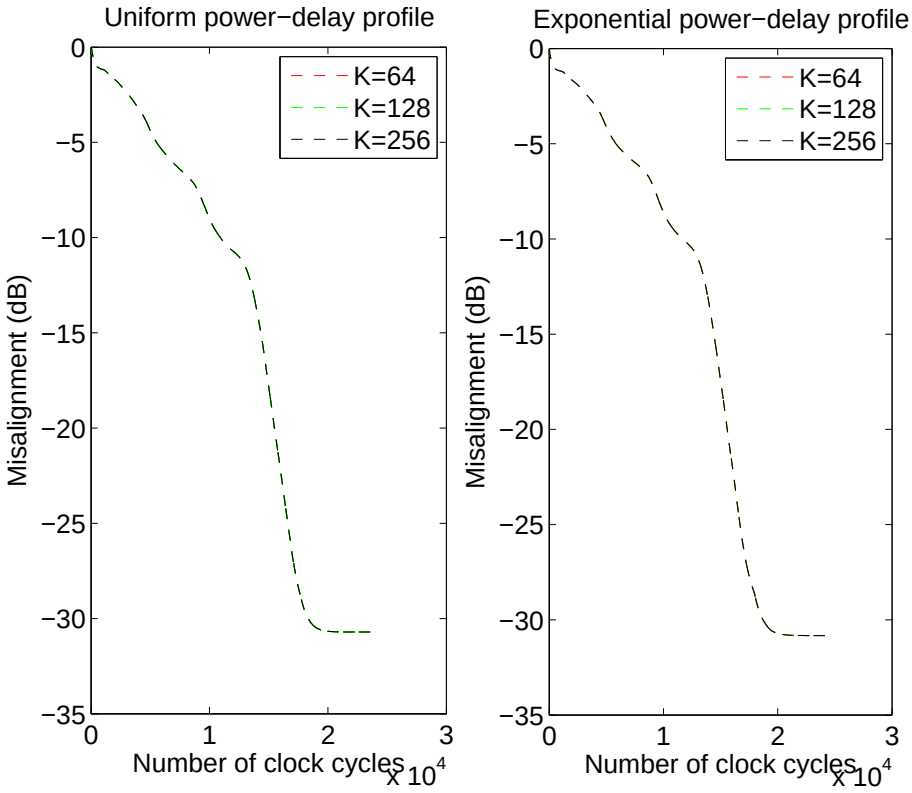


Figure 8.6: Effect of the number of sub-carriers K on the misalignment: the misalignment vs. the number of clock cycles for the 1st sub-carrier; $M_T = 4$, $L = 4$, $N_{u_1} = 400$.

Fig.8.7 presents the misalignment against the total number of successful iterations for the 1st sub-carrier. The results show that the misalignments of the DCD-based inversion in the 1st sub-carrier are not affected significantly by the different number of K. The misalignment of the DCD-based inversion could achieve -32 dB using approximately 1600 successful iterations. The misalignment of the DCD-based inversion in the channel with an exponential power-delay profile is smaller than that of it in the channel with a uniform power-delay profile.

Fig.8.8 shows the misalignment against the number of clock cycles for every other ($k > 1$) sub-carrier. The results show that increase in K (*e.g.* $K = 64, 128, 256$), the misalignment of the DCD-based inversion is reduced. The results also show that the misalignment obtained in a channel with an exponential power-delay profile is smaller than that of it in the channel with a uniform power-delay profile with the same K.

Fig.8.9 presents the misalignment against the number of iterations for every other ($k > 1$) sub-carrier. The results show that increase in K, the misalignment of the DCD-based inversion is reduced. By using 40 successful iterations in total, the DCD-based inversion

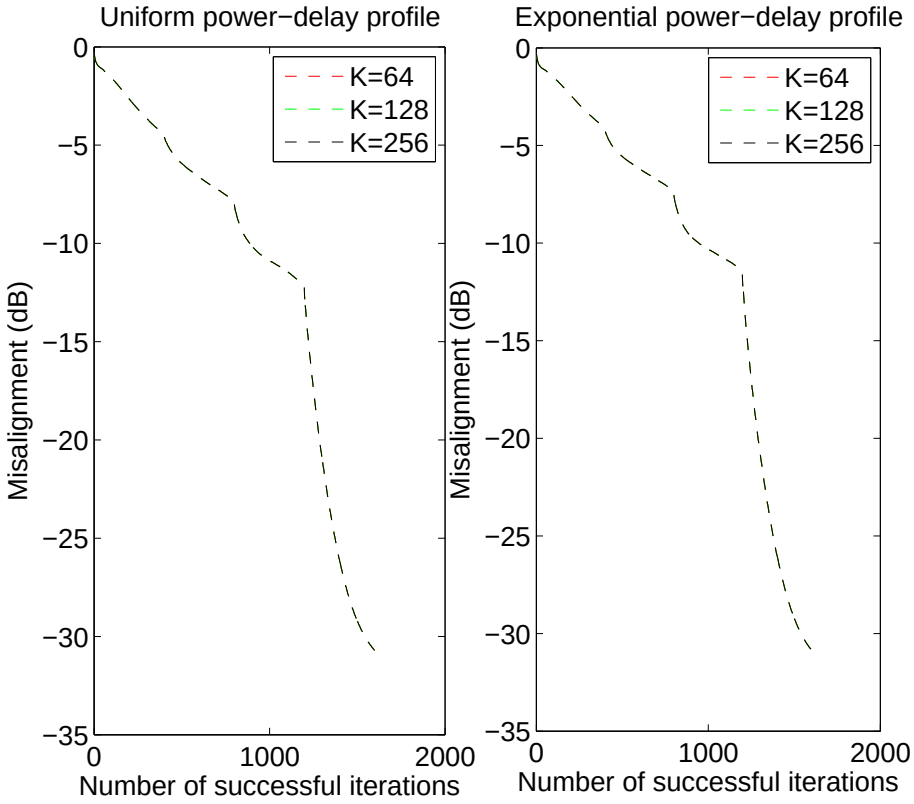


Figure 8.7: Effect of the number of sub-carriers K on the misalignment: the misalignment vs. the number of successful iterations for the 1st sub-carrier; $M_T = 4$, $L = 4$, $N_{u_1} = 400$.

algorithm in the channel with a uniform power-delay profile obtains the misalignment of -14 dB when $K = 256$; the DCD-based inversion in a channel with an exponential power-delay profile obtains the misalignment of -15 dB when $K = 256$.

We evaluate the effects of N_{u_1} on the misalignment. We set $M_T = 4$, $K = 128$, $L = 4$ and $N_{u_2} = 10$. Fig.8.10 and Fig.8.11 show the misalignment against the number of clock cycles for the 1st sub-carrier, and every other ($k > 1$) sub-carrier, respectively, with the different N_{u_1}. Fig.8.12 and Fig.8.13 show the misalignment against the number of total iterations used in the 1st sub-carrier, and every other ($k > 1$) sub-carrier, respectively. We can see that increase in N_{u_1}, the misalignment of the DCD-based inversion is reduced in the 1st sub-carrier, however, the misalignments in every other ($k > 1$) sub-carrier are not affected significantly. By using the same number of iterations or clock cycles, the misalignment of the DCD-based inversion in the channel with an exponential power-delay profile is 2 dB lower than that of it in the channel with a uniform power-delay profile.

Fig.8.14 shows the misalignments against the number of clock cycles for every other ($k >$

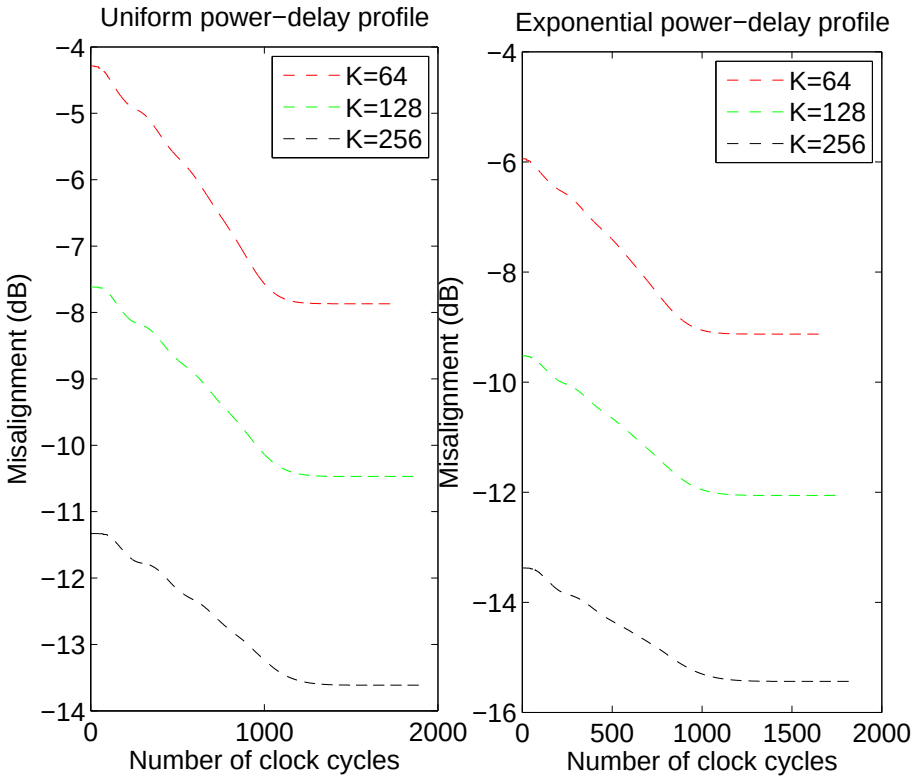


Figure 8.8: Effect of the number of sub-carriers K on the misalignment: the misalignment vs. the number of clock cycles for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, N_{u_1}=400, $N_{u_2} = 10$.

1) sub-carrier with the different number of N_{u_2}. Fig.8.15 show the misalignments against the number of total iterations used for every other ($k > 1$) sub-carrier with different N_{u_2}. We can see that increase in N_{u_2}, the misalignment of the DCD-based inversion is reduced for every other ($k > 1$) sub-carrier.

8.6 Discussion on application of the DCD based matrix inversion

According to the simulation results above for an OFDM symbol with 128 sub-carriers, we can see that the inversion in the first sub-carrier approximately requires 2×10^4 clock cycles using $N_{u_1} = 400$. The inversion for every other ($k > 1$) sub-carrier, approximately requires 10^3 clock cycles when using $N_{u_2} = 10$. Therefore, the inversion for a complete OFDM symbol needs $10^3 \times 127 + 2 \times 10^4 \approx 1.47 \times 10^5$ clock cycles in total. A processor

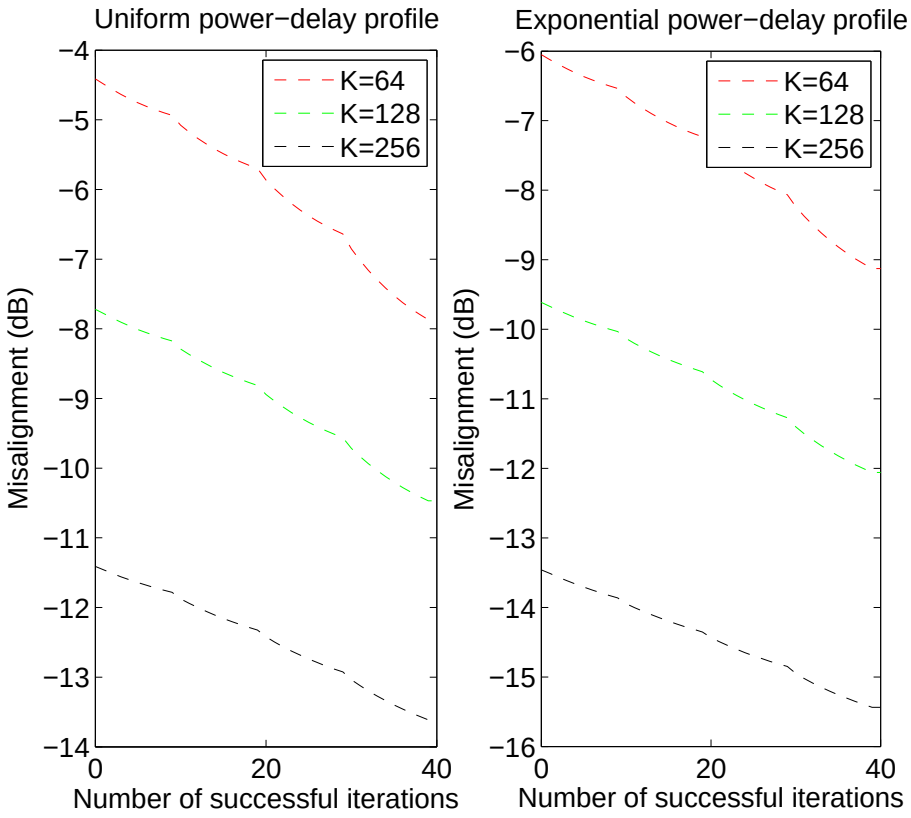


Figure 8.9: Effect of the number of sub-carriers K on the misalignment: the misalignment vs. the number of successful iterations for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, $N_{u_1} = 400$, $N_{u_2} = 10$.

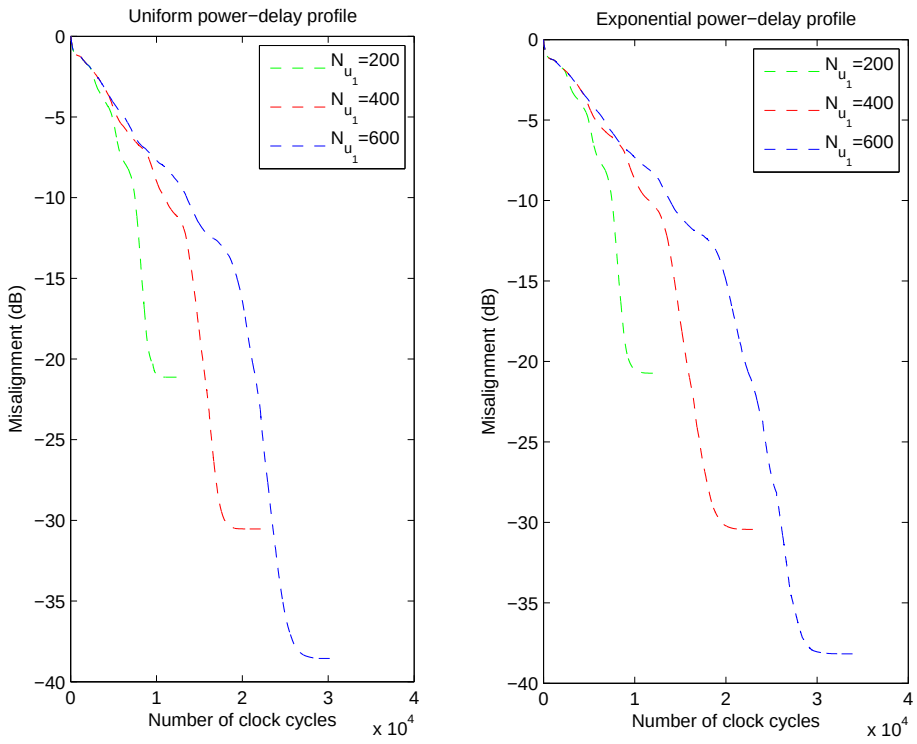


Figure 8.10: Effect of N_{u_1} on the misalignment: the misalignment vs. the number of clock cycles for the 1st sub-carrier; $M_T = 4$, $L = 4$, $K = 128$.

with clock frequency 100 MHz, requires roughly $1.47 \times 10^5/10^8 = 1.47$ ms for such an OFDM symbol inversion. Thus it is not suitable for radio systems with very high data rates, *e.g.* if the duration of an OFDM symbol is 100 us or less.

However, it is suitable in systems with low data rates, *e.g.* the underwater acoustic (UWA) channel is characterized by frequency-selective, (multipath) channel [119]. The use of OFDM signals is now considered as a promising technique for data transmission in the underwater acoustic channel [119–121]. Furthermore, the MIMO techniques have been also applied to UWA communication [122, 123]. The combination of MIMO and OFDM leads to an appealing solution for high data rate transmission. Due to the high complexity of the inversion, the existing inversion approaches can only be implemented for small size systems. Because the typical OFDM symbol duration in UWA is 1 s [124], the proposed approach can be efficiently used in UWA communication systems. Some numerical results are given in the following.

We assume $L = 20$, $K = 1024$ and $T = 1$ s. The required number of FPGA slices (*i.e.* in Xilinx XC2VP30, where 13696 slices are available) and clock cycles for an OFDM symbol inversion in systems of different size at the misalignment of -30 dB are presented

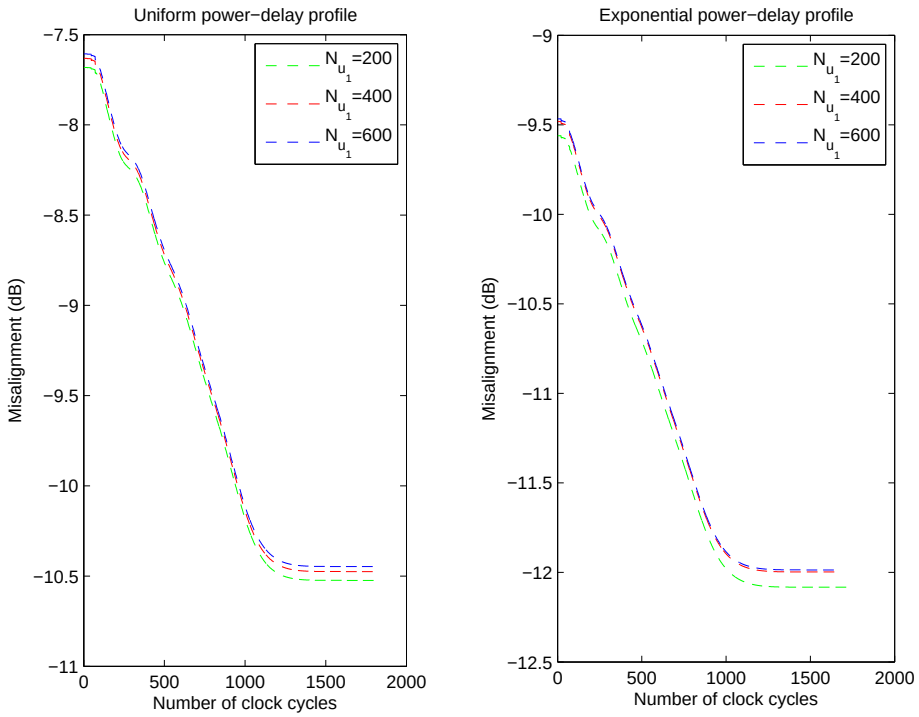


Figure 8.11: Effect of N_{u_1} on the misalignment: the misalignment vs. the number of the clock cycles for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, $K = 128$, $N_{u_2} = 10$.

Table 8.3: Slices and clock cycles required for the DCD-based matrix inversion for MIMO-OFDM systems of size 4×4, 8×8 and 16×16 at the misalignment of -30 dB.

System size	FPGA slices	Clock cycles
$4 \times 4\,(N_{u_1} = 300, N_{u_2} = 20)$	586 (4.2%)	1.75M
$8 \times 8\,(N_{u_1} = 1000, N_{u_2} = 60)$	592 (4.3%)	12.3M
$16 \times 16\,(N_{u_1} = 1200, N_{u_2} = 100)$	610 (4.5%)	72M

in Table 8.3. According to the DCD algorithm FPGA design in [117], the DCD-based matrix inversion requires the number of FPGA slices of 586, 592 or 610 for the system sizes of 4×4, 8×8 or 16×16. Since the DCD algorithm implementation costs only a very small number of slices, the matrix inversion can be performed by using M_T DCD blocks working in parallel, which results in at most $4.5\% \times 16 = 72\%$ slices usage, however, the throughput will be significantly increased.

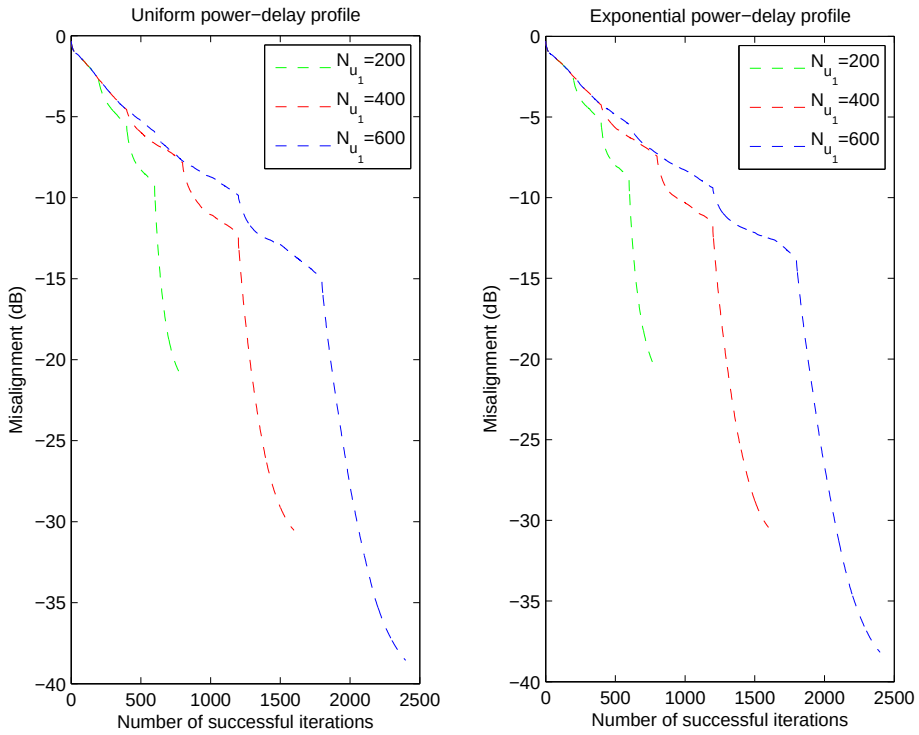


Figure 8.12: Effect of N_{u_1} on the misalignment: the misalignment vs. the number of the successful iterations for the 1st sub-carriers; $M_T = 4$, $L = 4$, $K = 128$.

8.7 Conclusions

In this chapter, we have proposed an approach based on the DCD algorithm to simplify the matrix inversion. This approach obtains separately the individual columns of the inverse of the matrix and costs a very small number of slices, which is suitable for a large size matrix inversion. However, the experiment results show that the throughput of this approach is not high for the standard wireless transmission. Compared to the radio wireless communications, the underwater acoustic communications requires less throughput, which the proposed approach can be applied for. In addition, since this proposed approach requires a very small hardware area, a block of DCD processors can be used in parallel to improve the throughput.

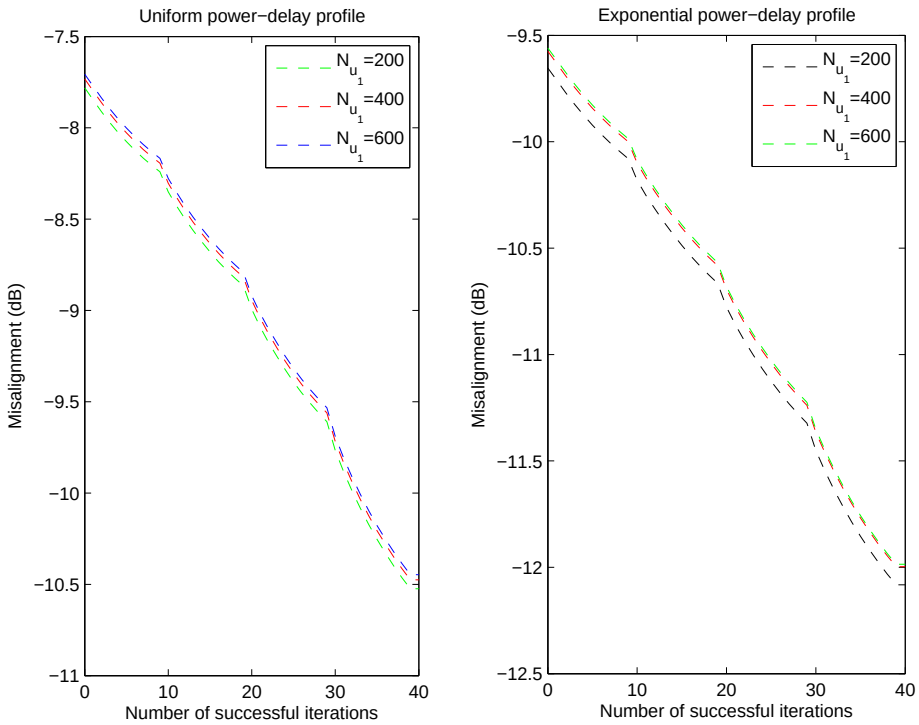


Figure 8.13: Effect of N_{u_1} on the misalignment: the misalignment vs. the number of the successful iterations for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, $K = 128$.

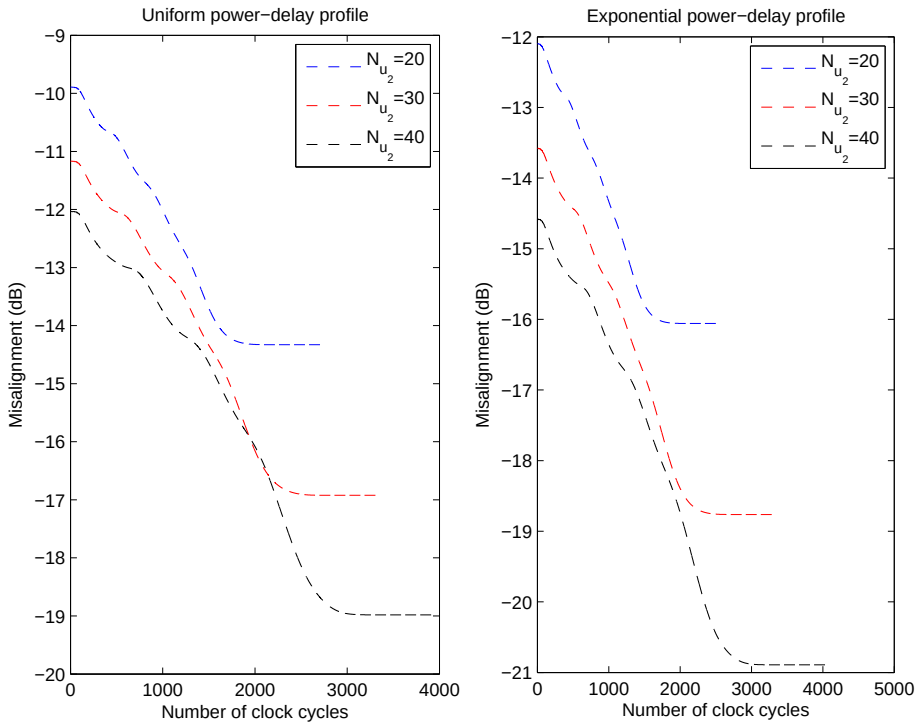


Figure 8.14: Effect of N_{u_2} on the misalignment: the misalignment vs. the number of the clock cycles for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, $K = 128$, $N_{u_1} = 400$.

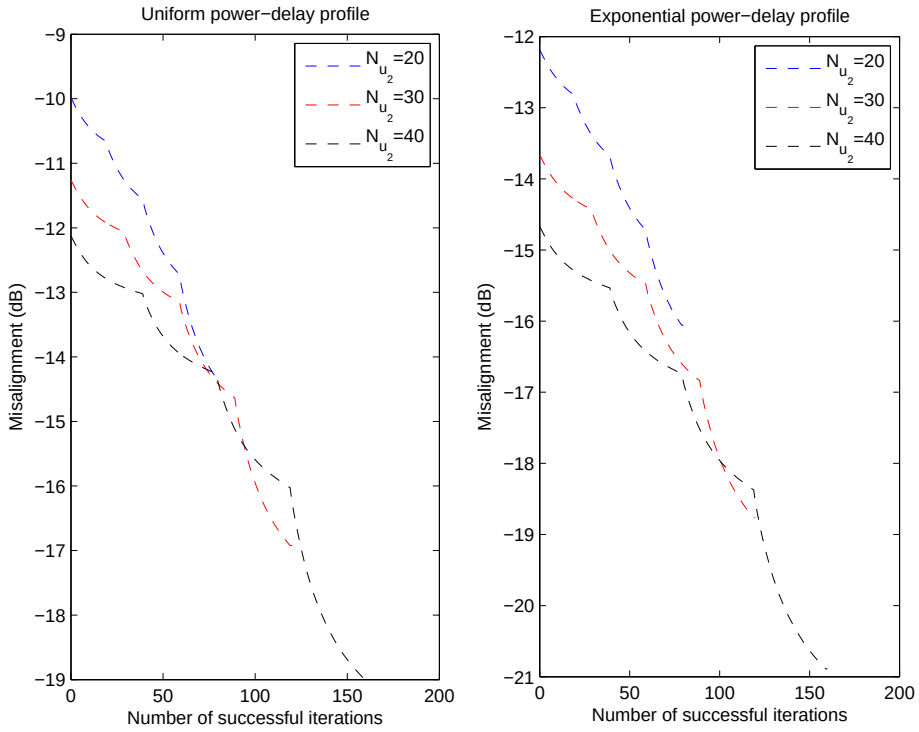


Figure 8.15: Effect of N_{u_2} on the misalignment: the misalignment vs. the number of the successful iterations for every other ($k > 1$) sub-carrier; $M_T = 4$, $L = 4$, $K = 128$, $N_{u_1} = 400$.

Chapter 9

Summary

In this thesis, we have carried out an analysis and optimization of algorithms for multiuser and MIMO detection, and investigated various architecture and implementations of these algorithms on an FPGA platform. The following is a specific summary on the methods and results shown in each chapter:

In Chapter 1, we have identified the challenges to the algorithm design for multiuser and MIMO detection. The signal and channel models have been introduced. In the multiuser detection field, various channels can be converted into a synchronous CDMA channel with modified signature waveforms. The complex channels are transformed to a synchronous CDMA channel, which simplifies the problem analysis. A brief literature survey of the suboptimal multiuser and MIMO receivers have been introduced, which constitute a trade-off compromise in terms of the achievable performance and the associated complexity. These suboptimal detection algorithms however, cannot provide a satisfying trade-off between detection performance and complexity. They are also complicated for hardware implementation.

In Chapter 2, a box-constrained multiuser detector based on the DCD algorithm has been investigated. It is efficient for real-time implementation since it is multiplication and division free. The performance and complexity of this algorithm have been investigated. The results have shown that the hardware area of the box-constrained DCD algorithm is very low even for a high number of users. The performance of the FPGA design of this algorithm has shown to be very close to the floating-point detection performance. Furthermore, two parallelisation architecture designs enable the elements in residual vector to be

updated in one clock cycle, instead of successive clock cycles, and so, could improve the throughput significantly.

In Chapter 3, the box-constrained DCD algorithm has been applied to the MIMO detection. The box-constrained DCD MIMO detector of symbols with QAM modulation has been compared with the MMSE MIMO detector in terms of the design area, throughput, and detection performance. The proposed box-constrained DCD MIMO detector provides a better detection performance than the MMSE detector. Since the DCD MIMO detector has shown relatively low complexity at the the low SNRs region, where the sphere decoding algorithm has exponential complexity, we suggested a combined detector, which at high SNRs region, the sphere decoding algorithm is used for MIMO detection, and the box-constrained DCD algorithm can be used for the detection at low SNRs region.

In Chapter 4, we have proposed an advanced multiuser detection algorithm, the DCD-BTN detector, which is based on box-constrained relaxation, iterated regularization with negative diagonal loading, and DCD iterations. The DCD-BTN algorithm has been compared to a variety of advanced detectors in terms of detection performance and complexity. The DCD-BTN detector has shown the lowest complexity among these detectors and provided the optimal ML performance. The DCD-BTN detector has been implemented on an FPGA board. The fixed-point FPGA DCD-BTN detector has shown a very close detection performance to that of the floating-point implementation. In addition, compared to the DCD detector the DCD-BTN multiuser detector has made a significantly improve on the detection performance with a very low increase in the number of slices.

In Chapter 5, we have proposed a multiuser detector for M-PSK symbols. This proposed detector DCD-BTN-M, is based on the DCD-BTN with some refinements. For large number of users, the proposed detector has been shown to provide a good detection performance close to the single user bound.

In Chapter 6, we have proposed a multiple phase decoder (MPD) for joint detection of M-PSK symbols. The decoder is based on a phase descent search. In the multiuser detection, the MPD has shown a very close detection performance to the single-user bound even in highly loaded scenarios. In comparison to the semi-definite relaxation detector, the MPD has a better detection performance, with a lower complexity. In the MIMO systems, for the QPSK signals, the MPD achieves a very similar performance as that of the sphere decoder. The complexity analysis has shown that the worst-case complexity of the MPD did not vary with SNR, significantly lower than that of the sphere decoder, especially at

low SNRs.

In Chapter 7, we have proposed to combine the box-constrained DCD detector with the fast branch and bound detector (fast BB). The fast BB detector can provide the optimal detection performance, however, the prohibitive worst-case computational complexity makes it difficult for real-time applications. The box-constrained DCD detector has a low computational complexity. But its detection performance is inferior to the fast BB detector. Simulation results have shown that the combined BB-DCD detector provides a better detection performance than the box-constrained DCD detector. In addition, the BB-DCD detector provides a significant reduction in complexity in comparison to the fast BB detector with only a small loss in the detection performance with respect to the ML detector.

In Chapter 8, we have proposed an efficient method based on DCD algorithm which simplifies the matrix inversion operation in MIMO-OFDM systems. The DCD-based inverse for the OFDM system consumes a lower number of slices in the FPGA implementation than other existing advanced matrix inversion techniques. The results show that even for the high system size *e.g.* 8×8 and 16×16, the DCD-based inverse is also applicable with small FPGA slices. In addition, using the implementation of the parallel DCD algorithm could significantly improve the throughput, while the complexity is still acceptable for the FPGA implementation.

9.1 Future Work

In this section, we present several ideas to extend the proposed detectors in our future research.

The box-constrained DCD detector in Chapter 2, the DCD-BTN detector in Chapter 4 have been investigated in AWGN channel only. Other channels, such as Rayleigh or multipath, can also be considered to explore detection performance of these detectors.

In Chapter 6, the hardware architecture of the MPD has been introduced. Most critical steps in PDS algorithm has been clearly given the idea how to implement. Due to the limit time, the proposed detector has not be practically implemented. The detector could be implemented in FPGA board according to the description of the FPGA architecture

design. In addition, the MPD is especially designed for M-PSK symbols detection. We haven't provided the simulation results to show if the MPD provides the good detection performance for QAM symbols detection. The MPD can be optimized by changing the constrained structure for QAM symbols detection.

In Chapter 7, we briefly introduced another combination of the multiuser detection. After the matched filter, the first feasible solution of the fast BB detector can be obtained by the box-constrained DCD algorithm instead of using the decorrelating decision feedback detector. We expect this will reduce the complexity and improve the precision of the initial solution. Moreover, the proposed combination of multiuser detection has not been verified by hardware implementation. The FPGA architecture design has been given in the Chapter 2 and the hardware design information of the branch and bound detector are available from reference.

Bibliography

[1] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO Systems", *IEEE International Conference on Communications, 2006, ICC'06*, vol. 7, pp. 3082–3087, June. 2006.

[2] G. Linlin and S. Puthusserypady, "Performance analysis of Volterra-based nonlinear blind multiuser detector for DS-CDMA systems", *Signal Processing*, vol. 85, no. 10, pp. 1941–1956, July 2004.

[3] S. Verdu, *Optimum multiuser signal detection*, University of Illinois at Urbana-Champaign, Aug. 1984.

[4] Z. Xie, R. T. Short, and C. K. Rushforth, "A family of suboptimum detectors for coherent multiuser communications", *IEEE Journal on Selected Areas In Communications.*, vol. 8, no. 5, pp. 683–690, May 1990.

[5] A. Nordio and G. Taricco, "Linear receivers for the multiple-input and multiple-output multiple access channel", *IEEE Trans, Commun.*, vol. 54, pp. 1446–1456, Aug. 2006.

[6] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels", *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[7] M. O. Damen, H. EI. Gaml, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point", *IEEE Trans. Inf. Theory*, vol. 49, no. 170, pp. 2389–2402, Oct. 2003.

[8] Y. V. Zakharov. and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem", *Eelctronics Letters*, vol. 40, no. 9, pp. 567–569, Apr. 2004.

[9] S. Verdu, *Multiuser Detection*, Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

[10] S. Verdu, "Minimum probability of error for asynchronous Gaussian multiple-access channels", *IEEE Trans. Inform. Theory*, vol. 32, pp. 85–96, Jan. 1986.

[11] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Constrained maximum-likelihood detection in CDMA", *IEEE Trans. Commun.*, vol. 49, no. 1, pp. 142–153, Jan. 2001.

[12] W. K. Ma, T. N. Ching, and Z. Ding, "Semidefinite relaxation based multiuser detection for M-ary PSK multiuser systems", *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2862–2872, Oct. 2004.

[13] X. M. Wang, W. S. Lu, and A. Antoniou, "Multiuser detectors for synchronous DS-CDMA systems based on a recursive p-norm convex relaxation approach", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 5, pp. 1021–1031, May 2005.

[14] J. Luo, K. R. Pattipati, P. K. Willett, and F. Hasegawa, "Near-optimal multiuser detection in synchronous CDMA using probabilistic data association", *IEEE Comm. Letters*, vol. 5, no. 9, pp. 361–363, Sept. 2001.

[15] F. Hasegawa, J. Luo, K. Pattipati, P. Willett, and D. Pham, "Speed and accuracy comparison of techniques for multiuser detection in synchronous CDMA", *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 540–545, Apr. 2004.

[16] S. Moshavi, "Multi-user detection for DS-CDMA communications", *IEEE Communications Magazine.*, vol. 34, no. 10, pp. 124–136, Oct 1996.

[17] R. Gharsallah and R. Bouallegue, "Combined ML-MMSE receiver of an STBC-CDMA system for PSK/QAM modulation", *International Journal of Computer Science and Network Security*, vol. 7, no. 1, pp. 254–258, Jan. 2007.

[18] D. U. Campos-Delgado, F. J. Martinez-Lopez, and J. M. Luna-Rivera, "Analysis and performance evaluation of linear multiuser detectors in the downlink of DS-CDMA systems applying spectral decomposition", *Circuits Systems Signal, Processing*, vol. 26, no. 5, pp. 689–713, 2007.

[19] A. Duel-Hallen, "A family of multiuser decision-feedback detectors for asynchronous CDMA channels", *IEEE Trans. Commun.*, vol. 43, pp. 421–434, Feb.-Apr. 1995.

[20] M. K. Varanasi and T. Guess, "Optimum decision feedback multiuser equalization with successive decoding achieves the total capacity of the Gaussian multiple-access channel", *in Proc. 31st Asilomar Conf. Signals, Systems and Computers, Monterey*, vol. 2, pp. 1405–1409, Nov. 1997.

[21] M. K. Varanasi, "Decision feedback multiuser detection: a systematic approach", *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 219–240, Jan. 1999.

[22] G. Woodward, R. Ratasuk, M. L. Honig, and P. Rapajic, "Multistage multiuser decision-feedback detection for DS-CDMA", *in Proc. IEEE ICC*, vol. 1, pp. 68–72, June 1999.

[23] G. Woodward, R. Ratasuk, M. L. Honig, and P. Rapajic, "Minimum mean-squared error multiuser decision-feedback detectors for DS-CDMA", *IEEE Trans. Commun.*, vol. 50, no. 12, pp. 2104–2112, Dec 2002.

[24] M. Honig, G. Woodward, and Y. Sun, "Adaptive iterative multiuser decision feedback detection", *IEEE Trans. Wireless Commun*, vol. 3, no. 2, pp. 477–485, Mar. 2004.

[25] J. Luo, K. R. Pattipati, P. K. Willet, and F. Hasegawa, "Optimal user ordering and time labeling for ideal decision feedback detection in asynchronous CDMA", *IEEE Trans. Commun*, vol. 51, no. 11, pp. 1754–1757, Nov. 2003.

[26] R. C. de Lamare and R. Sampaio-Neto, "Minimum mean-squared error iterative successive parallel arbitrated decision feedback detectors for DS-CDMA systems", *IEEE Transactions on Communications*, vol. 56, no. 5, pp. 778–789, May 2008.

[27] W. K. Ma, T. N. Davidson, K. M. Wong, Z. Q. Luo, and p. C. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA", *IEEE Trans. Signal Processing*, vol. 50, no. 4, pp. 912–922, Apr. 2002.

[28] J. Jalden and B. Ottersten, "The diversity order of the semidefinite relaxation detector", *IEEE Transaction on Information Theory*, vol. 54, no. 4, pp. 1046–1422, April 2008.

[29] Ma. Wing-Kin, T. N. Davidson, K. M. Wong, Z. Q. Luo, and P. C. Ching, "Efficient quasi-maximum-likelihood multiuser detection by semi-definite relaxation", *Communications, 2001. ICC 2001. IEEE International Conference on.*, pp. 11–14, 2001.

[30] P. M. Pardalos and M. G. C. Resende, "Interior point methods for global optimization", *in Interior Point Method of Mathematical Programming, T. Terlaky, Ed. Norwell, MA: Kluwer*, vol. 5, pp. 467–500, 1996.

[31] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Sphere-constrained maximum-likelihood detection in CDMA", *in Proc. 2000. Int. Zurich Sem. Broadband Communications, Zurich, Switzerland*, pp. 55–62, Feb. 2000.

[32] B. P. Tan, L. K. Rasmussen, and T. J. Lim, "Constrained maximum-likelihood detection in CDMA", *IEEE Trans.Commun*, vol. 49, no. 1, pp. 142–153, Jan. 2001.

[33] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Box-constrained maximum-likelihood detection in CDMA", *in Proc. IEEE 51st Vehicular Technology Conf. Spring-2000,Tokyo, Japan*, vol. 1, pp. 517–521, May 2000.

[34] J. Luo, "Improved multiuser detection in code-division multiple access communications", *PhD Thesis in University of Connecticut*, 2002.

[35] P. H. Tan, L. K. Rasmussen, and J. Luo, "Iterative multiuser decoding based on probabilistic data association", *in Proc. IEEE International Commun. Lett.*, pp. 301–301, June 2003.

[36] Y. V. Zakharov and F. Albu, "Coordinate Descent Iterations in Fast Affine Projection Algorithm", *IEEE Signal Processing Lett.*, vol. 12, pp. 353C356, 2005.

[37] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes", *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 227–232, Mar. 1982.

[38] C. H. Papadimitriou and K. Stieglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentic Hall, Englewood Cliffs, NJ, 1982.

[39] J. Luo, K. Pattipati, P. Willett, and L. Brunel, "Branch-and-bound-based fast optimal algorithm for multiuser detection in synchronous CDMA", *in Proceedings of the IEEE International Conference on Communications*, vol. 5, pp. 3336–3340, May 2003.

[40] D. Bertsekas, *Network Optimization, Continous and Discrete Models*, Athena Scientific, Belmont, MA., 1998.

[41] J. Luo, K. P. Pattipati, P. Willet, and G. M. Levchuk, "Fast optimal and suboptimal any-time algorithms for CDMA multiuser detection based on branch and bound", *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 632–641, 2004.

[42] S. Fouladi and C. Tellambura, "Smart maximum-likelihood CDMA multiuser detection", *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*, pp. 522–525, 2005.

[43] H. Z. Wang, P. Leray, and J. Palicot, "Reconfigurable architecture for MIMO systems based on CORDIC operators", *Elsevier Comptes rendus, Physique*, vol. 7, no. 7, pp. 735–750, Sept. 2006.

[44] D. Gesbert, M. Shafi, D. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems", *IEEE Journal On Selected Areas in Communications*, vol. 21, no. 3, pp. 281–301, Apr. 2003.

[45] C. Windpassinger, L. Lampe, R. F. H. Fischer, and T. Hehn, "A performance study of MIMO detectors", *Wireless Communications, IEEE Transactions on*, vol. 5, pp. 2004–2008, 2006.

[46] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding", *IEEE J. Solid-State Circuits.*, vol. 39, no. 9, pp. 1544–1552, Sep. 2004.

[47] L. Azzam and E. Ayanoqlu, "Reduction of ML decoding complexity for MIMO Sphere Decoding, QOSTBC, and OSTBC", *Information Theory and Applicaitons Workshop, 2008*, pp. 18–25, Feb. 2008.

[48] A. Burg, M. Borgmanr, M. Wenk, C. Studer, and H. Bolcskei, "Advanced receiver algorithms for MIMO wireless communications", *Proceedings of Design, Automation and Test in Eurpoe, 2006.*, vol. 1, pp. 592–598, 2006.

[49] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications", *SIGSAM Bull.*, vol. 15, pp. 37–44, 1981.

[50] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis", *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, 1985.

[51] Andrea Goldsmth, *Wireless Communication*, Cambridge University Press, 2005.

[52] L. J. Cimini, "Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing", *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 665–675, Jul. 1985.

[53] Y. Li, L. J. Cimini, and N. R. Sollenberger, "Robust channel estimation for OFDM systems with rapid dispersive fading channels", *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 902–915, July 1998.

[54] S. S. Riaz Ahamed, "Performance analysis of OFDM", *Journal of Theoretical and Applied Information Technology*, pp. 23–30, 2008.

[55] K. Wong, C. Tsiu, R. K. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO systems", *in Proc. IEEE International Symposium on Circuits and Systems (ISCAS'02), Scottsdale, AZ*, vol. 3, pp. 273–276, May 2002.

[56] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Eurchner decoder for MIMO systems", *in Proc. IEEE 6th Circuit and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, Shanghai, China*, vol. 1, pp. 65–68, June 2004.

[57] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI Implementation of MIMO detection using the sphere decoding algorithm", *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[58] G. Foschini and M. Gans, "On limits of wireless communications in a fading environment when using multiple antenas", *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, Mar. 1998.

[59] I. Telatar, "Capacity of multi-antenna Gaussian channels", *European Transactions on Telecommunicatoins*, vol. 10, no. 6, pp. 585–595, 1999.

[60] A. J. Paulraj and C. B. Papadias, "Space-time processing for wireless communications", *IEEE Signal Processing Magazine*, vol. 14, no. 6, pp. 49–83, 1997.

[61] A. J. Paulraj and D. A. Gore, "An overview of MIMO communications-a key to gigabit wireless", *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–217, 2004.

[62] X. Huang, C. Liang, and J. Ma, "System architecture and implementation of MIMO sphere decoders on FPGA", *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 16, no. 2, pp. 188–197, Feb. 2008.

[63] A. DeHon, "The density advantage of configurable computing", *IEEE Comput.*, vol. 33, no. 4, pp. 41–49, Apr. 2000.

[64] A. Adjoudani, E. C. Beck, A. P. Burg, G. M. Djuknic, T. G. Gvoth, D. Haessing, S. Manji, M. A. Milbrodt, M. Rupp, D. Samardzjia, A. B. Siegel, T. II. Sizer, C. Tran, S. Walker, S. A. Wilkus, and P. W. Wolniansky, "Prototype experience for MIMO Blast over third-generation wirless system", *IEEE J. Sel. Areas Commun.*, vol. 21, no. 3, pp. 440–451, Mar. 2003.

[65] M. Guillaud, A. Burg, M. Rupp, E. Beck, and S. Das, "Rapid prototyping design of a 4x4 Blast-over-UMTS system", *in Proc. 35th Asilomar Conf. Pacific Grove, CA.*, vol. 2, pp. 1256–1260, Nov. 2001.

[66] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems", *in Proc. IEEE ICC'06, Istanbul, Turkey*, vol. 7, pp. 3082–3087, June 2006.

[67] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis", *Mathematics of Computation*, vol. 44, pp. 463–471, April 1985.

[68] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel", *IEEE Trans.Commun*, vol. 51, pp. 389–399, Mar. 2003.

[69] J. B Anderson and S. Mohan, "Sequential coding algorithms: a survey and cost analysis", *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.

[70] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm", *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[71] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO signal detector design and VLSI implementation", *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 15, no. 3, pp. 328–337, March 2007.

[72] Q. T. Ho and D. Massicotte, "FPGA implementation of adaptive multiuser detector for DS-CDMA systems", *in Proceedings of 14th International Conference on Field Programmable Logic and Application (FPL'04)*, , no. 1, pp. 959–964, Aug.-Sep. 2004.

[73] Adel-Omar Dahmane Quoc-Thai Ho, Daniel Massicotte, "FPGA implementation of an MUD based on cascade filters for a WCDMA system", *EURASIP Journal on Applied Signal Processing*, , no. 1, pp. 1–12, 2006.

[74] T. Cesear and R. Uribe, "Exploration of least-squares solutions of linear systems of equations with fixed-point arithmetic hardware", *in Proceeding of the SDR 05 Technical Conference and Product Exposition*, Nov. 2005.

[75] Y. V. Zakharov and T. C. Tozer, "Box-constrained multiuser detection based on multiplication-free coordinate descent optimisation", *in Proc. Fifth IEEE Workshop on Signal Processing Advances in Wireless Communications, Lisboa, Portugal*, pp. 11–14, Jul. 2004.

[76] J. Jalden and B. Ottersten, "An exponential lower bound on the expected complexity of sphere decoding", *in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. iv–393– iv–396, May 2004.

[77] "Virtex-4 Libraries Guide for HDL Designs", .

[78] J. Liu, B. Weaver, and G. White, "FPGA implementation of the DCD algorithm", *in London Communication Sysposium, London, U.K.*, pp. 125–128, Sept. 2006.

[79] J. Liu, Z. Quan, and Y. V. Zakharov, "Parallel FPGA implementation of DCD algorithm", *15th Interntional Conference on Digital Signal Processing, Cardiff, U.K.*, pp. 331–334, July. 2007.

[80] T. Koike, Y. Seki, H. Murata, S. Yoshida, and K. Araki, "Prototype implementation of real-time ML detectors for spatial multiplexing transmission", *IEICE Transactions on Communications*, pp. 845–852, 2006.

[81] X. Huang, C. Liang, and J. Ma, "System architecture and implementation of MIMO sphere decoders on FPGA", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 188–197, 2008.

[82] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems", *IEEE Int. Conf. Communications, ICC'2006*, vol. 7, pp. 3082–3087, 2006.

[83] H. S. Kim, W. Zhu, K. Mohammed, A. Shah, and B. Daneshrad, "An efficient FPGA based MIMO-MMSE detector", *EUSIPCO'2007, Poznan, Poland*, pp. 1131–1135, 2007.

[84] H. S. Kim, W. Zhu, J. Bhatia, K. Mohammed, A. Shah, and B. Daneshrad, "A practical,hardware friendly MMSE detector for MIMO-OFDM-based systems", *EURASIP Journal and Advances in Signal Processing*, pp. ID 267460,14 pages, 2008.

[85] J. Eilert, D. Wu, and D. Liu, "Implementation of a programmable linear MMSE detector for MIMO-OFDM", *IEEE Int. Conf. Acoustis, Speech, and Signal Processing, ICASSP-2008*, pp. 5396–5399, 2008.

[86] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Box-constrained maximum-likelihood detection in CDMA", in *Proc. IEEE 51st Vehicular Technology Conf. Spring-2000, Tokyo, Japan*, May 2000, vol. 1, pp. 517–521.

[87] A. Bateman and I. Paterson-Stephens, *The DSP handbook: algorithms, applications and design techniques*, Prentice Hall, 2002.

[88] "Virtex-5 FPGA User Guides".

[89] Y. V. Zakharov, J. Luo, and C. Kasparis, "Joint box-constraint and deregularization in multiusr detection", *14th European Signal Processing Conference, Florence, Italy*, Sep. 2006.

[90] Engl. HeinzW, "Necessary and sufficient conditions for convergence of regularization methods for solving linear operator equations of the first kind", *Nuerical Functional Analysis and Optimization*, vol. 3, no. 2, pp. 201–222, 1981.

[91] S. A. Vorobyov, A. B. Gershman, and Z. Q. Luo, "Robust adaptive beamforming using worst-case performance optimization: a solution to the mismatch problem", *IEEE Trans. Signal Processing*, vol. 51, no. 2, pp. 313–324, Feb. 2003.

[92] A. Neumaier, "Solving ill-conditioned and singular linear systems: a turtorial on regularization", *SIAM Review*, vol. 40, no. 3, pp. 636–666, 1998.

[93] M. Hanke and C. W. Groetsch, "Nonstationary iterated Tikhonov regularization", *J.Optim. Theory Appl.*, vol. 98, pp. 37–53, Jul. 1998.

[94] C. Kasparis, R. J. Piechocki, P. N. Fletcher, and A. R. Nix, "A bootstrap multiuser detector for CDMA based on Tikhonov regularization", *in Proc. IEEE ICASSP, Hong Kong*, vol. 4, pp. 69–72, Apr. 2003.

[95] A. Yener, R. D. Yates, and S. Ulukus, "CDMA multiuser detection:a nonlinear programming approach", *IEEE Trans. Commun*, vol. 50, no. 6, pp. 1016–1024, Jun. 2002.

[96] Y. Zakharov, J. Luo, and C. Kasparis, "Joint box-constraint and deregularization in multiuser detection", *in Proceedings of EUSIPCO*, 2006.

[97] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices", *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[98] J. Jalden and B. Otterson, "On the complexity of sphere decoding in digital communications", *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[99] P. H. Tan and L. K. Rasmussen, "Multiuser detection in CDMA - A comparison of relaxations, exact, and heuristic search methods", *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, pp. 1802–1809, Sep. 2004.

[100] M. O. Damen, A. Chkeif, and J. C. Belfiore, "Lattice codes decoder for space-time codes", *IEEE Communications Letters*, vol. 4, no. 5, pp. 161–163, May 2000.

[101] P. W. Wolniansky and G. J. Foschini, "V-BLAST: an architecture for realizing very high data rate over the rich-scatting wireless channel", *Proceedings of ISSSE-98, Pisa, Italy*, pp. 295–300, Sept. 1998.

[102] L. G. Barbero and J. S. Thompson, "Performance analysis of a fixed-complexity sphere decoder in high-dimensional MIMO systems", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 557–560, May 2006.

[103] W. Xu, Y. Wang, Z. Zhou, and J. Wang, "A computationally efficient exact ML sphere decoder", *IEEE Global Telecommunications Conference, USA*, vol. 4, pp. 2594–2598, Dec. 2004.

[104] F. P. Vasiliev, *Numerical methods for solution of extremum problems (in Russian)*, Nauka, Moscow, 1988.

[105] W.-K. Ma, P.-C. Ching, and Z. Ding, "Semidefinite relaxation based multiuser detection for M-ary PSK multiuser systems", *IEEE Trans. Signal Processing*, vol. 52, no. 10, pp. 2862–2872, Oct. 2004.

[106] T. Cui and C. Tellambura, "Generalized feedback detection for MIMO systems", *IEEE Global Telecommunications Conference, St. Louis, MO, USA*, vol. 5, pp. 3077–3081, Dec. 2005.

[107] K. Lee and J. Chun, "ML symbol detection based on the shortest path algorithm for MIMO systems", *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5477–5484, Nov. 2007.

[108] B. Steingrimsson, T. Luo, and K. M. Wong, "Soft quasi-maximum likelihood detection for multiple-antenna wireless channels", *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2710–2719, Nov. 2003.

[109] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm: Part I, the expected complexiy", *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[110] S. Verdu, "Computational complexity of optimum multiuser detection", *Algorithmica*, vol. 4, no. 1, 2005.

[111] S. Verdu and H. Poor, "Abstract dynamic programming models under communtativity conditions", *SIAM J. Control Optim.*, vol. 25, no. 4, pp. 990–1006, 1987.

[112] Y. Takano, H. Sekiya, J. M. Lu, and T. Yahagi, "Multiuser detection with branch-and-bound-based algorithm using user grouping for synchronous CDMA", *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005*, vol. 1, pp. 442–446, Sept. 2005.

[113] R. Lupas and S. Verdu, "Linear multiuser detectors for synchronous code-division multiple-access channels", *IEEE Trans. Inform. Theory*, vol. 35, no. 1, pp. 123–136, 1898.

[114] S. Verdu, *Multiuser Detection*, Cambridge University Press, 1998.

[115] P. F. Driessen and G. J. Foschini, "On the capacity formula for multiple input multiple output wireless channels: a geometric interpretation", *IEEE Transactions on Communications*, vol. 47, no. 2, pp. 173–176, Feb. 1999.

[116] I. Barhumi, G. Leus, and M. Moonen, "Optimal training design for MIMO OFDM systems in mobile wireless channels", *Signal Processing, IEEE Transactions on*, vol. 51, no. 6, pp. 1615–1624, June 2003.

[117] J. Liu, Y. Zakharov, and B. Weaver, "Architecture and FPGA implemenation of dichotomous coordinate descent algorithm", *IEEE Transactions on Circuits and Systems Part I: Regular Paper, to appear.*

[118] M. Karooti, J. R. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm", *in Proc. on 2005 Asilomar conference, Pacific Grove, USA*, Oct 2005.

[119] E. Bejjani and J. C. Belfiore, "Multicarrier coherent communications for the underwater acoustic channel", *in Processings MTS/IEEE OCEANS'96, Fort Lauderdale, FL, USA*, vol. 3, pp. 1125–1130, Sep. 1996.

[120] W. K. Lam and R. F. Ormondroyd, "A coheret COFDM modulation system for a time-varying frequency-selective underwater acoustic channel", *Proc. of the 7th International Conference on Electroni Engineering in Oceanography*, pp. 198–203, Jun. 1997.

[121] Y. V. Zakharov and V. P. Kodanev, "Multipath-Doppler diversity of OFDM signals in an underwater acoustic channel", *in Procc. IEEE ICASSP'2000*, vol. 5, pp. 2941–2944, June 2000.

[122] D. B. Kilfoyle, J. C. Preisig, and A. B. Baggeroer, "Spatial modulation experiments in the underwater acoustic channel", *IEEE Journal of Oceanic Engineering*, vol. 30, pp. 406–415, Apr. 2005.

[123] S. Roy, T. M. Duman, V. McDonald, and J. G. Proakis, "High rate communication for underwater acoustic channels using multiple transmitters and space-time coding: Receiver structures and experimental results", *IEEE Journal of Oceanic Engineering*, vol. 32, no. 3, pp. 663–688, Feb. 2007.

[124] Xu. Zhengyi, Y. V. Zakharov, and V. P. Kodanev, "Space-time signal processing of OFDM signals in fast-varying underwater acoustic channel", *Oceans 2007-Europe*, pp. 1–6, June 2007.