UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Design and Implementation of Efficient Algorithms for Wireless MIMO Communication Systems

DOCTORAL THESIS

by
Sandra Roger Varea

Supervisors:
Prof. Alberto Gonzalez Salvador
Dr. Vicenç Almenar Terré

Valencia, Spain
July 2012

To Max

"No queda más remedio que jugar la partida con
las cartas que el burlón Destino te pone en las manos,
aunque éstas sean pésimas"

"There is nothing to do but to play the game
with the hand dealt by a mocking Destiny,
no matter how terrible"

El Capitán Alatriste,
Arturo Pérez Reverte

Abstract

In the last decade, one of the most significant technological developments that led to the new broadband wireless generation is the communication via multiple-input multiple-output (MIMO) systems. MIMO technologies have been adopted by many wireless standards such as Long Term Evolution (LTE), Wordlwide interoperability for Microwave Access (WiMAX) and Wireless Local Area Network (WLAN). This is mainly due to their ability to increase the maximum transmission rates, together with the achieved reliability and coverage of current wireless communications, all without the need for additional bandwidth nor transmit power. Nevertheless, the advantages provided by MIMO systems come at the expense of a substantial increase in the cost to deploy multiple antennas and also in the receiver complexity, which has a major impact on the power consumption. Therefore, the design of low-complexity receivers is an important issue which is tackled throughout this thesis.

First, the use of MIMO channel matrix preprocessing techniques to either decrease the computational cost of optimal sphere decoders or to improve the performance of suboptimal linear, successive interference cancellation (SIC) or tree-search detectors is investigated. A detailed overview of two widely employed preprocessing techniques, the Lenstra, Lenstra, Lovasz (LLL) lattice-reduction (LR) algorithm and the vertical Bell-Labs layered space-time zero forcing decision feedback equalization (VBLAST ZF-DFE) ordering, is presented. Both the complexity and performance of these methods are evaluated and compared. In addition, a low-complexity implementation of the VBLAST ZF-DFE is proposed and included in the evaluation.

Second, a low-complexity tree-search MIMO detector, called the variable-breadth (VB) K-Best detector, is developed. The main idea of this method is to exploit the impact of the channel matrix condition number in data detection in order to decrease the complexity of already proposed detection schemes. In the VB K-Best method, the value of its K parameter is varied depending on the channel matrix condition number. The proposed approach includes a low-complexity condition number estimator stage and a threshold selection method. The results show that the proposed scheme has lower average complexity than a fixed K-Best detector of similar perfor-

mance. In addition, a second detection scheme is proposed, which employs the idea of condition number thresholding to avoid carrying out a lattice-reduction stage when the channel has already good condition number. This way, a high number of LR calls is avoided while keeping good detection performance.

In the third part of this thesis, several contributions which involve the use of LR for MIMO communications are presented. First, the combination of LR with the K-Best algorithm is investigated and alternative implementations that outperform previous proposals are developed. An extended LLL algorithm for LR is proposed to assist the preprocessing part of some lattice-reduction-aided (LRA) K-Best schemes. Finally, this extended LLL algorithm is exploited to decrease the computational cost of several LRA precoding methods. In addition, the most employed signal precoding approaches are evaluated and compared in terms of both computational cost and performance.

Next, the problem of efficient soft detection in MIMO bit-interleaved-coded-modulation (MIMO-BICM) systems is addressed. An efficient fixed-complexity demodulator for systems working with quantized reliability information is proposed. This approach reduces the complexity of previously proposed schemes through the combination of two strategies: a novel tree-pruning based on quantization and a clipping-based pruning. Results after quantization reveal that a significant complexity reduction is achieved with negligible performance degradation.

The last part of the thesis is devoted to the use of Graphic Processing Units (GPU) for the efficient implementation of MIMO receivers. Both a hard-output and a soft-output version of the fixed-complexity sphere decoder are implemented in GPU. Results show that the proposed implementations decrease the computational time required for the data detection stage in MIMO systems considerably, with respect to conventional CPU implementation. Moreover, a fully-parallel soft-output scheme with a GPU-aware preprocessing stage is developed. Again the execution time of the proposed GPU implementation is compared with its execution time on a high performance CPU, showing that the GPU outperforms the CPU. Furthermore, the throughputs of all the algorithms are shown to be higher than those of other recent implementations while ensuring nearly-optimal detection performance.

Keywords: MIMO, Sphere decoding, Tree-search detection, lattice reduction, log-likelihood ratio quantization, GPU.

Resumen

En la última década, uno de los avances tecnológicos más importantes que han hecho culminar la nueva generación de banda ancha inalámbrica es la comunicación mediante sistemas de múltiples entradas y múltiples salidas (MIMO). Las tecnologías MIMO han sido adoptadas por muchos estándares inalámbricos tales como Long Term Evolution (LTE), Wordlwide interoperability for Microwave Access (WiMAX) y Wireless Local Area Network (WLAN). Esto se debe principalmente a su capacidad de aumentar la máxima velocidad de transmisión, junto con la fiabilidad alcanzada y la cobertura de las comunicaciones inalámbricas actuales sin la necesidad de ancho de banda extra ni de potencia de transmisión adicional. Sin embargo, las ventajas proporcionadas por los sistemas MIMO se producen a expensas de un aumento sustancial del coste de implementación de múltiples antenas y de la complejidad del receptor, la cual tiene un gran impacto sobre el consumo de energía. Por esta razón, el diseño de receptores de baja complejidad es un tema importante que se abordará a lo largo de esta tesis.

En primer lugar, se investiga el uso de técnicas de preprocesado de la matriz de canal MIMO bien para disminuir el coste computacional de decodificadores óptimos o bien para mejorar las prestaciones de detectores subóptimos lineales, de cancelación sucesiva de interferencias (SIC) o de búsqueda en árbol. Se presenta una descripción detallada de dos técnicas de preprocesado ampliamente utilizadas: el método de Lenstra, Lenstra, Lovasz (LLL) para *lattice reduction* (LR) y el algoritmo VBLAST ZF-DFE. Tanto la complejidad como las prestaciones de ambos métodos se han evaluado y comparado entre sí. Además, se propone una implementación de bajo coste del algoritmo VBLAST ZF-DFE, la cual se incluye en la evaluación.

En segundo lugar, se ha desarrollado un detector MIMO basado en búsqueda en árbol de baja complejidad, denominado detector K-Best de amplitud variable (VB K-Best). La idea principal de este método es aprovechar el impacto del número de condición de la matriz de canal sobre la detección de datos con el fin de disminuir la complejidad de los sistemas de detección anteriormente propuestos. El valor del parámetro K se varía dependiendo del número de condición del canal. El esquema propuesto incluye una etapa de estimación del número de condición de baja complejidad y un método de selección del umbral. Los resultados demuestran que el

sistema propuesto tiene menor complejidad media que un detector K-Best fijo de prestaciones similares. Además, se propone un segundo esquema de detección que utiliza umbralización basada en número de condición para evitar llevar a cabo una etapa de LR cuando el canal ya tiene un número de condición suficientemente bueno. De esta manera, se evita un elevado número de llamadas al algoritmo LR manteniendo buenas prestaciones.

En la tercera parte de esta tesis se presentan varias contribuciones relacionadas con el uso de LR en sistemas de comunicaciones MIMO. En primer lugar, se investiga la combinación de LR con el algoritmo K-Best y se proponen implementaciones alternativas para mejorar a las desarrolladas anteriormente. Se propone un algoritmo LLL extendido para apoyar a la parte de preprocesado de algunos esquemas K-Best con LR. Por último, este algoritmo LLL extendido se ha explotado para reducir el coste computacional de varios métodos de precodificación MIMO basados en LR. Además, los algoritmos de precodificación de señal más utilizados se han evaluado y comparado en términos de coste computacional y prestaciones. A continuación se aborda el problema de la detección soft eficiente en sistemas MIMO con modulación codificada de bits entrelazados (MIMO-BICM). Se propone un demodulador eficiente de complejidad fija para sistemas que trabajen con información de fiabilidad cuantificada. Este esquema reduce la complejidad de los algoritmos propuestos previamente a través de la combinación de dos estrategias: un algoritmo de poda basado en cuantificación y una poda basada en clipping. Con ello se consigue una importante reducción de la complejidad con una degradación marginal en las prestaciones.

La última parte de la tesis se centra en el uso de unidades gráficas de proceso (GPU) para la implementación eficiente de receptores para sistemas MIMO. Se han implementado en GPU algoritmos de complejidad fija tanto con salidas hard como soft. Los resultados muestran que las implementaciones propuestas disminuyen considerablemente el tiempo de cómputo requerido por la etapa de detección de datos en sistemas MIMO con respecto a la implementación convencional en CPU. Por otro lado, se ha desarrollado un algoritmo soft totalmente paralelo con una etapa de preprocesado adecuada para implementación en GPU. De nuevo se compara el tiempo de ejecución de la implementación en GPU con el tiempo de ejecución en una CPU de alto rendimiento, mostrando que la GPU supera a la CPU. Además, los *throughputs* de todos los algoritmos demuestran ser superiores a los de otras implementaciones recientes al tiempo que garantizan unas prestaciones de detección casi óptimas.

Palabras Clave: MIMO, Sphere decoding, detección por búsqueda en árbol, Lattice reduction, cuantificación de LLR, GPU.

Resum

En l'última dècada, un dels avanços tecnològics més importants que han fet culminar la nova generació de banda ampla inalàmbrica és la comunicació mitjançant sistemes de múltiples entrades i múltiples eixides (MIMO). Les tecnologies MIMO han estat adoptades per molts estàndards sense fils com ara Long Term Evolution (LTE), Wordlwide interoperability for Microwave Access (WiMAX) i Wireless Local Area Network (WLAN). Això es deu principalment a la seua capacitat d'augmentar la màxima velocitat de transmissió, juntament amb la fiabilitat assolida i la cobertura de les comunicacions sense fils actuals, tot açò sense la necessitat d'ample de banda extra ni de potència de transmisió addicional. No obstant això, els avantatges proporcionats pels sistemes MIMO es produeixen a costa d'un important augment dels costos d'implementació de múltiples antenes així com de la complexitat del receptor, la qual té un gran impacte sobre el consum energètic. Per aquesta raò, el disseny de receptors de baixa complexitat és un tema important que s'abordarà al llarg d'aquesta tesi.

En primer lloc, s'investiga l'ús de tècniques de preprocessat de la matriu de canal MIMO bé per disminuir el cost computacional de descodificadors òptims o bé per millorar les prestacions de detectors subòptims lineals, de cancellació successiva d'interferències (SIC) o de recerca en arbre. Es presenta una descripció detallada de dues tècniques de preprocessat àmpliament utilitzades: el mètode de Lenstra, Lenstra, Lovasz (LLL) per a *lattice reduction* (LR) i l'algorisme VBLAST ZF-DFE. Tant la complexitat com les prestacions d'ambdós mètodes s'han avaluat i comparat. Amés, es proposa una implementació de baix cost de l'algorisme VBLAST ZF-DFE, la qual s'inclou en la avaluació.

En segon lloc, s'ha desenvolupat un detector MIMO basat en recerca en arbre de baixa complexitat, denominat detector K-Best d'amplitud variable (VB K-Best). La idea principal d'aquest mètode és aprofitar l'impacte del nombre de condició de la matriu de canal sobre la detecció de dades per tal de disminuir la complexitat dels sistemes de detecció anteriorment proposats. El valor del paràmetre K es varia depenent del nombre de condició del canal. L'esquema proposat inclou una etapa de estimació del nombre de condició de baixa complexitat i un mètode de selecció del llindar. Els resultats demostren que el sistema proposat té menor complexitat mitjana

que un detector K-Best fix de prestacions similars. Amés, es proposa un segon esquema de detecció que utilitza el llindar en el nombre de condició per evitar dur a terme una etapa de LR quan el canal ja té un nombre de condició prou bo. D'aquesta manera, s'evita un elevat nombre d'execucions de l'algorisme LR mantenint bones prestacions.

A la tercera part d'aquesta tesi es presenten diverses contribucions relacionades amb l'ús de LR en sistemes de comunicacions MIMO. En primer lloc, s'investiga la combinació de LR amb l'algorisme K-Best i es proposen implementacions alternatives per millorar les desenvolupades anteriorment. Es proposa un algorisme LLL estès per donar suport a la part de preprocessat d'alguns esquemes K-Best amb LR. Per últim, aquest algorisme LLL estès s'ha explotat per reduir el cost computacional de diversos mètodes de precodificació MIMO basats en LR. Amés, els algorismes de precodificació de senyal més utilitzats s'han avaluat i comparat en termes de cost computacional i prestacions. A continuació s'aborda el problema de la detecció soft eficient en sistemes MIMO amb modulació codificada de bits entrellaçats (MIMO-BICM). Es proposa un demodulador eficient de complexitat fixa per a sistemes que treballen amb informació de fiabilitat quantificada. Aquest esquema redueix la complexitat dels algorismes proposats prèviament mitjançant la combinació de dues estratègies: un algorisme de poda basat en quantificació i una poda basada en clipping. Els resultats després de quantificació revelen que s'aconsegueix una reducció important de la complexitat amb una degradació marginal de les prestacions.

L'última part de la tesi està destinada a l'ús d'unitats gràfiques de procés (GPU) per a la implementació eficient de receptors per a sistemes MIMO. S'han implementat en GPU algorismes de complexitat fixa tant amb eixides hard com soft. Els resultats mostren que les implementacions propostes disminueixen considerablement el temps de còmput requerit per l'etapa de detecció de dades en sistemes MIMO respecte a la implementació convencional en CPU. D'altra banda, s'ha desenvolupat un algorisme soft totalment parallel amb una etapa de preprocessat adequada per a implementació en GPU. De nou es compara el temps d'execució de la implementació en GPU amb el temps d'execució en una CPU d'alt rendiment, mostrant que la GPU supera la CPU. Amés, els *throughputs* de tots els algorismes demostren ser superiors als d'altres implementacions recents, alhora que garanteixen unes prestacions de detecció quasi òptimes.

Paraules Clau: MIMO, Sphere Decoding, detecció per recerca en arbre, Lattice reduction, quantificació de LLR, GPU.

Acknowledgements

It is a pleasure for me to thank those who made this thesis possible. First and foremost, I offer my sincerest gratitude to my supervisors, Prof. Alberto Gonzalez and Dr. Vicenc Almenar, who supported me throughout this thesis with their knowledge and advice whilst allowing me the room to work in my own way.

I am very grateful to Prof. Mark Flanagan from the University College Dublin for serving as a reviewer of this thesis. Special thanks also to Dr. Alexandre Graell i Amat from the Chalmers University of Technology and to Dr. Miguel Gonzalez Lopez from the Universidade da Coruña for also serving as reviewers of this thesis and for acting as members of the committee. All of them provided me with very useful comments that definitely helped to improve the final manuscript. I thank Prof. Javier Rodriguez Fonollosa for also being a member of the committee.

I would like to thank Prof. Gerald Matz, who hosted me at the Institute of Telecommunications of the Vienna University of Technology during some months. I really appreciate the opportunity he gave me to work within his team. A special mention goes to Dr. Clemens Novak for all his technical advice and to Arrate Alonso for making me spend a wonderful time in Vienna.

I owe my deepest gratitude to Prof. Antonio M. Vidal for his generous advice regarding many computational complexity aspects and for sharing with me his wide mathematical knowledge every time that I needed it.

I would like to show my gratitude to all the people at the Universitat Politècnica de València that shared my daily work at the Institute of Telecommunications and Multimedia Applications (iTEAM). In particular, I would like to thank Prof. Jose J. Lopez, Dr. Gema Piñero, Dr. María de Diego, Dr. Miguel Ferrer, Dr. Paco Martinez and Dr. Victor M. Garcia. Thanks also to my current and former colleagues at the iTEAM: Emanuel Aguilera, Amparo Martí, Fernando Domene, Luis Maciá, Jorge Lorente,

Carla Ramiro, Laura Fuster, Ana Torres, Pedro Zuccarello, Sergio Tejero and Eliseo Villanueva for all the good moments spent together while working at the lab. Also, thanks to my Master's classmates Maria Cabanes and Nuria Ortigosa.

Special thanks go to Jose A. Belloch for more than a decade of friendship and of continuous collaboration. I cannot forget to thank also my colleague and close friend David Gozalvez for all his daily support at the iTEAM and for his help regarding my professional future.

I wish also to thank the Spanish Ministry of Science and Innovation for the received financial support under the FPU program.

Thanks to my dear friends Inma, Vero, Patri, Pili, Soraya, Marta, Pamela, Lucía and Laura for being always by my side and for making me smile even in my worse days.

I would like to acknowledge the help and encouragement given by my parents, Jose and Mica, and my sister Miriam. A big hug to my grandmas Mica and Gloria and to the rest of my family.

Last, but not least, I would like to express my sincere gratitude to Max, who offered me unconditional love and support throughout the course of this thesis. Without you, going through this would have been even harder. Thank you for showing me always the bright side of life. I love you.

Sandra Roger
July 2012

Contents

List of Figures

List of symbols

$\mathbf{X}$	Matrix
$\mathbf{x}$	Vector
x	Scalar
$X_{i,j}$	i, j component of matrix $\mathbf{X}$
$\mathbf{X}_{i,:}$	i-th row of matrix $\mathbf{X}$
$\mathbf{X}_{:,i}$	i-th column of matrix $\mathbf{X}$
$\mathbf{X}_{i:j,k:l}$	Elements from i-th to j-th row and from k-th to l-th column of $\mathbf{X}$
$Tr(\mathbf{X})$	Trace of matrix $\mathbf{X}$
x_i	i-th component of vector $\mathbf{x}$
$(\cdot)^T$	Transpose
$(\cdot)^*$	Complex conjugation
$(\cdot)^H$	Conjugate transpose
$(\cdot)^+$	Moore-Penrose pseudoinverse
$(\cdot)^{-1}$	Matrix inversion
$\mathbf{I}_{N\times N}$	Identity matrix of size $N \times N$
$\lvert\cdot\rvert$	Absolute value
$\lVert\cdot\rVert_p$	ℓ_p norm
$\lVert\cdot\rVert_F$	Frobenius norm
$P(A)$	Marginal probability of A
$P(A\lvert B)$	Conditional probability of A, given B
$\Re\{.\}$	Real part of a complex number
$\Im\{.\}$	Imaginary part of a complex number
max	Maximum of a set
min	Minimum of a set
arg max	Argument of the maximum of a set
arg min	Argument of the minimum of a set
log	Natural logarithm
$\mathcal{Q}\{\cdot\}$	Quantization (slicing) operation
$\mathcal{O}(\cdot)$	Denotes an asymptotic complexity order of $\cdot$

Abbreviations and Acronyms

a.k.a.	also known as
ASD	Automatic Sphere Decoder
AWGN	Additive White Gaussian Noise
BER	Bit error rate
BICM	Bit-interleaved coded-modulation
BLAST	Bell Labs Layered Space-Time
BPSK	Binary phase shift keying
BS	Base station
CSI	Channel State Information
CUDA	Compute unified device architecture
DFE	Decision feedback equalizer
DSP	Digital signal processor
e.g.	for example (from the latin *exempli gratia*)
fcLLL	Fixed-complexity Lenstra, Lenstra, Lovasz algorithm
FE	Full expansion
FPGA	Field Programmable Gate Array
FP-SD	Fincke-Pohst Sphere Decoder
FPFSD	Fully-parallel Fixed-complexity Sphere Decoder
FSD	Fixed-complexity Sphere Decoder
GPU	Graphic processing unit
LDPC	Low-Density Parity-Check
i.e.	that is (from the latin *id est*)
i.i.d.	Independently, identically distributed
LLL	Lenstra, Lenstra, Lovasz algorithm
LLR	Log-likelihood ratio
LR	Lattice reduction
LRA	Lattice-reduction-aided
LSD	List Sphere Decoder
LTE	Long Term Evolution
MAP	Maximum-a-posteriori
MF	Matched filter
MIMO	Multiple-input multiple-output
MISO	Multiple-input single-output

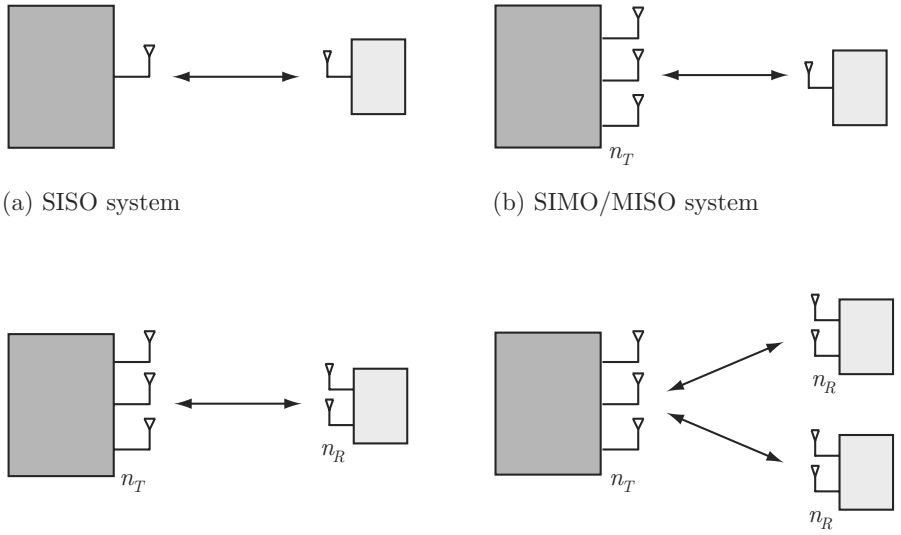
ML	Maximum-likelihood
MMSE	Minimum mean squared error
MU	Multiuser
OFDM	Orthogonal frequency division multiplexing
OSIC	Ordered successive interference cancellation
pdf	Probability density function
PED	Partial Euclidean distance
PM	Power method
QAM	Quadrature amplitude modulation
QPSK	Quadrature phase shift keying
RTS	Repeated tree search
SD	Sphere decoder
SDMA	Space-Division Multiple Access
SE	Single expansion
SE-SD	Schnorr-Euchner Sphere Decoder
SFSD	Soft-output Fixed-complexity Sphere Decoder
SIC	Successive interference cancellation
SISO	Single-input single-output
SIMO	Single-input multiple-output
SINR	Signal to interference plus noise ratio
SM	Stream Multiprocessor
SNR	Signal to noise ratio
STS	Single tree search
THP	Tomlinson-Harashima Precoding
VB	Variable breadth
V-BLAST	Vertical Bell Labs Layered Space-Time
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
ZF	Zero forcing

1 Introduction

Introduction 1

1.1 Introduction

In the last decade, one of the most significant technological developments that led to the new broadband wireless generation is the communication via multiple-input multiple-output (MIMO) systems [1]. The popularity of MIMO technologies is not limited to the research world but it has also impacted wireless communication industry dramatically, proof of that is the adoption of MIMO techniques by many existing wireless standards such as Long Term Evolution (LTE), Wordlwide interoperability for Microwave Access (WiMAX) and Wireless Local Area Network (WLAN). The recent attention attracted by MIMO systems is mainly due to their ability to increase the maximum transmission rates, together with the achieved reliability and coverage of current wireless communications without the need for additional bandwidth nor transmit power. This is indeed a huge advantage, since spectrum resources are very scarce and expensive. Moreover, keeping the transmit power as low as possible is a crucial factor in the battery life of wireless communication devices. Due to all these reasons, MIMO is a current field of active international research.



(a) SISO system

(b) SIMO/MISO system

(c) MIMO system

(d) MU-MIMO system

Figure 1.1. MIMO systems evolution.

MIMO systems appeared as the evolution of single-input single-output (SISO) systems (see Fig. 1.1(a)), i.e. those using only a single transmitting and a single receiving antenna. SISO systems capacity is only limited by the signal-to-noise ratio (SNR) of the system. The first step towards MIMO was to employ several antennas at either the transmitter or at the receiver side such as the single-input multiple-output (SIMO) and multiple-input single-output (MISO) systems shown in Fig. 1.1(b). Smart antennas, which use arrays of antennas, are examples of the latter and improve performance in terms of coverage, capacity or quality of the radio link. To this end, by using beamforming and/or spatial diversity techniques, a logarithmic increase in spectral efficiency can be achieved. Finally, point-to-point or single-user MIMO (SU-MIMO) systems, i.e. those employing multiple antennas at the transmitter and receiver side simultaneously, appeared with the basic aim of exploiting the spatial dimension as well as the temporal dimension [2]. A general example of a MIMO system is shown in Fig. 1.1(c).

The vision for current generation cellular networks includes data rates approaching 100 Mbps for highly mobile users and up to 1 Gbps for low mobile or stationary users [3], which calls for high spectral efficiency. There-

fore, despite the use of MIMO systems was traditionally intended for point-to-point communication, multiuser MIMO (MU-MIMO) technology is expected to play a key role in this context. MU-MIMO systems are those where there is a communication between a base station (BS) and multiple users, all of them using more than one antenna, as shown by Fig. 1.1(d). There are two different challenges in MU-MIMO scenarios: uplink (where multiple users transmit simultaneously to a single BS) and downlink (where the BS transmits to multiple independent users). The uplink challenge is addressed using array processing and multiuser detection techniques by the base station in order to separate the signals transmitted by the users. The downlink challenge is somewhat different, it is similar to that of the point-to-point MIMO scenario except for the fact that the receiver antennas are distributed among different independent users. This creates a challenge in detecting the received symbols since joint detection requires each user to have the symbol received from all the receiver antennas of all the users. Almost all the proposed techniques intended to address the MU-MIMO downlink challenge process the data symbols at the transmitter itself, that is, using precoding.

The benefits of MIMO technology that help achieve significant performance improvements are *array gain*, *spatial diversity gain*, *spatial multiplexing gain* and *interference reduction* [2]. These benefits are described below.

- *Array gain* is an increase in the average SNR at the receiver that results from a coherent combining effect of the received wireless signals. Array gain improves the coverage and range of wireless networks by improving their resistance to noise. The coherent combining of signals may be carried out through spatial processing at the receiving antennas and/or spatial pre-processing at the transmitting antennas. Channel knowledge is required at the part of the system where array gain is desired. Channel knowledge at the receiver side is typically available whereas at the transmitter side is more difficult to obtain.

- *Spatial diversity gain* is a useful tool to mitigate the random fluctuations (fades) that signal power suffers in common wireless links. Diversity techniques are based on transmitting the same signal over multiple paths with (ideally) independent fading. Diversity can be carried out in time, frequency or space. Spatial diversity is preferred

over time/frequency diversity as it does not involve an increase in transmission time or bandwidth. With an increasing number of independent copies of the signal, the probability that at least one of the copies is not experiencing a deep fade increases and, thus, the quality and reliability of the reception is improved. A MIMO channel with n_T transmitting antennas and n_R receiving antennas can ideally offer $n_T n_R$ independently fading links. Spatial diversity gain in the absence of channel knowledge at the transmitter is possible using space-time coding techniques, where the transmitted signals are carefully designed.

- *Spatial multiplexing gain* is achieved when multiple, independent data streams are transmitted simultaneously within the bandwidth of operation. This technique can offer a linear increase in data rate provided suitable channel conditions are available, such as rich scattering in the environment. This way, the receiver can separate the different streams, yielding a linear increase in the capacity of the wireless network.

- *Interference reduction and avoidance.* Interference results from multiple users sharing time and frequency resources in a wireless network. When multiple antennas are used, the differentiation between the spatial signatures of the desired signal and cochannel signals can be exploited to reduce interference. For instance, in the presence of interference, array gain increases the tolerance to noise as well as the interference power, hence improving the signal-to-noise-plus-interference ratio (SINR). Interference reduction and avoidance improve the coverage and range of a wireless network.

Generally, it is not possible to exploit all the advantages of MIMO technology simultaneously, due to conflicting demands on the spatial degrees of freedom [2]. Nevertheless, a suitable combination of the above described benefits through the careful design of the signaling scheme and transceiver will result in improved capacity, coverage and reliability.

The performance improvements resulting from the use of MIMO systems are due to three main techniques: *precoding*, *spatial multiplexing* and *space-time coding* [1]. In the following, these techniques are briefly introduced.

- *Precoding* for point-to-point MIMO systems is a generalization of beamforming to support multi-layer transmission in MIMO wireless communications. In (single-layer) beamforming, the same signal is emitted from each of the transmit antennas with appropriate phase (and sometimes gain) weighting such that the signal power is maximized at the receiver ouput. The benefits of beamforming are to increase the signal gain from constructive combining and to reduce the multipath fading effect. When the receiver has multiple antennas, the transmit beamforming cannot simultaneously maximize the signal level at all receive antennas and then precoding is used [4]. In MU-MIMO, the data streams are intended for different users (known as space-time division multiple access (SDMA)) and the performance cannot be simultaneously maximized for all the users. Thus, some measure of the total throughput (e.g., the sum performance) is maximized.

- *Spatial multiplexing* splits a high rate signal into multiple lower rate streams and each stream is transmitted from a different transmit antenna in the same frequency channel. If these signals arrive at the receiver antenna array with sufficiently different spatial signatures, the receiver can separate these streams, creating parallel channels for free. Spatial multiplexing is a very powerful technique to increase channel capacity at high SNR. The maximum number of spatial streams is limited by the lesser in the number of antennas at the transmitter or receiver [4][5]. Spatial multiplexing can be used with or without channel state information (CSI) at the transmitter.

- *Space-time coding techniques* (also known as diversity coding techniques) are used to achieve spatial diversity gain. A single stream (unlike multiple streams in spatial multiplexing) is transmitted but, in this case, the signal is coded using certain principles of full or near orthogonal coding [6][7][8]. Diversity coding exploits the independent fading in the multiple antenna links to enhance signal diversity. As previously said, these techniques are used without CSI at the transmitter, thus, they do not provide any array gain.

Spatial multiplexing can also be combined with precoding when the channel is known at the transmitter or combined with diversity coding when decoding reliability is in trade-off [9].

MIMO techniques can be also used to enhance the performance of orthogonal frequency division multiplexing (OFDM) systems by exploiting the spatial domain. OFDM is an easy technique to mitigate the effects of inter-symbol interference in frequency selective channels, turning a broadband frequency selective channel into a set of narrowband channels by transmitting data in parallel over the different subcarriers. OFDM combined with MIMO systems, also known as MIMO-OFDM [10], allows transmitting different streams over the different subcarriers and, through MIMO precoding, different spatial beams in each one of the subcarriers. The advantages of MIMO-OFDM are the increased data throughput and link reliability, as well as improved interference suppression. Here, all techniques to exploit the MIMO gains can be applied on a subcarrier-by-subcarrier basis.

The wide range of advantages provided by MIMO systems comes at the expense of a substantial increase in the receiver complexity (and sometimes in the transmitter complexity as well). In fact, when spatial multiplexing is considered, while the capacity increases linearly with the minimum of the number of antennas at the transmitter and the receiver, the detection complexity undergoes a more than linear increase. Therefore, if wireless communication devices with a high level of integrability and affordable cost are aimed, the design of low-complexity receivers is mandatory [2].

1.2 Motivation and Scope

It is known that the use of MIMO communication systems complicates the receiver stage and, specially, the detection part. The detector has the task of processing the received mixture of signals affected by the channel in order to recover the transmitted data with the highest reliability. Hence, the search for low-complexity MIMO detectors is imperative. Achieving this general goal is the main motivation of this dissertation. This section describes some particular motivations leading towards low-complexity detectors design.

If nearly optimal data detection is desired, the detector becomes often the most computationally expensive algorithm within a MIMO receiver. Nevertheless, the computational cost of data detection can be decreased by following different strategies such as the combination of suboptimal MIMO

detection with channel matrix preprocessing techniques. However, care must be taken to avoid an unexpected increase of the computational cost of the joint detection scheme. To this end, it seems useful to compare some of the most employed preprocessing approaches in terms of performance and complexity before choosing the best option. In addition, trying to reduce the computational cost of the receivers preprocessing stage is also a challenging task.

On the other hand, sphere decoding (SD) has been shown to substantially decrease the computational complexity of exhaustive *maximum-likelihood* (ML) MIMO hard-output detection [11]. Unfortunately, the worst-case complexity of SD methods still remains exponential with the number of transmitting antennas [12]. Therefore, the search for low and predictable complexity tree-search methods is still an intense research topic [13]. Among many others, the K-Best tree-search algorithm is a useful approach to trade detection performance and complexity, proof of that are the high number of published variants of this method [14][15]. Thus, further improving the K-Best algorithm is another motivation behind this thesis.

Lattice-reduction (LR) techniques, which are a particular case of channel matrix preprocessing, have been widely employed for different applications, such as data detection in MIMO systems or signal precoding in MU-MIMO systems [16][17][18]. However, these strategies not only transform the MIMO channel matrix but also include a non-linear transformation in the transmitted constellation symbols. This non-linear transformation often complicates the use of LR techniques with tree-search detection methods. Thus, it is necessary to search for strategies that allow both techniques being used jointly.

In MIMO systems with bit-interleaved-coded-modulation (MIMO-BICM), the demodulation (or soft detection) and channel decoding are not usually performed jointly at the receiver, but in two differentiated stages. First, the demodulator provides reliability information (soft outputs) about the transmitted coded bits in the form of real-valued log-likelihood ratios (LLR). Next, these values are used by the channel decoder to make final decisions on the transmitted coded bits. The complexity to perform optimum soft detection is even higher than the one of hard detection, thus, the search for efficient soft detectors will be another topic to be tackled throughout this thesis.

Moving from the design of MIMO receivers towards practical physical

setups, the search for high-throughput MIMO receiver implementations is a major importance issue. Furthermore, scalability in the number of subcarriers per MIMO-OFDM symbol as well as in the system size are key factors in LTE and 4G wireless standards [3]. Therefore, focusing on the requirements of high-throughput and scalability, the use of many-core processors such as general purpose Graphic Processing Units (GPU) is becoming increasingly attractive for the efficient implementation of signal processing algorithms for communication systems [19][20]. The use of this hardware can accelerate the computation by employing many cores to execute in parallel as many parts of the algorithms as possible. Hence, exploiting the potential of GPU to implement MIMO detection algorithms seems a very promising goal. Finally, it is advisable to compare the throughput achieved by the developed GPU implementations with the minimum required by current wireless standards in order to find out which transmission configurations could be supported.

Taking into account the above presented motivations, the main scope of this thesis is the following:

- *To evaluate the computational cost and performance of existing MIMO channel matrix preprocessing algorithms.*
- *To decrease the computational cost of existing MIMO channel matrix preprocessing algorithms.*
- *To contribute with new tree-search-based hard-output MIMO detection algorithms with lower computational cost than previously proposed approaches.*
- *To develop new lattice-reduction-aided schemes with lower computational cost than existing ones.*
- *To contribute with new tree-search-based soft-output MIMO detection algorithms with lower computational cost than previously proposed approaches.*
- *To develop high-throughput implementations of hard- and soft-output schemes using hardware architectures with high parallel processing capabilities and to evaluate them in terms of speedup and throughput.*

1.3 Key Contributions

Although this thesis tackles different particular problems arising in MIMO communication systems, note that all of them are aimed at achieving a common goal: designing computationally-efficient MIMO receiver algorithms. In this section, the main contributions of this thesis are summarized.

MIMO Channel Matrix Preprocessing Techniques

The use of channel matrix preprocessing techniques before data detection in MIMO systems has been shown to be promising for improving diverse aspects of MIMO detectors. On the one hand, some preprocessing techniques can help to decrease the computational cost of optimal performance detection methods. On the other hand, the use of preprocessing can improve the performance in terms of bit-error-rate (BER) of suboptimal detectors.

In this study, a detailed overview of two widely employed preprocessing techniques (the Lenstra, Lenstra, Lovasz lattice-reduction (LR) algorithm [21] and the vertical Bell-Labs layered space-time zero-forcing decision feedback equalizer (VBLAST ZF-DFE) ordering [22]) is presented. First, the computational costs of both methods are evaluated and compared. After that, the performance improvement achieved by both methods when used previously to K-Best tree-search detection is assessed. The purpose is to show the SNR ranges at which each detector performs better than the other one and vice versa. Moreover, a low-complexity implementation of the VBLAST ZF-DFE preprocessing is proposed, which is based on the QR decomposition of the matrices involved in the channel matrix reordering process.

Hard-Output Tree-Search MIMO Detectors

Previous works have shown that the performance of MIMO detectors is highly influenced by the MIMO channel matrix condition number [23]. In this thesis, the impact of the 2-norm channel matrix condition number in data detection is exploited in order to decrease the computational cost of previously proposed MIMO detectors. First, a variable-breadth (VB) K-Best detector where the value of its K parameter is varied depending on the channel matrix condition number is developed. The method makes use of a low-complexity condition number estimator stage and a threshold

selection method, which are also contributions of this work. The results show that the proposed scheme has lower average complexity than a fixed K-Best detector of similar performance and, thus, it allows a meaningful power-saving.

Another novel contribution is a detection scheme that exploits the knowledge of the channel matrix condition number to avoid carrying out a high percentage of LR calls. This way, complexity is saved while keeping good detection performance.

Lattice-Reduction-Aided Schemes

In this study, several new approaches involving the use of LR are included. First, the combination of LR with the K-Best algorithm is investigated and alternative implementations that outperform previous proposals are developed. Furthermore, an extended LLL algorithm for LR is proposed to assist the preprocessing part of some lattice-reduction-aided (LRA) K-Best schemes. Moreover, the proposed extended LLL algorithm is shown to decrease the computational cost of several LRA precoding methods for MU-MIMO systems. For the sake of completeness, the computational cost and performance of the most employed signal precoding approaches are evaluated and compared.

Soft-Output Tree-Search MIMO Detectors

Channel codes used in current wireless communications (low-density parity-check codes (LDPC), turbo codes, etc.) work with bit sequences of thousands to ten thousands of bits. Hence, the use of LLR quantizers previous to LLR storage and further processing is becoming very usual in practical MIMO-BICM systems.

In this thesis we focused on the use of LLR quantization to reduce the complexity of the soft detection stage of MIMO-BICM receivers. An efficient soft detection scheme based on the soft-output fixed-complexity sphere decoder (SFSD) is proposed. The proposed approach uses the fact that the quantized LLRs belong to an *a priori* known discrete set of values (quantization levels) to avoid some of the computations carried out by the soft detector. In addition, LLR clipping is also included to save complexity. The performance and computational cost of the method are both evaluated and compared to other existing approaches.

Implementation of High-Throughput MIMO Detection Schemes

As claimed throughout this thesis, the use of MIMO communication systems complicates the receiver stage, which has the task of processing the received mixture of signals affected by the channel in order to recover the transmitted data with the highest reliability. Therefore, having high-throughput receiver implementations is nowadays an important challenge.

As previously said, the use of many-core processors such as GPU has recently become attractive for the efficient implementation of signal processing algorithms for communication systems. In this work, several GPU-based implementations of sphere decoding algorithms providing either hard or soft outputs are presented. The proposed implementations are evaluated and compared in terms of achieved throughput as well as measuring the computational time to execute the algorithms in GPU and on a high-performance CPU.

1.4 Organization of the Thesis

The remainder of this thesis describes the research that has been undertaken to develop the aims stated above. The chapters are organized and presented as follows:

- Chapter 2: This chapter contains an introduction to the topic of MIMO communications and describes many concepts necessary for the understanding of this dissertation. In the first sections, the data detection task in MIMO-Bell Labs layered space-time (BLAST) systems together with many widely employed detection algorithms is described. Special attention is paid to those detectors based on a tree-search strategy. Next, the transformation of an uncoded MIMO system into a MIMO-BICM scheme is presented and, subsequently, the use of MIMO demodulators is introduced. Finally, some preliminaries related to signal precoding in multiuser MIMO systems are also discussed.

- Chapter 3: This chapter presents a detailed overview of two widely employed preprocessing techniques aimed at improving MIMO detection performance: the LLL LR algorithm and the VBLAST ZF-DFE

ordering. Both the complexity and performance of the two methods are evaluated and compared. In addition, a low-complexity implementation of the VBLAST ZF-DFE is proposed and included in the evaluation.

- Chapter 4: In this chapter, the impact of the 2-norm channel matrix condition number in data detection is exploited to decrease the complexity of two already proposed detection schemes. First, a variable-breadth K-Best detector is developed, where the value of its K parameter is varied depending on the channel matrix condition number. The proposed approach includes a low-complexity condition number estimator stage and a threshold selection method. Furthermore, the idea of condition number thresholding is applied to a LRA detection scheme in order to avoid carrying out the preprocessing stage when the channel has already good condition number.

- Chapter 5: In this chapter, several contributions involving lattice-reduction-aided algorithms are presented. First, the combination of LR preprocessing with the K-Best algorithm is investigated and alternative implementations that outperform previous proposals are developed. An extended LLL algorithm for LR is proposed to assist the preprocessing part of some LRA K-Best schemes. In the last part of the chapter, this extended LLL algorithm is exploited to decrease the computational cost of several LRA precoding methods. In addition, the most employed signal precoding approaches are evaluated and compared in terms of both computational cost and complexity.

- Chapter 6: This chapter is focused on the use of soft detection in MIMO-BICM. In practice, the reliability information delivered by soft detectors (one LLR per coded bit) is usually represented with a finite word-length, thus, LLR quantization is often mandatory. In this chapter, an efficient fixed-complexity demodulator for systems working with quantized LLRs is proposed. This approach reduces the complexity of previously proposed schemes through the combination of two strategies: a novel tree-pruning based on quantization and a clipping-based pruning. The performance of the method is evaluated to further justify the interest of the proposed approach.

- Chapter 7: This chapter presents several fixed-complexity sphere decoders implementations on GPU, which allow to considerably de-

crease the computational time required for the data detection stage in MIMO systems. Both, hard- and soft-output detection methods are implemented. In addition, a novel fully-parallel soft-output scheme which can suitably exploit the GPU capabilities is proposed. The execution times of all the proposed GPU implementations are compared with their execution times on a high performance CPU. Moreover, the throughput of the algorithms is evaluated and compared to other recent implementations and to the requirements specified by current wireless standards.

- Chapter 8: Finally, the conclusions obtained throughout this thesis are presented, including some guidelines for future research lines. A list of published work related to this thesis is also given.

2 Preliminaries and State of the Art

2 Preliminaries and State of the Art

This chapter contains an introduction to the topic of MIMO communications and describes many concepts necessary for the understanding of this dissertation. In the first sections, the data detection task in MIMO-BLAST systems together with many widely employed detection algorithms are described. Special attention is paid to those detectors based on a tree-search strategy. Next, how an uncoded MIMO system is transformed into a MIMO-BICM scheme is presented. Then, the use of MIMO demodulators is introduced. Finally, some preliminaries related to signal precoding in multiuser MIMO systems are also presented.

2.1 Multiple-Input Multiple-Output Systems

2.1.1 The BLAST System

The well-known Bell-Labs layered space-time system (BLAST) is a high speed wireless communication system that employs multiple antennas at both the transmitter and the receiver [22]. Spatial multiplexing gain is achieved by splitting the data bitstream into n_T transmit antennas (see

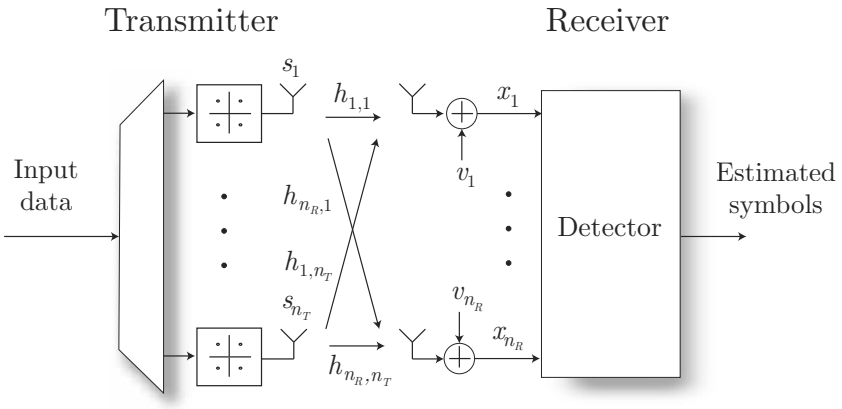


Figure 2.1. Spatial multiplexing MIMO system with n_T transmitting antennas and n_R receiving antennas.

Fig. 2.1). The data is simultaneously sent to the channel, thus overlapping in both time and frequency. The signals are received by n_R receiving antennas, as shown in Fig. 2.1, and the receiver has the task of processing the received signals in order to recover the transmitted data.

Let us consider a MIMO-BLAST system characterized as block fading (the channel remains constant along the whole transmission of a data block), with n_T transmit antennas, n_R receive antennas, $n_R \geq n_T$, and a certain SNR. The baseband equivalent model for such MIMO system is given by

$$\mathbf{x} = \mathbf{H}\mathbf{s} + \mathbf{v}, \tag{2.1}$$

where $\mathbf{s}$ represents the baseband signal vector transmitted during each symbol period, $\mathbf{s} = (s_1, \ldots, s_{n_\mathrm{T}})^T$.

Fig. 2.2 shows the quadrature-amplitud-modulation (QAM) constellations usually employed in MIMO communications, which are known as M-QAM. M stands for the number of constellation points (constellation size) and is usually in the set $M = \{4, 16, 64\}$. We let $P_M = \{-\sqrt{M} + 1, \cdots, -1, 1, \cdots, \sqrt{M} - 1\}$ be the real-valued representation of a QAM constellation and define the constellation as $\Omega = \{a + bj : a, b \in P_M\}$. Then, the symbols s_i are taken from Ω and carry $\log_2 M$ Gray-encoded bits each,

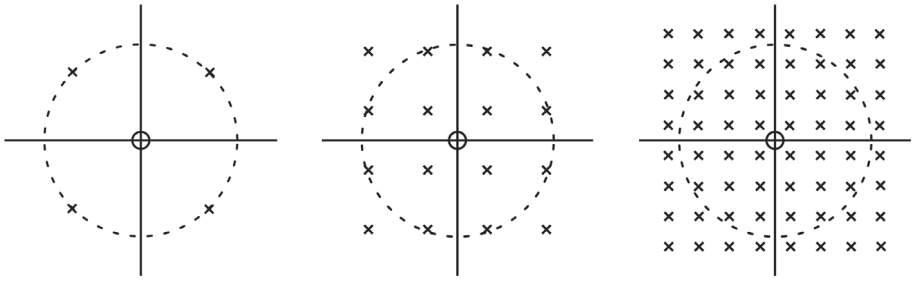


Figure 2.2. Most employed M-QAM constellations.

which in a BLAST system are grouped after being demultiplexed into n_T streams.

Vector $\mathbf{x}$ in (2.1) denotes the received symbol vector and $\mathbf{v}$ is a complex white Gaussian noise vector with zero mean and variance $\sigma^2/2$ per real dimension, being the variance of each complex noise component equal to σ^2. The channel matrix $\mathbf{H}$ is formed by $n_R \times n_T$ complex-valued elements, $h_{i,j}$, which represent the complex fading gain from the jth transmit antenna to the ith receive antenna:

$$\mathbf{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,n_T} \\ h_{2,1} & h_{2,2} & \dots & h_{2,n_T} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n_R,1} & h_{n_R,2} & \dots & h_{n_R,n_T} \end{pmatrix}. \tag{2.2}$$

A Rayleigh fading model without correlation is often considered for the channel matrix, i.e., its entries are chosen independently as zero-mean complex Gaussian random variables with variance 1/2 per real dimension. Also, the channel matrix is commonly assumed to be known at the receiver after a channel estimation stage and to remain constant during a block of L_{ch} symbol vectors.

The SNR of the system is defined as the ratio between the average

transmitted symbol energy and the noise variance [24]

$$\mathrm{SNR} = \frac{E_s}{\sigma^2}, \tag{2.3}$$

where E_s is the average energy of the transmitted signal vector $\mathbf{s}$, which is often normalized before transmission to ensure $E_s = 1$. As said before, the elements of $\mathbf{s}$ belong to one of the sets in Fig. 2.2 in most cases. Therefore, in order to normalize the symbols, the average energy of each constellation (E_M) can be easily calculated as:

$$E_M = \sqrt{\frac{2(M-1)}{3}}. \tag{2.4}$$

In a MIMO system without channel coding, the SNR can be related to the E_b/N_0 (SNR per bit) as follows:

$$\frac{E_b}{N_0} = \frac{n_R \cdot SNR}{n_T \cdot \log_2 M}. \tag{2.5}$$

If the bits are encoded using a rate-R error correcting code, the expression of the E_b/N_0 is:

$$\frac{E_b}{N_0} = \frac{n_R \cdot SNR}{n_T \cdot R \cdot \log_2 M}. \tag{2.6}$$

As digital signal processors (DSP) do not often support complex-valued operations [25], in practice it is sometimes useful to transform the ($n_R \times n_T$)-dimensional complex equation (2.1) into an equivalent ($2n_R \times 2n_T$)-dimensional real-valued representation (2.7), as in [14]

$$\begin{bmatrix} \Re(\mathbf{x}) \\ \Im(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \Re(\mathbf{H}) & -\Im(\mathbf{H}) \\ \Im(\mathbf{H}) & \Re(\mathbf{H}) \end{bmatrix} \begin{bmatrix} \Re(\mathbf{s}) \\ \Im(\mathbf{s}) \end{bmatrix} + \begin{bmatrix} \Re(\mathbf{v}) \\ \Im(\mathbf{v}) \end{bmatrix}. \tag{2.7}$$

Note that the new size of $\mathbf{x}$ and $\mathbf{v}$ is $2n_R \times 1$, vector $\mathbf{s}$ turns into a $2n_T \times 1$ vector and thus the channel matrix $\mathbf{H}$ has $2n_R \times 2n_T$ entries. In the parts of the thesis where the the real-valued system (2.7) is employed, this fact will be highlighted in the text.

2.1.2 Capacity of the MIMO Channel

The capacity of a channel determines the theoretical limit of the amount of data bits that can be transmitted through the communication link without errors. The capacity of SISO narrowband memoryless channels was derived in [26] and is represented by

$$C = \log[1 + \mathrm{SNR}|h|^2], \tag{2.8}$$

where h represents the complex gain of the wireless channel. This expression was firstly extended to the MIMO case in [27]. When the channel is constant and known perfectly at the transmitter and the receiver, the capacity of the MIMO system is

$$C = \max_{\mathbf{Q}:Tr(\mathbf{Q})=P} \log\{\det(\mathbf{I}_{n_R} + \mathbf{HQH}^H]\}, \tag{2.9}$$

where $\mathbf{Q}$ is the input covariance matrix and P is the total power in the system [28]. A MIMO channel can be converted to parallel, independent SISO channels through a singular value decomposition (SVD) of the channel matrix [5]. The SVD yields $\min(n_T, n_R)$ parallel channels with gains corresponding to the singular values of $\mathbf{H}$.

The constant channel model is relatively easy to analyze, however, real wireless channels are not fixed or constant. Thus, a certain fading model must be considered to obtain the actual channel capacity. We here consider the spatial multiplexing BLAST system, where perfect CSI at the receiver is available and a zero mean spatially white distribution is considered at the transmitter [28]. Recall that, in this case, the channel matrix $\mathbf{H}$ is assumed to have independent identically distributed (i.i.d) complex Gaussian entries. As said in [4] and [5], the next step is to find the optimum input covariance matrix in the sense the ergodic capacity is maximized subject to a transmit power constraint. It was shown that the optimum input covariance matrix that maximizes ergodic capacity for this case is the scaled identity matrix, i.e. $\mathbf{Q} = \frac{P}{n_T}\mathbf{I}_{n_T}$ [4][5]. Therefore, in practice the transmit power must be divided equally among all the transmit antennas (a.k.a. uniform power allocation). Hence, the ergodic capacity is given by

$$C = \mathbb{E}\left\{\max \log\left\{\det\left(\mathbf{I}_{n_R} + \frac{P}{n_T}\mathbf{HH}^H\right)\right\}\right\}. \tag{2.10}$$

We will assume the uniform power allocation approach through the rest of the thesis.

2.1.3 Maximum-Likelihood Detection

Lattices are periodic arrangements of discrete points. Apart from their well-known use in pure mathematics, lattices have found applications in a wide variety of signal processing problems [29]. In particular, lattices are interesting for the design of MIMO detection algorithms, as it will be further seen throughout this thesis.

The system model of Fig. 2.1 shows that the receiving antennas see the superposition of all the transmitted signals affected by the different paths that form the channel. Here, all the possible n_T-dimensional transmitted vectors with symbols belonging to a previously known finite M-ary alphabet Ω can be represented as a lattice. Thus, the channel matrix can be seen as a basis (or generator) of the *lattice points*.

Given the received signal $\mathbf{x}$, the detection problem consists in determining the vector $\hat{\mathbf{s}}$ with the highest a posteriori probability of having been transmitted:

$$\hat{\mathbf{s}} = \arg \max_{s \in \Omega^{n_T}} P\{\hat{\mathbf{s}} = \mathbf{s} | \mathbf{x}\}, \tag{2.11}$$

where Ω^{n_T} denotes the set of possibly transmitted lattice points [30].

After applying Bayes' theorem, (2.11) turns into

$$\hat{\mathbf{s}} = \arg \max_{s \in \Omega^{n_T}} \frac{P\{\mathbf{x} | \hat{\mathbf{s}} = \mathbf{s}\} P\{\mathbf{s}\}}{P\{\mathbf{x}\}}. \tag{2.12}$$

Equation (2.12) is known as the *maximum-a-posteriori* probability (MAP) rule. If *a priori* equally likely inputs are assumed ($P\{\mathbf{s}\}$ is constant), the MAP rule turns into the *maximum-likelihood* (ML) detection rule as follows:

$$\hat{\mathbf{s}} = \arg \max_{s \in \Omega^{n_T}} P\{\mathbf{x} | \hat{\mathbf{s}} = \mathbf{s}\}. \tag{2.13}$$

Furthermore, if additive, white and Gaussian noise (AWGN) is considered, the ML detection rule is equivalent to solving the following least

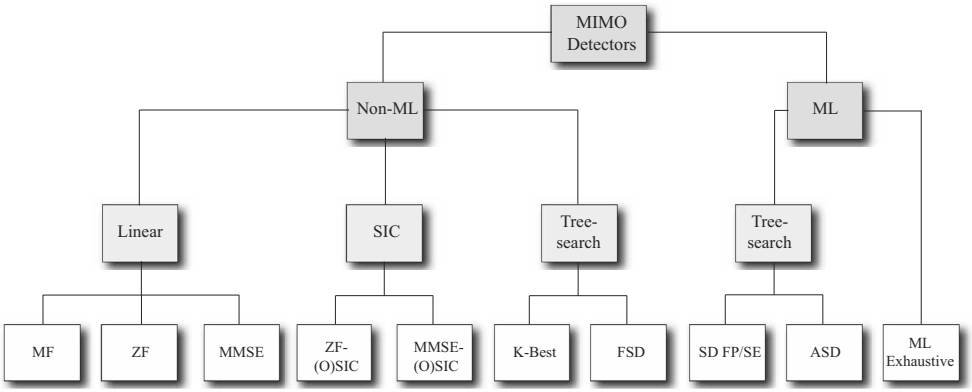


Figure 2.3. Classification of MIMO detection algorithms.

squares problem

$$\hat{\mathbf{s}} = \arg \min_{s \in \Omega^{n_T}} ||\mathbf{x} - \mathbf{H}\mathbf{s}||^2 , \tag{2.14}$$

where $\| \cdot \|$ denotes the 2-norm. The ML detector is widely known as the optimum detector in terms of lowest probability of vector error [1].

Due to the fact that all the possible $\mathbf{s}$ vectors belong to a finite n_T-dimensional lattice, a direct way to find the solution of (2.14) is performing an exhaustive search of points in the lattice and selecting the one that minimizes (2.14). This strategy leads to an algorithm with a computational cost exponentially growing with the number of transmit antennas.

The high complexity of exhaustive ML detection motivates the current research on efficient detection schemes for MIMO systems [13], ranging from the well-known linear detectors [31] to tree-search detection techniques [32]. Fig. 2.3 shows the classification of most of the existing MIMO detection strategies. In a first level, detection algorithms can be classified between ML (or optimal) methods and non-ML (or suboptimal) methods. A second classification depends on the detection strategy, which can be either in a linear or successive interference cancellation (SIC) way or via a tree search. The detection schemes shown in Fig. 2.3 will be described in detail throughout the following two sections.

Finally, it is useful to introduce the (spatial) *diversity order* of a re-

ceiver, which is defined as the asymptotic slope of the error probability on a log-log scale and depends on the MIMO channel statistics and on the transceiver scheme. The maximum diversity order in a spatial multiplexing system equals the number of receiving antennas (n_R) and is attained (among others) by ML detectors.

2.2 Linear and Successive Interference Cancellation Detectors

2.2.1 Matched Filter Detector

The *Matched Filter* (MF) detector appeared as an extension from the classical data detection in SISO channels [31]. The detection step is carried out just by multiplying the received vector by the transpose and conjugate of the channel matrix and quantizing the result to round it off to the closest symbol in the alphabet considered

$$\hat{\mathbf{s}} = \mathcal{Q}\{\mathbf{H}^H \mathbf{x}\}, \tag{2.15}$$

where $\mathcal{Q}(\cdot)$ stands for component-wise quantization. This algorithm exhibits near optimum behavior when the columns of $\mathbf{H}$ are nearly orthogonal, since it means that the several channels that exist in parallel are almost independent among themselves.

2.2.2 Zero-Forcing and Minimum Mean Square Error Detectors

The *Zero Forcing* (ZF) detector considers the signal from each transmit antenna as the target signal and the rest of signals as interferers [31]. The main goal of this detector is setting the interferers amplitude to zero, which is done by inverting the channel response and rounding the result to the closest symbol in the alphabet considered. When the MIMO channel matrix is square ($n_R = n_T$) and non-singular (invertible) the inversion step is performed just using the inverse of the channel matrix

$$\hat{\mathbf{s}} = \mathcal{Q}\{\mathbf{H}^{-1} \mathbf{x}\}. \tag{2.16}$$

However, when the channel matrix is tall ($n_R > n_T$), the pseudo-inverse of $\mathbf{H}$ is then used, what leads to the following inversion step

$$\hat{\mathbf{s}} = \mathcal{Q}\{(\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{x}\}. \tag{2.17}$$

The matrix that pre-multiplies the received vector is often called as *nulling matrix*. In the case of ZF detection, it equals either the inverse or the pseudo-inverse of $\mathbf{H}$, depending on its dimensions.

The ZF detector has the drawback of, in some cases, finding singular channel matrices that are not invertible. Another disadvantage is the fact that ZF focuses on cancelling completely the interferences, at the expense of enhancing the noise [31].

The *Minimum Mean Square Error* (MMSE) detector appeared with the aim of counteracting the noise enhancement problem of the ZF detector [31]. This method minimizes the error due to the noise and the interference combined by using the following detection step:

$$\hat{\mathbf{s}} = \mathcal{Q}\{(\mathbf{H}^H\mathbf{H} + N_o\mathbf{I})^{-1}\mathbf{H}^H\mathbf{x}\}, \tag{2.18}$$

where N_0 denotes the noise power.

2.2.3 Successive Interference Cancellation Detectors

The performance of already presented algorithms (ZF and MMSE) can be improved by using nonlinear techniques as symbol cancellation [33], leading to the well-known nulling and cancellation detectors. By using symbol cancellation, an already detected and quantized symbol from each transmit antenna is extracted out from the received signal vector, similarly to what is done in decision feedback equalization (DFE) or multiuser detection with successive interference cancellation (SIC). Therefore, as soon as a signal is detected, the next one will see one interferer less.

There are different ways to carry out SIC detection, one of them is through the use of the QR decomposition of the inverse of the nulling matrix. In the case of ZF, the channel matrix is decomposed into $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is unitary, $\mathbf{QQ}^H = \mathbf{I}$, and matrix $\mathbf{R}$ can be decomposed into an upper triangular $n_T \times n_T$ matrix, denoted by $\mathbf{R}'$, and a $(n_R - n_T) \times n_T$

matrix of zeroes. After this QR factorization and calling $\mathbf{y} = \mathbf{Q}^H\mathbf{x}$, the components of $\hat{\mathbf{s}}$ can be obtained as:

$$\hat{s}_i = \mathcal{Q}\left\{\frac{\left(y_i - \sum_{l=i+1}^{n_T} R'_{i,l}\hat{s}_l\right)}{R'_{i,i}}\right\}, \qquad i = n_T, \ldots, 1. \tag{2.19}$$

Note that since the nulling matrix for MMSE is $\mathbf{W} = (\mathbf{H}^H\mathbf{H}+N_o\mathbf{I})^{-1}\mathbf{H}^H$, the QR decomposition needed in this case to perform SIC detection is $\mathbf{W}^{-1} = \mathbf{QR}$.

2.2.4 Reordered Detection

Nulling and cancellation detectors have the drawback of error propagation to the next symbols to be detected when there has been any wrong decision in the already detected symbols. It can be shown that it is advantageous to perform the detection following a certain order, which can be sometimes different from the initial one. In fact, finding a suitable detection ordering has been shown to either improve SIC detection performance or to speed optimal (ML) tree-search detection, as will be further described in the section devoted to tree-search detection. Thus, the use of channel matrix preprocessing and ordering techniques before the detection is an interesting strategy in our context.

At the preprocessing stage, the MIMO channel matrix $\mathbf{H}$ is transformed via a matrix $\mathbf{P}$ into a new channel matrix $\widetilde{\mathbf{H}} = \mathbf{HP}$. In order to keep the system in (2.7) unaltered, it is convenient to define the new detected symbol as $\mathbf{z} = \mathbf{P}^{-1}\mathbf{s}$. Therefore, the received signal vector $\mathbf{x}$ can be rewritten as

$$\mathbf{x} = \mathbf{HPP}^{-1}\mathbf{s} + \mathbf{v} = \widetilde{\mathbf{H}}\mathbf{z} + \mathbf{v}. \tag{2.20}$$

The structure and properties of the transformation matrix $\mathbf{P}$ depend on the preprocessing algorithm used.

A quite simple reordering method calculates the norm of all the channel matrix columns and reorders them in ascending order. This is an approach to detect first the symbols with the highest SNR, i.e., the most reliable ones. In [30], an optimal ordering was proposed by the BLAST laboratories (VBLAST) and employed for nulling and cancellation detection. The

resulting scheme was named ordered SIC (OSIC) detector. The VBLAST method improves the performance of the column-norm-based ordering technique at the expense of increasing its complexity. Both, column-norm-based and VBLAST orderings are purely based on the knowledge of the channel matrix coefficients, thus, there is no need to perform a new channel reordering as long as the channel remains unchanged (usually during the transmission of several signal vectors).

There are other preprocessing strategies that also use the received signal vector to decide on the most suitable channel ordering, such as the ordering proposed in [34] or the gradient-based ordering described in [35]. The latter techniques, however, require a new ordering to be performed every time a new signal vector has to be detected, which increases the computational cost necessary for the detection stage. Note that, for instance, detection strategies that involve SIC detection would require the recalculation of the QR decomposition of the channel matrix for every signal to be detected.

In addition, several channel matrix preprocessing techniques based on lattice-reduction (LR) [21] have been widely employed to improve MIMO detectors performance [17][18]. These strategies will be included and detailed in a later chapter of the thesis, together with the steps of the aforementioned VBLAST ordering. Also, a QR-based low-complexity implementation of the VBLAST method will be proposed.

2.2.5 Performance Comparison

The performance of the above described linear and SIC detectors was assessed by means of simulations to obtain bit-error-rate (BER) curves versus SNR. Since current wireless standards that combine MIMO with OFDM, such as WLAN, WiMAX or LTE, make use of QAM constellation schemes (as these are the most spectrally efficient), the 16-QAM alphabet was selected for the simulation. The size of the MIMO system was 4×4 and the detection results were averaged over 10^4 channel matrix realizations.

Fig. 2.4 shows the BER curves for ZF and MMSE detection together with the curves of the SIC versions of both algorithms. The MMSE-SIC al-

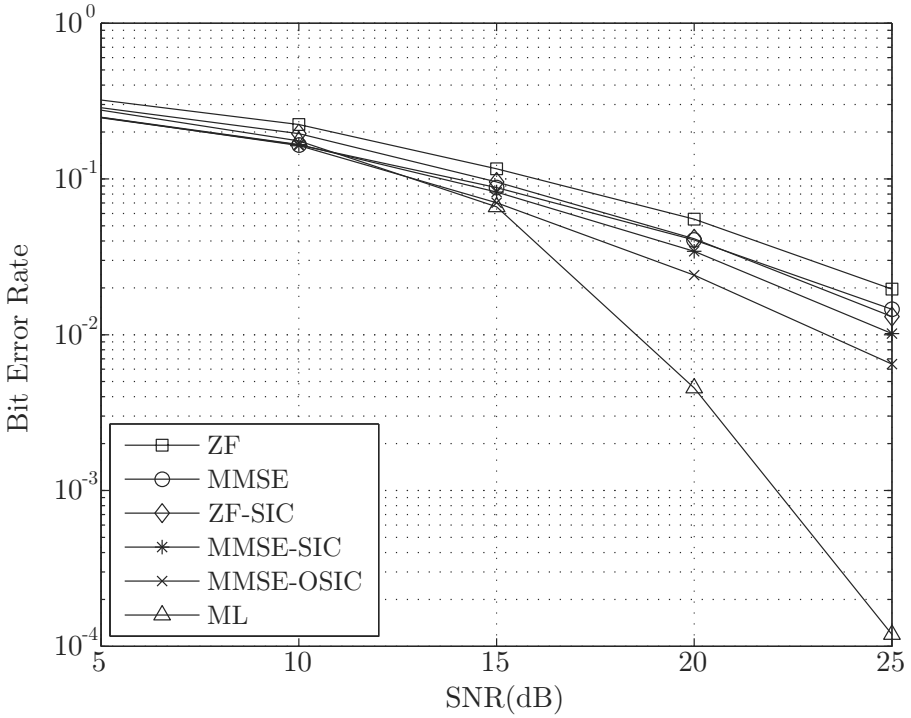


Figure 2.4. Bit Error Rate of the classical detectors in a 4×4 MIMO system with 16-QAM.

gorithm was also evaluated considering a previous channel matrix ordering based on the column norms; this scheme is labelled as MMSE-OSIC. It can be seen that the SIC schemes outperform the linear detection algorithms, as expected. The MMSE-OSIC decreases the BER attained by the plain MMSE-SIC. Note that this result gives a preliminary idea of the usefulness of channel matrix preprocessing to improve MIMO detectors performance.

Nevertheless, although the above presented linear and SIC algorithms are interesting due to their low complexity, their performance is still far away from the optimal ML performance. This fact motivates the search for better performance detectors, such as the tree-search-based detectors, which are described in the next section.

2.3 Tree-Search-Based Detection/Sphere Decoding

The ML detection rule (2.14) can be seen as the minimization of the n_T-dimensional squared distance between vector $\mathbf{x}$ and vector $\mathbf{Hs}$. The original idea behind the well-known *sphere decoding* (SD) methods comes from the expression of the n_T-dimensional squared Euclidean distance in (2.14) as an addition of one-dimensional squared Euclidean distances [11]. Taking into account this equivalence, SD methods intend to reach the ML solution with lower complexity than the exhaustive search. This is done by looking for the ML solution just within a subset of the total M^{n_T} possible vectors. This subset is a n_T-dimensional hypersphere centered at the received signal vector [12][36] with a certain radius. All the insights of SD are shown graphically and mathematically throughout this section.

2.3.1 Sphere Decoding Fundamentals

The main idea of SD methods relies on, instead of performing an exhaustive search over the total n_T-dimensional lattice points, limiting the search for the solution to only the lattice points located at a distance of the received vector lower than a given maximum distance [12], called *sphere radius* (D). The sphere radius constraint can be included in the ML detection rule as follows

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \Omega^{n_T}} \{||\mathbf{x} - \mathbf{Hs}||^2 \leq D\}. \tag{2.21}$$

For instance, Fig. 2.5 shows the possible transmitted vectors (lattice points) for a 2×2 MIMO system using a binary phase shift keying (BPSK) constellation ($M = 2$) and assuming $\mathbf{H} = \mathbf{I}$. In this case, $\mathbf{Hs} = \mathbf{s}$ and, thus, the received vector can be easily related to the transmitted components. It can be seen that, if a sphere radius D is chosen, there are two lattice points that lie inside the sphere. These two points represent the *candidate solutions* that would fulfill (2.21). Thus, the ML solution would then be the closest lattice point of the list of candidate points to the received vector $\mathbf{x}$, which is labelled in Fig. 2.5 as *ML*.

Regarding complexity saving, in this simple case the search reduces from among 4 candidates to just among 2 (i.e. it goes to the half). More

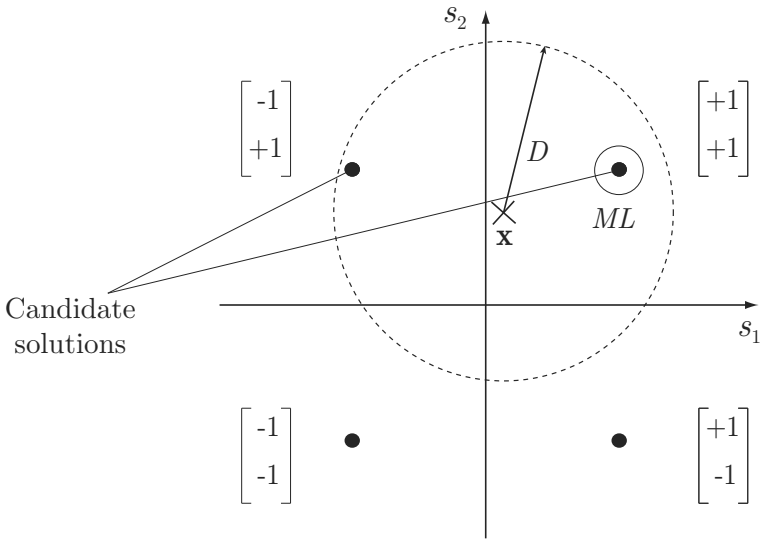


Figure 2.5. Decoding sphere of radius D for limiting the candidate lattice points in a 2×2 MIMO system using a BPSK constellation.

generally, as the system and/or constellation size gets higher, the complexity reduction achieved by SD methods increases substantially [12]. This complexity reduction, however, can be attained provided a suitable sphere radius has been previously selected, which is not a straightforward task at all.

Similarly to what is done in SIC detection, a QR factorization of the channel matrix ($\mathbf{H} = \mathbf{QR}$) is employed. This allows to transform the problem (2.14) into an equivalent one that can be solved using a tree structure [11]. The properties and dimensions of $\mathbf{Q}$ and $\mathbf{R}$ are the ones described in Section 2.2.3 ($\mathbf{Q}$ unitary and $\mathbf{R}$ upper triangular). For the sake of simplicity, a system with $n_T = n_R$ is assumed.

In case of multiplying (2.14) by $\mathbf{Q}^H$ and calling $\mathbf{y} = \mathbf{Q}^H\mathbf{x}$, the ML problem (2.14) can be equivalently expressed as

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s}\in\Omega^{n_T}} \{||\mathbf{y} - \mathbf{Rs}||^2\}, \tag{2.22}$$

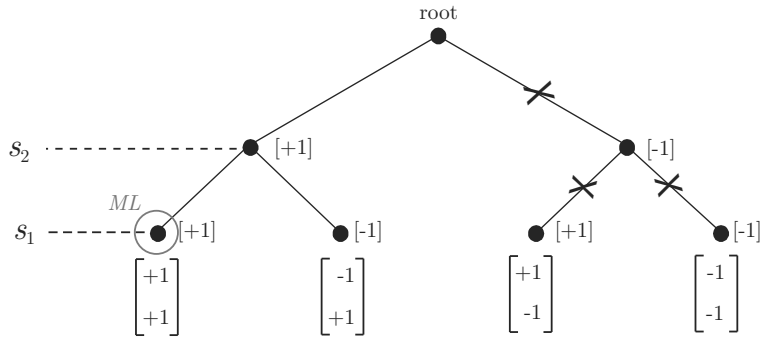


Figure 2.6. Decoding tree associated to the decoding sphere of Fig. 2.5.

and the SD approach in a more detailed way as

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \Omega^{n_T}} \left\{ \sum_{i=1}^{n_T} \left| y_i - \sum_{j=i}^{n_T} R_{ij} s_j \right|^2 \leq D \right\}, \tag{2.23}$$

where the triangular structure of $\mathbf{R}$ has been exploited.

In a first step, a search-tree is built containing all the candidate lattice points associated to the problem to be solved. The tree must have as many levels as transmit antennas and each symbol value is represented by a tree node. The tree-paths are built by connecting nodes and stand for candidate solutions. For instance, a tree-path containing selected symbols from the root up to level i has the form

$$S^{(i)} = [s_i, s_{i+1}, \ldots, s_{n_T}]^T. \tag{2.24}$$

Fig. 2.6 depicts the decoding tree associated to the decoding sphere of Fig. 2.5, which has two detection levels and four possible candidate solutions.

In order to solve (2.23) the tree-search starts from the root (assumed in level $n_T + 1$) and every time that the search descends from a node in level i (called *parent node*) to the nodes in level $i - 1$ that are connected to it (called *children nodes*), the partial Euclidean distances (PED) of the children nodes are computed as follows:

$$e_i(S^{(i)}) = y_i - \sum_{j=i}^{n_T} R_{ij} s_j, \tag{2.25}$$

which are also known as *branch weights*. Then, it is said that the parent node has been *expanded*.

The calculation of the branch weights allows to update the accumulated PED of each path as follows

$$T_i(S^{(i)}) = T_{i+1}(S^{(i+1)}) + |e_i(S^{(i)})|^2 \tag{2.26}$$

assuming that the root node starts with accumulated PED equal to zero:

$$T_{2n_T+1}(S^{(n_T+1)}) = 0. \tag{2.27}$$

Following with the example, the candidate solutions discarded in Fig. 2.6 (both on the right side of the tree) would have accumulated PED exceeding the value of D and, for this reason, they are outside the sphere in Fig. 2.5. Note that when the accumulated PED of a parent node is higher than D, its children nodes can be pruned in advance, resulting in a faster tree-search. Therefore, the average number of expanded nodes can be used as a measure of average computational complexity. To perform an efficient tree-search, the ML solution should be found after having expanded the minimum number of nodes.

Many different tree-search strategies have been proposed during the last years, some of which can be found in [12][14][32][37]. Generally, tree-search strategies can be classified into two main types: depth-first-based and breadth-first-based. In the depth-first algorithms, the tree is explored from top to bottom, i.e. starting at the root and ending at the bottom nodes (*leaves*). No children nodes located on the right side of the tree can be explored before those on the left are already explored. Fig. 2.7 illustrates this kind of search.

In the breadth-first algorithms the tree is explored descending level by level up to the leaf nodes, but every node in a certain level has to be visited before starting to visit nodes in the level below. Fig. 2.8 depicts the general idea behind breadth-first algorithms.

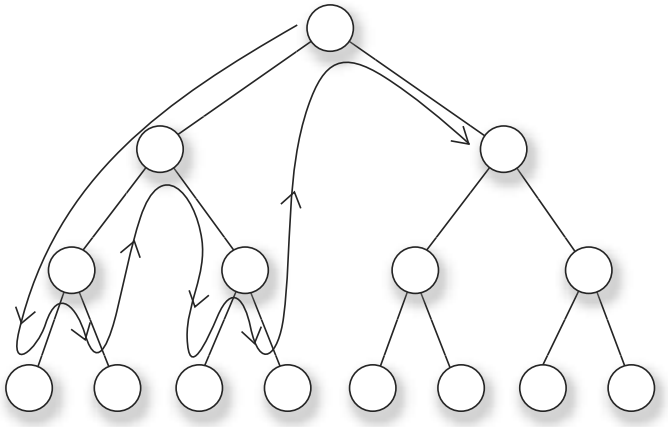


Figure 2.7. Decoding tree where a depth-first strategy is followed.

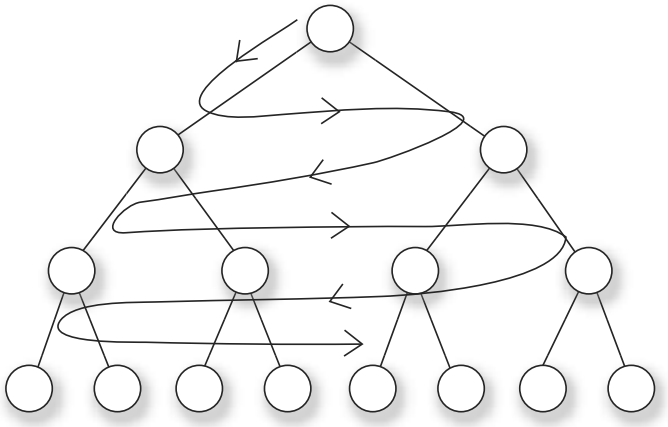


Figure 2.8. Decoding tree where a breadth-first strategy is followed.

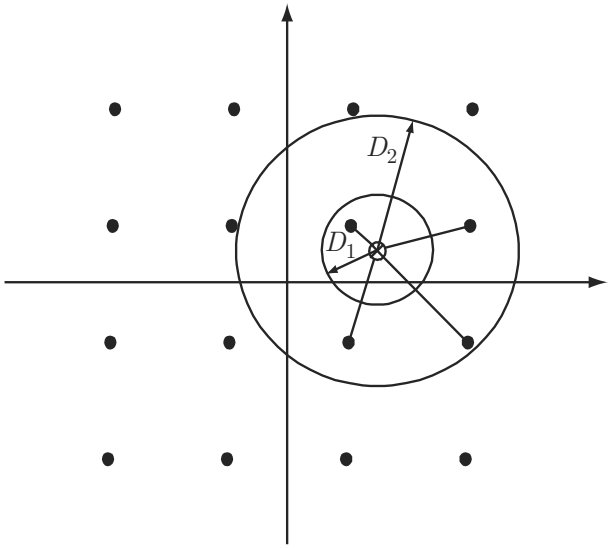


Figure 2.9. Comparison between the number of candidate points inside spheres of radius D_1 and D_2.

As said above, a suitable sphere radius is generally needed for getting the ML solution expanding as few nodes as possible. However, if a too small sphere radius is chosen, there can be no candidate solutions and the algorithm will not perform correctly. On the other hand, if a too large sphere radius is selected, too many candidate points may be found and the complexity of the algorithm can equal the one of an ML exhaustive search, without any advantage over existing methods. For instance, Fig. 2.9 shows the candidate points inside two spheres of different radius, for a two-dimensional case with $D_1 < D_2$. Note that D_1 provides just one candidate point whereas D_2 leads to a search among four candidate points.

There are several methods to estimate the sphere radius [12]. A useful radius estimate can be obtained by calculating the distance between the received vector and the solution provided by a low-complexity detection method such as ZF or MMSE. This reads

$$\hat{D} = \left|\left|\mathbf{y} - \mathbf{R}\hat{\mathbf{s}}^{\mathrm{B}}\right|\right|^2, \tag{2.28}$$

where $\hat{\mathbf{s}}^{\mathrm{B}}$ is the solution obtained via a suboptimal method such as ZF

or MMSE. This radius estimator guarantees at least one point inside the sphere.

Other authors suggested choosing a scaled version of the noise variance as a candidate radius, since it seems reasonable to consider that the transmitted vector will be moved away from its original position a distance related to the variance of the noise present in the system. Nevertheless, no general estimator adequate for every single particular case has been yet found.

2.3.2 Fincke-Pohst and Schnorr-Euchner Enumerations

One of the first-appeared depth-first SD algorithms is the one based on the *Fincke-Pohst* enumeration (FP-SD) [38]. Let us show how this method works with the example shown in Fig. 2.10, where a decoding tree for a 3×3 MIMO system with a BPSK symbol alphabet is represented. An appropriate sphere radius is assumed before starting the search for the solution. In order to follow the FP enumeration, the tree-nodes are visited following the order given by the node numbers, which makes the tree be traversed from left to right and from top to bottom. Some branches and nodes have been depicted in gray color to give examples of candidate solutions that have been discarded because their PED exceed the sphere radius. It can be seen that, after having completed the search, there are just three possible solutions in the tree (i.e. inside the sphere), which correspond to the leaf nodes numbered as 5, 6 and 9. The ML estimate would be the path with the minimum PED among these final candidates.

The main drawback of this algorithm is its dependency on the sphere radius which, as said before, can sometimes lead to a too small or even negligible complexity reduction.

The *Schnorr-Euchner* SD (SE-SD) [39] also performs a depth-first search but it modifies the FP algorithm to further decrease the number of expanded nodes. Instead of exploring the tree naively from left to right, the SE-SD computes the branch weights of the children nodes of a certain node before expanding it. Then, it explores the children nodes according to the increasing order of their branch weights. This improvement leads to

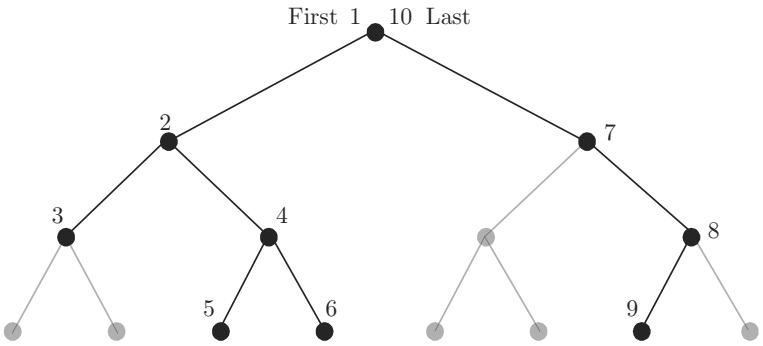


Figure 2.10. Decoding tree for a 3×3 MIMO system with a BPSK constellation which follows a Fincke-Pohst search strategy.

reaching valid leaf nodes faster than with the FP-SD.

Note that, however, this first modification is not sufficient, as the number of branch weights to be computed remains still the same than in the FP-SD case. To overcome this problem, the SE-SD updates the search radius adaptively every time a new leaf node is reached. This way, no valid points are discarded since, after having explored a certain point in the search set, the algorithm must be only interested in visiting those points that are even closer to the target than the just-visited points.

Therefore, the SE-SD allows working without any previous sphere radius selection since the radius can be initially set to infinity and be updated every time a new leaf node is found. Thus, making a good choice of the sphere radius is no longer a critical factor.

A further pruning of useless candidate solutions was introduced in [40]. It is based on using bounds for the PED in the tree-levels still to be explored. These bounds can help to further discard useless candidate solutions in the SE-SD. This way, the search radius is decreased faster. A concrete implementation of this idea was firstly proposed in [41] and improved in [42].

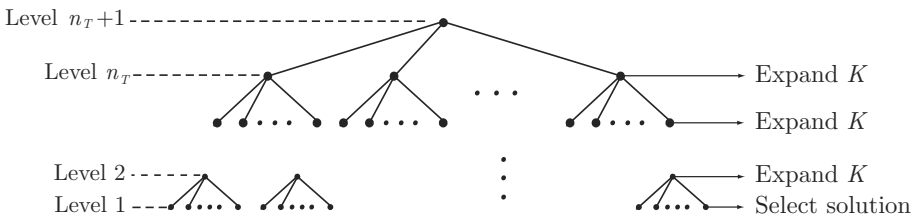


Figure 2.11. Decoding tree of the K-Best algorithm.

2.3.3 K-Best Sphere Decoder

The K-Best SD [14] (also known as K-Best tree-search detector or QRD-M detector) is a breadth-first algorithm that expands only those K survivor nodes that show the smallest accumulated PED at each level of the decoding tree (see Fig. 2.11). Besides having a different search strategy than the FP-SD and SE-SD methods, the K-Best SD includes another considerable difference with respect to FP-SD: the candidate solutions are no longer discarded using a sphere radius but setting a constraint in the number of survivor paths per level. This constraint is represented by the K parameter. Hence, strictly speaking, the K-Best SD is not an actual SD method.

Once the PED of each path at a certain level has been computed with (2.25) and (2.26), the $T_i(S^{(i)})$ values are sorted in ascending order and the K paths having the minimum PED values are stored. Then, the search continues from these K survivor paths and follows the same strategy until the lowest tree-level is reached. The detected signal vector $\hat{\mathbf{s}}$ is given by the path from the root up to the leaf node with the smallest accumulated PED.

The main advantage of this method is that its maximum number of paths is limited, yielding a fixed computational effort. In addition, the complexity and memory requirements coincide at every level. This parallelism among detection levels allows an easier hardware implementation of the algorithm. Several variants of this algorithm also include a sphere radius in order to reduce the number of explored paths [43][44], however, the complexity becomes non-fixed and unknown.

As it is shown in [45], it is more likely to discard the ML solution at

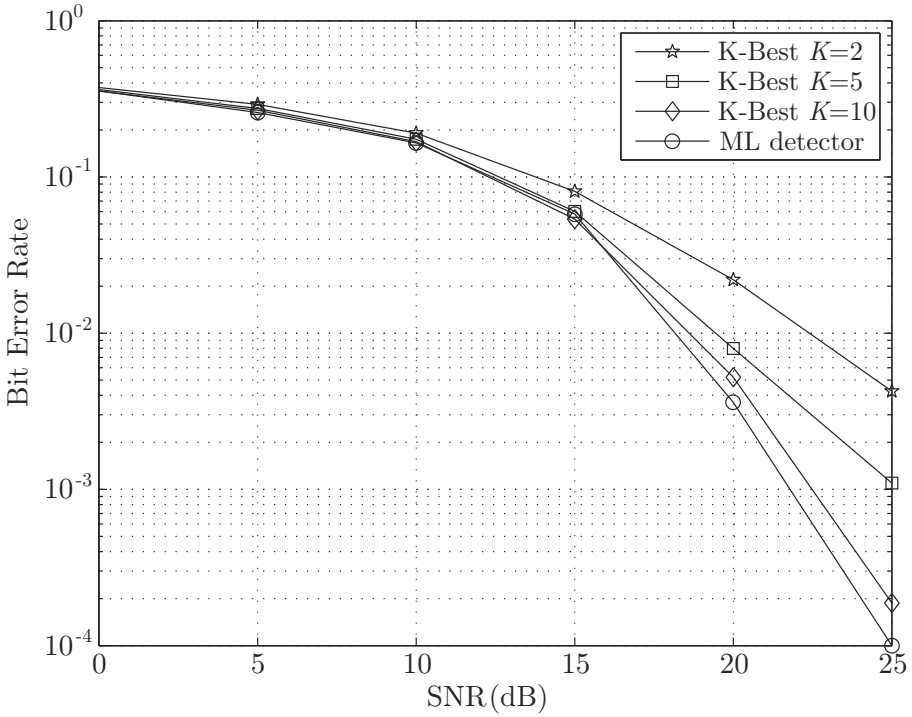


Figure 2.12. Bit Error Rate of the K-Best detector for different values of K in a 4×4 MIMO system with 16-QAM symbols.

early decoding stages, since in the lowest levels the accumulated PED is closer to the final total distance. Thus, the method can be also modified to work with different K values at different decoding levels, which is called as dynamic K-Best detection. Although the dynamic K-Best approach keeps the advantage of fixed and predictable complexity, it has the drawback of not having the same complexity and memory needs at every level. Therefore, the parallelism among detection levels can no longer be exploited as in the case of the K-Best method.

Fig. 2.12 shows several curves of BER versus SNR for the K-Best detector with three different values of K. A 4×4 MIMO system with 16-QAM symbols was considered for the simulations. Note that higher values of K improve the performance of the K-Best detector.

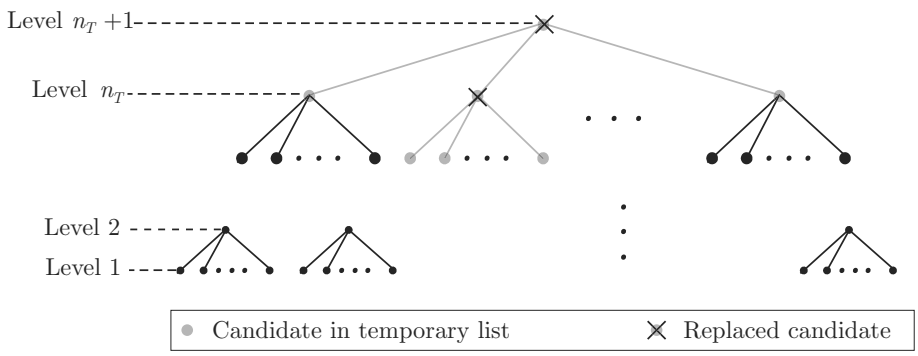


Figure 2.13. Decoding tree of the ASD algorithm.

2.3.4 Automatic Sphere Decoder

The *Automatic sphere decoder* (ASD) was initially proposed in [37]. It is a breadth-first algorithm that, as the K-Best SD, does not make use of a sphere radius to solve the detection problem. It decreases the complexity of the exhaustive ML search by storing a list of candidates which defines the limit between the already explored part of the tree and the non-yet-explored part.

At the beginning of the algorithm, the list only contains the root node and its associated accumulated PED (equal to zero). In each iteration, the method selects and expands the node inside the list with smallest PED. This just expanded node, is removed from the list and replaced by its children nodes. Once a leaf node is reached and selected for the next expansion, the algorithm ends and returns its associated path as the solution. In Fig. 2.13 it can be seen an example decoding tree where just the root node and a node located at the n_T level have been expanded and replaced by their children nodes. The already visited branches are depicted in gray.

The main disadvantage of this method is the need for a variable-size list of candidate solutions, which can be a drawback for hardware implementations. In addition, its complexity has be shown to be higher than these of depth-first methods.

2.3.5 Fixed-Complexity Sphere Decoder

In [46] the authors proposed a MIMO detection strategy intended to overcome the two main problems of depth-first SD methods from an implementation point of view: their variable complexity and their sequential nature. This algorithm was called fixed-complexity SD (FSD) and combines a preprocessing stage followed by a predetermined tree-search composed of two different stages: a full expansion of the tree (FE) in the first (highest) T levels and a single-path expansion (SE) in the remaining tree-levels $n_T - T$ [46].

The symbols are detected following a specific ordering also proposed in [46] which is based on the following reasoning: if all the symbol possibilities are explored in one tree-level (FE), the robustness of the signal at such level is not relevant to the final performance, as no candidates are being discarded. Therefore, the signals that suffer the largest noise amplification are placed at the levels where a FE is performed. On the other hand, the signals that suffer the smallest noise amplification are placed at the levels associated to the SE, i.e. at those levels where the tree will be pruned.

The FSD ordering iteratively orders the n_T columns of the channel matrix $\mathbf{H}$. At the ith iteration only those components still to be detected are considered. If the corresponding tree-level belongs to the SE stage, the component of $\mathbf{s}$ with the smallest post-detection noise amplification is selected and placed in this tree-level. Otherwise, the signal with the largest noise amplification is selected instead.

The particular steps carried out for each iteration are the following. First, the pseudoinverse of the matrix containing only the columns of the indexes not selected yet ($\mathbf{H}_i$) is computed:

$$\mathbf{H}_i^{+} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H, \qquad i = n_T, \ldots, 1. \tag{2.29}$$

Then, if a symbol for the FE is searched, the index that satisfies the following is selected

$$k = \arg\max \|\mathbf{H}_i^{+}\|^2, \tag{2.30}$$

otherwise, the selected index must fulfill

$$k = \arg\min \|\mathbf{H}_i^{+}\|^2. \tag{2.31}$$

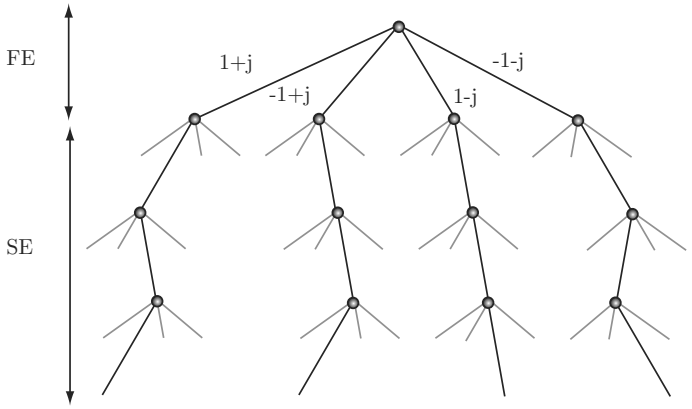


Figure 2.14. Decoding tree of the FSD algorithm for a 4×4 MIMO system with QPSK symbols.

Although the FSD does not guarantee to find the ML estimate in all cases, it achieves the maximum detection diversity if the following value of T is chosen [47]:

$$T \geq \sqrt{n_T} - 1. \tag{2.32}$$

Fig. 2.14 shows the search tree of the FSD algorithm for the case with $n_T = 4$ ($T = 1$) and QPSK symbols. At the FE stage, for each survivor path, all the possible values of the constellation are assigned to the symbol at the current level. The SE stage starts from each retained path and proceeds in the tree calculating the solution of the remaining SIC problem (see subsection 2.2.3).

As seen in Fig. 2.15, which shows a performance comparison between the BER of optimal ML decoders and those achieved by the FSD for different modulations, the performance of the FSD comes stunningly close to that of an optimal ML detector.

The main disadvantages of this method are, on the one hand, the computational cost necessary for the preprocessing (which involves the calculation of several pseudoinverse matrices) and, on the other hand, the fact that the number of expanded nodes can be often higher than that of a depth-

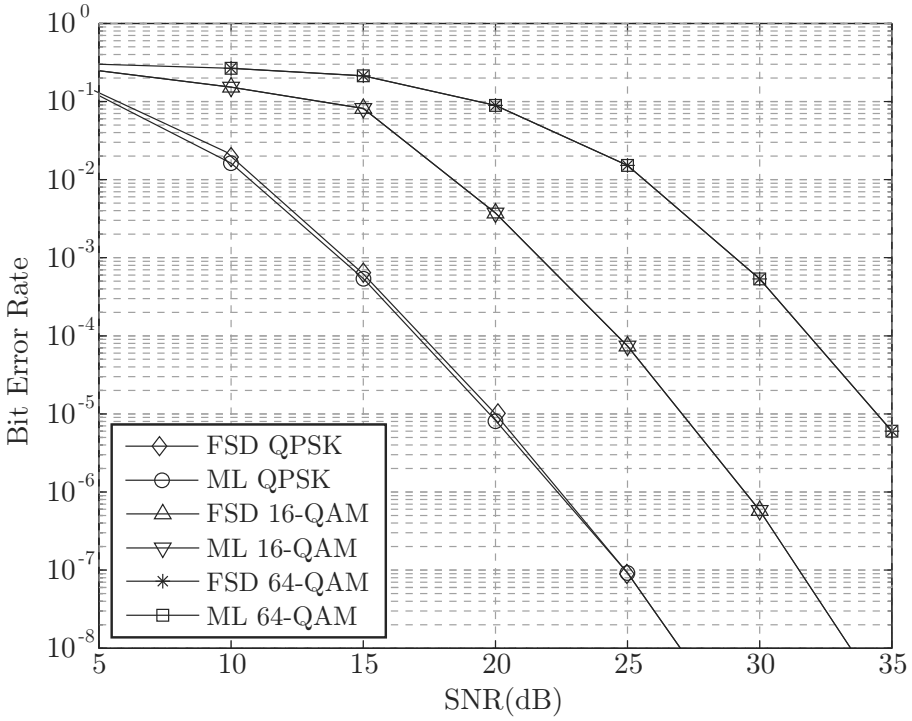


Figure 2.15. BER achieved by the FSD in a 4×4 MIMO system using QPSK, 16-QAM and 64-QAM compared to ML performance.

first SD [48]. Also, although it guarantees near-ML average performance, its solution might not coincide with the ML estimate in some cases.

2.4 MIMO-Bit-Interleaved Coded-Modulation Systems

Another closely related scenario where the above described MIMO detection techniques are useful is the case when channel coding is inserted, such as in MIMO-BICM [49][50]. In fact, the system description (2.1) can be easily extended to describe a MIMO-BICM scheme with the same number of antennas. Therefore, it is well justified to consider such systems in this dissertation.

2.4.1 System Model and Log-Likelihood-Ratios

In a MIMO-BICM system (such as the one shown in Fig. 2.16), the sequence of information bits is encoded using an error-correcting code and passed through a bitwise interleaver Π prior to being demultiplexed and mapped to complex-valued transmit symbol vector $\mathbf{s} = (s_1, \ldots, s_{n_\mathrm{T}})^T$. Again the symbols s_i are taken from a constellation Ω of size $|\Omega| = M$ and hence carry $\log_2 M$ code bits each.

Slightly deviating from previously used notation, we here denote the baseband equivalent model for received vector by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{v}. \tag{2.33}$$

In a BICM receiver, the demodulation and channel decoding are not performed jointly but in two differentiated stages. At the receive side, the demodulator uses the model (2.33) to compute soft information about the code bits in terms of log-likelihood ratios (LLRs). The delivered soft information is used by the channel decoder to make final decisions about the transmitted sequence bits.

In this system model, $x_{j,b}$ denotes the bth bit in the bit label of symbol s_j and the LLR for each transmitted coded bit equals [51]:

$$L_{j,b} = \log \frac{f(x_{j,b} = 1|\mathbf{y}, \mathbf{H})}{f(x_{j,b} = 0|\mathbf{y}, \mathbf{H})}, \tag{2.34}$$

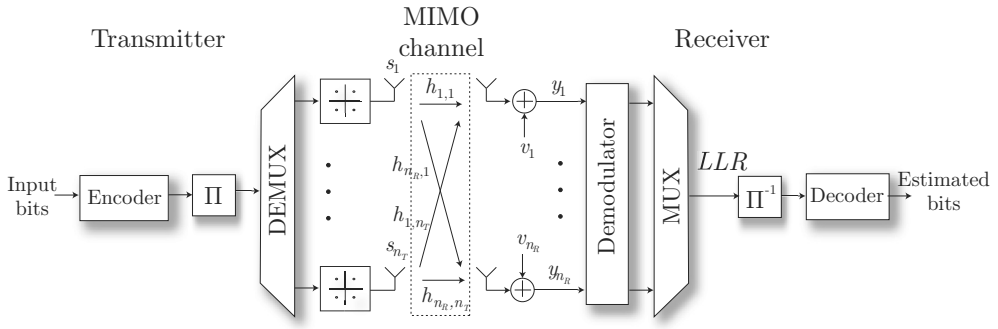


Figure 2.16. Block diagram of a MIMO-BICM system.

where $f(x_{j,b}|\mathbf{y},\mathbf{H})$ is the probability mass function of the coded bits $x_{j,b}$ conditioned on $\mathbf{y}$ and $\mathbf{H}$.

Using the max-log approximation [52], the LLR of the bth bit of the symbol in layer j is calculated as:

$$L_{j,b} = \frac{1}{\sigma^2}\left[\min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(0)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 - \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(1)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2\right], \tag{2.35}$$

where $\mathcal{X}_{j,b}^{(c)}$ denotes the set of symbol vectors for which the bth bit in layer j equals c.

The hard-output ML detection problem [1] can be shown to provide one of the two minima in (2.35), i.e.,

$$\mathbf{s}^{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\Omega^{n_\mathrm{T}}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2, \quad d^{\mathrm{ML}} = \|\mathbf{y}-\mathbf{H}\mathbf{s}^{\mathrm{ML}}\|^2. \tag{2.36}$$

Denote by $x_{j,b}^{\mathrm{ML}}$ the bth bit associated with s_j^{ML}. For each j and b, the second minimum in (2.35) can be computed as

$$\bar{d}_{j,b} = \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(\overline{x}_{j,b}^{\mathrm{ML}})}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2, \tag{2.37}$$

where $\overline{x}$ denotes the complement of bit x. Note that $\mathbf{s}\in\mathcal{X}_{j,b}^{(\overline{x}_{j,b}^{\mathrm{ML}})}$ represents the counter-hypothesis to the ML solution for bit b in layer j.

Once (2.36) and (2.37) have been calculated, the LLRs are obtained as

$$L_{j,b} = \frac{1}{\sigma^2}\left(d^{\mathrm{ML}} - \bar{d}_{j,b}\right)\left(1 - 2x_{j,b}^{\mathrm{ML}}\right), \tag{2.38}$$

where the term at the end adjusts the sign depending on whether d^{ML} corresponds to the first or the second minimum in (2.35).

As in the hard-output detection case, there are several methods to avoid an exhaustive search over the n_{T}-dimensional set $\Omega^{n_{\mathrm{T}}}$ in the ML problem (2.36). Some methods that are interesting in our context are those based on tree-search detection. The basics of tree-search-based soft demodulation are introduced next.

2.4.2 Tree-Search-Based Soft Demodulation

Tree-search detection methods can be employed to obtain the max-log-approximated LLRs efficiently. The straight way is to perform a *repeated tree-search* (RTS) [53]. In this case, an initial tree-search detection is carried out first to obtain $\mathbf{s}^{\mathrm{ML}}$ and d^{ML} (2.36). Then, $n_T \times \log_2 M$ new tree-searches are performed, each of them forcing the detector to keep one of the bits $\overline{x}_{j,b}^{\mathrm{ML}}$ fixed. The RTS strategy is very straightforward and intuitive but it performs several redundant calculations throughout the tree.

In [52], a list-based sphere decoding (LSD) scheme was proposed to approximate the sets $\mathcal{X}_{j,b}^{(0)}$ and $\mathcal{X}_{j,b}^{(1)}$ by a list of candidates with a certain size, smaller than the whole set Ω^{n_T}. Two parameters traded the complexity of the method versus performance: the list size and the sphere radius, both selected in a somewhat ad-hoc manner. There are other variants of this method which also set a constraint in the number of solutions per level, similarly to what is done in the K-Best tree-search detector. This approaches are known as K-Best LSD [25].

An efficient way to calculate the exact max-log LLRs without neither list-size constraints nor sphere radius was proposed in [54]. This scheme obtains all the necessary distances by performing a *single tree-search* (STS) and was shown to achieve meaningful results especially when combined with LLR clipping. However, its computational cost varies depending on the channel matrix and it is based on a purely sequential tree-search, which

is clearly a drawback for parallel implementation.

Focusing on detection methods with a fixed number of visited nodes, in [55] and [56] the hard-output FSD described in Section 2.3.5 was extended to provide soft information. The method obtains an improved list of candidates with respect to the one in [52]. Between the two soft-output-FSD (SFSD) proposals, the algorithm shown in [55] achieved good demodulation results in a turbo encoded system expanding a lower number of candidates than the one in [56].

In this thesis we paid special attention to the SFSD scheme, which will be detailed in the chapter devoted to efficient soft-output detection.

2.5 Multiuser MIMO-OFDM Communication Systems

As introduced in Chapter 1, despite the use of MIMO systems was traditionally intended for point-to-point communication, multiuser MIMO (MU-MIMO) technology is expected to play a key role in next generation cellular systems. In addition, there are some algorithms related to MIMO detection that can be also employed for some of the necessary tasks in multiuser signal precoding and vice versa. Thus, the MU-MIMO scenario will be also considered in this thesis.

2.5.1 System Model

We consider the downlink of a multiuser MIMO-OFDM communication from a base station (BS) with N antennas to $K \leq N$ single-antenna users (MS) (see Fig. 2.17). According to the technical specifications in LTE Release 8 [57], not all the subcarriers are occupied and the number of used subcarriers can be varied. Thus, the same set of N_c subcarriers are employed by all the users and the unused carriers are placed at the edge of the occupied bandwidth in order to reduce the requirements of analog filters.

The baseband received signal at the kth user for the mth subcarrier

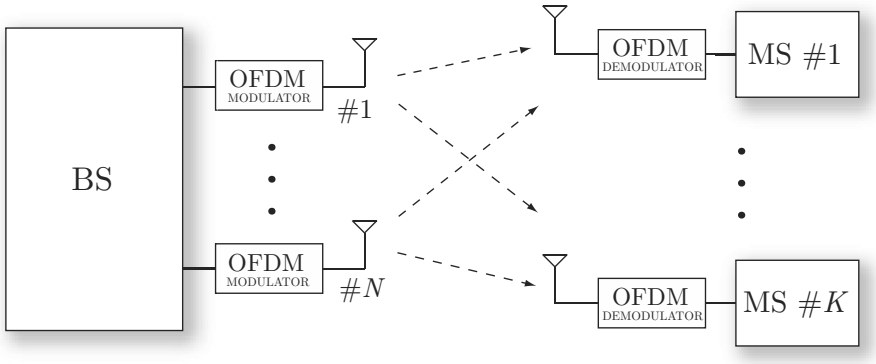


Figure 2.17. Multiuser MIMO-OFDM communication system from a BS with N antennas to K single-antenna users.

can be expressed as:

$$x_k[m] = \mathbf{h}_k^T[m]\mathbf{s}[m] + v_k[m], \tag{2.39}$$

where vector $\mathbf{s}[m] = (s_1[m], \ldots, s_N[m])^T$ includes the precoded information symbols for the mth subcarrier, $\mathbf{h}_k[m]$ is the N-elements channel vector for the mth subcarrier, and $v_k[m]$ is the received noise for the kth user at the mth subcarrier. Elements of $\mathbf{h}_k[m]$ contain the signal fading from each transmitter antenna to the kth user for the mth subcarrier. We assume again block-fading channels, constant on blocks of duration L_{ch} symbols, and changing according to some ergodic statistics from block to block.

The received signal for the K users of the system at the mth subcarrier can be expressed in a more compact way by means of a vector $\mathbf{x}[m] = (x_1[m], \ldots, x_K[m])^T$ as:

$$\mathbf{x}[m] = \mathbf{H}[m]\mathbf{s}[m] + \mathbf{v}[m], \tag{2.40}$$

turning into practically the same expression as the one of the BLAST system (2.1).

In the system considered, the users cannot cooperate to detect the received signals. Therefore, multiuser interference must be cancelled at the transmitter, which can be feasible if $N \geq K$ and CSI is available at the BS.

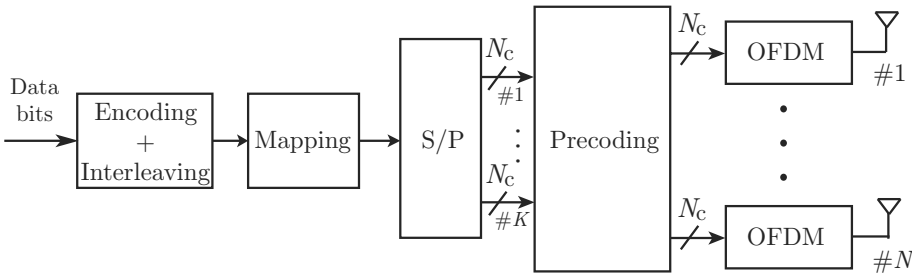


Figure 2.18. Block diagram of the transmitter of a multiuser MIMO-OFDM system with N antennas, K users and N_c subcarriers.

Multiuser interference is cancelled using signal precoding techniques. The block diagram of the transmitter including the precoding stage is illustrated in Fig. 2.18.

The type of precoding sets the way in which vector $\mathbf{s}$ is built from the constellation points to be transmitted. For the sake of simplicity, we drop the m index and focus on the original signal, precoded signal and channel matrix for a given subcarrier, since the precoding process needs to be performed for all subcarriers similarly. Moreover, the real-valued form of the system (see Section 2.1.1) is employed hereafter.

In what follows, the problem of precoding and the most employed precoding techniques are described in detail.

2.5.2 Vector Perturbation Precoding

The vector perturbation method [58] aims at finding the transmit signal that requires minimum power as:

$$\mathbf{s} = \mathbf{H}^{+}(\mathbf{s}' + \mathbf{p}), \tag{2.41}$$

where the perturbation vector is calculated as

$$\mathbf{p} = \arg \min_{\mathbf{p}' \in A\mathbb{Z}^{2K}} \|\mathbf{H}^{+}(\mathbf{s}' + \mathbf{p}')\|^2. \tag{2.42}$$

Note that (2.42) can be seen as a search for the point $\mathbf{H}^{+}\mathbf{p}'$ that is closest to $-\mathbf{H}^{+}\mathbf{s}'$ in the lattice $A\mathbb{Z}^{2K}$. This search for a $2K$-dimensional lattice point

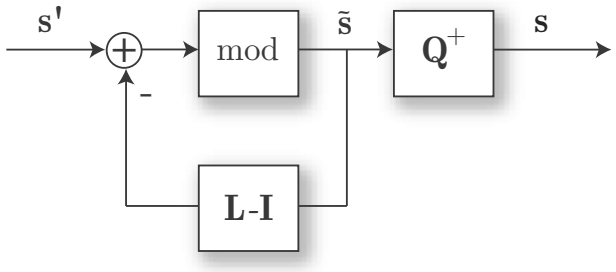


Figure 2.19. Block diagram of the Tomlinson-Harashima precoding.

can be carried out, for instance, using some of the tree-search algorithms described in Section 2.3. However, this search can be still computationally expensive, thus, other alternative approaches are usually employed to calculate the vector perturbation method or directly the signal to be transmitted.

2.5.3 Zero-Forcing Precoding

The simplest precoding methods are those where the transmitted vector $\mathbf{s}$ is the result of the product of a $2N \times 2K$ precoding matrix $\mathbf{P}$ by the vector $\mathbf{s}$. Similarly to linear detection, when the pseudoinverse of the channel matrix is selected as precoding matrix, the method is known as the zero-forcing (ZF) approach:

$$\mathbf{s} = \mathbf{H}^{+}\mathbf{s}' = \mathbf{H}^{H}(\mathbf{H}\mathbf{H}^{H})^{-1}\mathbf{s}'. \tag{2.43}$$

Note that vector $\mathbf{s}$ can be also obtained using the QR decomposition of matrix $\mathbf{H}$ to avoid computing $\mathbf{H}^{+}$ explicitly.

2.5.4 Tomlinson-Harashima Precoding

Tomlinson-Harashima precoding (THP) [59] can be interpreted as moving the feedback part of a DFE to the transmitter, as Fig. 2.19 shows. Since signals are perfectly known at the transmitter, errors do not propagate and multiuser interference is perfectly cancelled. The matrices that take part in the precoding process are obtained from the LQ decomposition of matrix

$\mathbf{H}$, which initially equals $\mathbf{H} = \mathbf{L}_0\mathbf{Q}_0$. A diagonal matrix $\mathbf{G}$ is introduced as follows:

$$\begin{aligned}\mathbf{H} = \mathbf{L}_0\mathbf{Q}_0 = (\mathbf{L}_0\mathbf{G}^{-1})(\mathbf{G}\mathbf{Q}_0) = \mathbf{L}\mathbf{Q},\\ \mathbf{Q}^+ = (\mathbf{G}\mathbf{Q}_0)^+ = \mathbf{Q}_0^T\mathbf{G}^{-1},\end{aligned} \tag{2.44}$$

where $\mathbf{L}$ is a $2K \times 2K$ lower triangular matrix with ones in its diagonal and $\mathbf{Q}_0$ contains orthogonal rows.

From the THP scheme in Fig. 2.19, it can be noted that the algorithm is a pseudo-linear equalization with matrix $\mathbf{L}$. Therefore, the precoded symbols $\hat{\mathbf{s}}$ can be initially expressed as

$$\hat{s}_i = s'_i - \sum_{l=1}^{k-1} L_{k,l}s_l, \qquad i = 1, \ldots, 2N. \tag{2.45}$$

Since this strategy increases the transmitted power significantly, a modulo operation is applied to restrict the symbols to the constellation $\hat{\Omega}$, which is a continuous constellation within a square. This reads:

$$\tilde{s}_i = \hat{s}_i \mathrm{mod} M = \hat{s}_i - M\left\lfloor \frac{\hat{s}_i + M/2}{M} \right\rfloor, \qquad i = 1, \ldots, 2N, \tag{2.46}$$

and thus the transmit power of this method is lower than with linear precoding. For a QPSK constellation, it is sufficient to add an integer multiple of 4 to the real and imaginary parts of the $\tilde{s}_i$ components to build $\hat{\Omega}$. For a general M-QAM constellation, integer multiples of $2\sqrt{M}$ should be added instead. This way, the original constellation is periodically extended and the vector to be precoded can be selected from the expanded constellation. Finally, $\mathbf{s} = \mathbf{Q}^+\tilde{\mathbf{s}}$ is transmitted over the channel.

2.5.5 Lattice-Reduction-Aided Precoding

The idea of preprocessing the channel matrix to improve data detection in MIMO systems can also be exploited to approximate the vector perturbation (2.42). In particular, the use of LR algorithms for signal precoding has been widely employed [16]. Determining the computational costs of these methods and trying to decrease them was one of the tasks carried out in

this thesis. For this reason, this kind of algorithms will be further detailed in a later chapter of this document together with the related contributions developed in this thesis.

2.6 Conclusion

In this chapter, the problem of ML detection in point-to-point MIMO (BLAST) systems was introduced. An overview of MIMO detection techniques was also presented. Detection methods were firstly classified according to their performance between optimal and suboptimal. In a second level, they were divided among linear, SIC and tree-search-based methods, paying special attention to those based on tree-search. In this context, the transformation of conventional ML detection into sphere decoding was addressed, showing that it is a promising approach for efficient MIMO detection.

Next, the extension of an uncoded MIMO system to a MIMO-BICM scheme was presented together with the motivation for MIMO soft demodulation. Some well-known soft demodulation schemes based on tree-search were cited and briefly introduced.

The last part of the chapter introduced the MU-MIMO scenario and system model, as a previous step towards describing multiuser signal precoding. The system model associated to a single subcarrier was shown to be equivalent to the MIMO-BLAST system model, showing the close relationship between both systems. Finally, some widely employed precoding techniques were described for the sake of completeness.

MIMO Preprocessing Techniques 3

3 MIMO Preprocessing Techniques

MIMO CHANNEL MATRIX PREPROCESSING techniques have been shown to be useful to either decrease the computational cost of optimal SD methods or to improve the performance of suboptimal linear, SIC or tree-search detectors. This chapter presents a detailed overview of two widely employed preprocessing techniques: the Lenstra, Lenstra, Lovasz (LLL) lattice-reduction (LR) algorithm and the VBLAST ZF-DFE ordering. Both the complexity and performance of these methods are evaluated and compared. In addition, a low-complexity implementation of the VBLAST ZF-DFE ordering is proposed and included within the evaluation.

3.1 Introduction

The complexity of optimal SD can be decreased if a preprocessing stage that transforms the MIMO channel matrix is included before the detection [32]. This means that the preprocessed channel matrix allows the tree-search to be completed after expanding less number of nodes than in the case with the original channel matrix. Among others, some of the most popular preprocessing techniques for MIMO detection are column ordering

techniques [32] and lattice-reduction (LR) techniques [29]. Most of the proposed column ordering techniques are based only on the MIMO channel matrix, and therefore, they require to be performed only when the channel changes (recall that when a block-fading channel is considered, it remains unchanged for a complete frame transmission). However, there exist other ordering strategies that also depend on the received vector and require a new channel matrix ordering for each received vector such as the gradient-based ordering [35] or the ordering proposed in [34].

As an alternative application, other authors have employed the above mentioned preprocessing techniques to improve the performance of suboptimal MIMO detectors [17][60]. In this case, the enhanced properties that the channel matrix enjoys after preprocessing help the detector to make a better decision on the solution.

This chapter presents an overview of two widely employed preprocessing techniques: the VBLAST ZF-DFE ordering [61] and the LLL LR algorithm [21]. The complexity of both methods is analyzed and a low-complexity implementation of the VBLAST ZF-DFE ordering is proposed. In addition, the performance of both methods to preprocess the channel matrix before using the K-Best tree-search detector is evaluated. The interest of this evaluation is twofold: it shows the detection performance improvements that can be achieved after preprocessing the channel matrix and also it allows to compare the detection performance achieved with both schemes. This comparison provides two criteria that can help to select the most suitable preprocessing technique for a final practical set-up, depending on the requirements and on the available hardware or software resources. Although the preprocessing techniques are tested with a particular detector, all the results and conclusions presented in this chapter can be directly exploited with other suboptimal detectors.

The chapter is structured as follows. Section 3.2 describes the VBLAST ZF-DFE ordering algorithm. A complexity and a performance evaluation of the method are introduced together with a proposed low-complexity implementation. In Section 3.3 the complexity and performance of two versions of the LLL method are presented. Finally, Section 3.4 is devoted to the results comparison between both preprocessing approaches and some

conclusions are presented in Section 3.5.

3.2 VBLAST ZF-DFE Ordering

As seen in Section 2.2.4, the preprocessing of the channel matrix is usually modelled by a transformation matrix $\mathbf{P}$. The steps to obtain the transformation matrix that models the VBLAST ZF-DFE ordering are described below.

3.2.1 Algorithm Description

Although many formulations for the VBLAST ZF-DFE ordering can be found in the literature [62][63], we will use in this work the one firstly proposed in [61] and suggested in [32] as an optimal ordering for SD-based detection.

The VBLAST ZF-DFE preprocessing and ordering algorithm intends to find the column ordering vector π that maximizes the minimum values in the diagonal of the $\mathbf{R}$ matrix (from the QR decomposition of $\mathbf{H}$) among all the possible column permutations. The transformation carried out by this algorithm can be expressed by means of a permutation matrix $\mathbf{\Pi}$ (i.e. $\mathbf{P} = \mathbf{\Pi}$), which holds $\mathbf{\Pi}\mathbf{\Pi}^{\mathrm{T}} = \mathbf{I}$, where $\mathbf{I}$ is an identity matrix.

The steps of the VBLAST ZF-DFE ordering are detailed in Algorithm 1. Note that $\mathbf{H}^{(kj)}$ denotes the matrix that results from selecting only the columns of the channel matrix $\mathbf{H}$ indexed by Δ'. Assuming the real-valued form of the system, the algorithm finds the optimal column ordering in $2n_T$ steps and stores it in vector π. The matrix $\mathbf{\Pi}$ can be easily built by ordering the columns of an identity matrix of size $2n_T$ according to the ordering contained in the vector of column indexes π.

Since this transformation does not modify the norm of the columns of the channel matrix, the components of the new vector to be detected, $\mathbf{z}$, belong to the same set as the components of the initial vector to be detected, $\mathbf{s}$. In fact, $\mathbf{z}$ is just a reordered version of $\mathbf{s}$ (since $\mathbf{P}^{-1} = \mathbf{P}^{T}$).

Algorithm 1 VBLAST ZF-DFE Ordering Algorithm

Input: $\mathbf{H}, n_T$
Output: π
1: $\Delta = \{1, \ldots, 2n_T\}$
2: **for** $k = 2n_T, \ldots, 2$ **do**
3: **for** $j = 1, \ldots, k$ **do**
4: $\Delta' = \Delta - \{\Delta_j\}$
5: $\mathbf{H}^{(kj)} = \mathbf{H}_{:,\Delta'}$
6: $p_j = \mathbf{H}_{:,j}^T[\mathbf{I} - \mathbf{H}^{(kj)}((\mathbf{H}^{(kj)})^T\mathbf{H}^{(kj)})^{-1}(\mathbf{H}^{(kj)})^T]\mathbf{H}_{:,j}$
7: **end for**
8: $\pi_k = \arg\max\{\mathbf{p}\}$
9: $\Delta = \Delta - \{\pi_k\}$
10: **end for**

3.2.2 Complexity Analysis

The number of floating point operations (flops) required for each run of the VBLAST ZF-DFE ordering algorithm can be determined in advance since it is fixed for a given channel matrix size. From Algorithm 1 it can be noted that step 6) contains the main contribution to the total complexity of the algorithm. The number of flops of this step, however, can vary depending on the way the operations are carried out therein.

At a first approach, let us describe the number of flops assuming a straight calculation of step 6). Note that they all depend on the value of k at each iteration.

- Product $\mathbf{A} = (\mathbf{H}^{(kj)})^T\mathbf{H}^{(kj)}$: $4n_T(k-1)^2$ flops.
- Inversion of matrix $\mathbf{A}$: $2(k-1)^3$ flops.
- Product $\mathbf{B} = \mathbf{H}^{(kj)}\mathbf{A}^{-1}$: $4n_T(k-1)^2$ flops.
- Product $\mathbf{C} = \mathbf{B}(\mathbf{H}^{(kj)})^T$: $8n_T^2(k-1)$ flops.
- Subtraction $\mathbf{D} = \mathbf{I} - \mathbf{C}$: $4n_T^2$ flops.
- Product $\mathbf{E} = \mathbf{H}_{:,j}^T\mathbf{D}$: $8n_T^2$ flops.
- Product $\mathbf{F} = \mathbf{E}\mathbf{H}_{:,j}$: $4n_T$ flops.

In addition, a search for the maximum value appears in step 8), which also depends on the k value. This operation exhibits a cost of k flops. Therefore, the total number of flops for each run of the algorithm can be calculated as:

$$\begin{aligned} C_{\mathrm{VB}} &= \sum_{k=2}^{2n_T} k\left[2(k-1)^3 + 8n_T(k-1)^2 + 8n_T^2(k-1) + 12n_T^2 + 4n_T\right] + k \\ &\cong \frac{992}{15} n_T^5 + \mathcal{O}(n_T^4). \end{aligned} \tag{3.1}$$

3.2.3 Low-Complexity Implementation

As seen in (3.1), the conventional implementation of the VBLAST ZF-DFE ordering has $\mathcal{O}(n_T^5)$ complexity, which is very high for practical implementations. In this thesis we focused on reducing this complexity and developed a more efficient QR-based approach. Recall that the new formulation of the VBLAST ZF-DFE method gets exactly the same solution as the original one with a much lower computational cost.

Making use of the QR factorization, the matrix $\mathbf{H}^{(kj)}$ is decomposed into the product $\mathbf{Q}^{(kj)}\mathbf{R}^{(kj)}$, where $\mathbf{Q}^{(kj)}$ is a $2n_T \times 2n_T$ orthogonal matrix and $\mathbf{R}^{(kj)} = [\mathbf{R}_1^T \quad \mathbf{0}]^T$ is a $2n_T \times (k-1)$ matrix, being $\mathbf{R}_1$ of size $(k-1) \times (k-1)$ and upper triangular.

If step 6) of Algorithm 1 is expressed as follows:

$$p_j = \|\mathbf{H}_{:,j}\|^2 - \mathbf{H}_{:,j}^T \mathbf{H}^{(kj)} ((\mathbf{H}^{(kj)})^T \mathbf{H}^{(kj)})^{-1} (\mathbf{H}^{(kj)})^T \mathbf{H}_{:,j}, \tag{3.2}$$

then, denoting as $\alpha = \mathbf{H}_{:,j}^T \mathbf{H}^{(kj)} ((\mathbf{H}^{(kj)})^T \mathbf{H}^{(kj)})^{-1} (\mathbf{H}^{(kj)})^T \mathbf{H}_{:,j}$, it can be shown that α can be computed more efficiently by using $\mathbf{Q}^{(kj)}$ and $\mathbf{R}^{(kj)}$ as:

$$\begin{aligned} \alpha &= \mathbf{H}_{:,j}^T (\mathbf{Q}^{(kj)} [\mathbf{R}_1^T \quad \mathbf{0}]^T ([\mathbf{R}_1^T \quad \mathbf{0}][\mathbf{R}_1^T \quad \mathbf{0}]^T)^{-1} [\mathbf{R}_1^T \quad \mathbf{0}] (\mathbf{Q}^{(kj)})^T) \mathbf{H}_{:,j} \\ &= \mathbf{H}_{:,j}^T \mathbf{Q}^{(kj)} \begin{bmatrix} \mathbf{I}_{(k-1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} (\mathbf{Q}^{(kj)})^T \mathbf{H}_{:,j} = \|\mathbf{w}_{1:(k-1)}\|^2, \end{aligned} \tag{3.3}$$

where $\mathbf{w} = (\mathbf{Q}^{(kj)})^T \mathbf{H}_{:,j}$.

Algorithm 2 Getting $\mathbf{w}$

Input: $\mathbf{Q}, \mathbf{R}$, j, k
Output: $\mathbf{w}$

1: $\mathbf{w} = \mathbf{R}_{:,j}$,
2: **for** $i = j+1, \ldots, k-1$ **do**
3: Calculate Givens Rotation $[c, s]$ for $R_{i-1,i}$ and $R_{i,i}$
4: **for** $l = i, \ldots, k-1$ **do**
5: $R_{i,l} = -sR_{i-1,l} + cR_{i,l}$
6: **end for**
7: $\Theta = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$
8: $\mathbf{w}_{i-1:i} = \Theta \mathbf{w}_{i-1:i}$
9: **end for**
10: $\mathbf{w} = \mathbf{w}_{1:k-1}$

Finally, p_j can be simply obtained as a subtraction of the norms of two vectors:

$$p_j = \|\mathbf{H}_{:,j}\|^2 - \|\mathbf{w}_{1:(k-1)}\|^2. \tag{3.4}$$

Note that the cost is further reduced if the equivalence $\|\mathbf{H}_{:,j}\|^2 = \|\mathbf{R}_{:,j}\|^2$ is taken into account.

The just described alternative formulation only affects some steps of Algorithm 1. First, the QR decomposition of $\mathbf{H}$ must be included among the inputs. Next, the norms of the columns of $\mathbf{H}$ are computed before step 2) and stored in a vector. This calculation requires $4n_T^2$ flops.

Once inside the inner *for* loop (step 3)), the vector $\mathbf{w}_{1:k-1}$ is obtained from the QR decomposition of $\mathbf{H}^{(kj)}$, which is just an update of another QR decomposition, as Algorithm 2 describes. In this updating process [64], the main goal is to compute the new $\mathbf{Q}^{(kj)}$ in order to get $\mathbf{w} = (\mathbf{Q}^{(kj)})^T \mathbf{H}_{:,j}$ subsequently. That is why matrix $\mathbf{R}$ is only partially updated, thus avoiding several flops.

Algorithm 3 shows how the product $(\mathbf{Q}^{(kj)})^T \mathbf{H}_{:,j}$ can be obtained by applying some Givens rotations to $\mathbf{R}_{:,j}$. Note that if $\mathbf{H} = \mathbf{QR}$, it is true that

$$\mathbf{H}^{(kj)} = \mathbf{Q}^{(kj)} \mathbf{R}^{(kj)} = (\mathbf{QG})(\mathbf{G}^T \mathbf{R}), \tag{3.5}$$

Algorithm 3 QR Update

Input: $\mathbf{Q}, \mathbf{R}$, π_k
Output: $\widetilde{\mathbf{Q}}, \widetilde{\mathbf{R}}$

1: **for** $i = \pi_k + 1, \ldots, k-1$ **do**
2: Calculate Givens Rotation $[c, s]$ for $R_{i-1,i}$ and $R_{i,i}$
3: $\Theta = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$
4: **for** $l = i, \ldots, k-1$ **do**
5: $[\ R_{i-1,l} \quad R_{i,l}\ ]^T = \Theta[\ R_{i-1,l} \quad R_{i,l}\ ]^T$
6: **end for**
7: **for** $l = 1, \ldots, 2n_T$ **do**
8: $[\ Q_{l,i-1} \quad Q_{l,i}\ ] = [\ Q_{l,i-1} \quad Q_{l,i}\ ]\Theta^T$
9: **end for**
10: **end for**
11: $\widetilde{\mathbf{Q}} = \mathbf{Q}$
12: $\widetilde{\mathbf{R}} = \mathbf{R}_{:,\{1:\pi_k-1,\pi_k+1:k-1\}}$

being $\mathbf{G}^T$ the product of Givens rotations (Θ matrices) which makes $\mathbf{Q}^T\mathbf{H}^{(kj)}$ be triangular. Thus, finally, $\mathbf{w}$ equals

$$\mathbf{w} = (\mathbf{Q}^{(kj)})^T\mathbf{H}_{:,j} = \mathbf{G}^T(\mathbf{Q}^T\mathbf{H}_{:,j}) = \mathbf{G}^T\mathbf{R}_{:,j}. \tag{3.6}$$

The cost to obtain $\mathbf{w}$, depending on k and j, equals $3(k-j)^2/2+13(k-j)$ flops.

As a result, the value of each p_j is calculated with (3.4) instead of (3.2). Thus, the total cost to compute all the p_j is $k^3/2 + 19k^2/2$ flops. Finally, a QR update routine is necessary to get the QR decomposition of the remaining columns of $\mathbf{H}$ after step 9) of Algorithm 1. The steps inside this QR update routine are included in Algorithm 3 and spend $3k^2+12kn_T+7k$ flops. From now on matrices $\widetilde{\mathbf{Q}}, \widetilde{\mathbf{R}}$ will represent the QR decomposition of the channel matrix after the preprocessing stage ($\widetilde{\mathbf{H}}$)

It must be again considered the search for the maximum (k flops). Finally, the total number of operations for each run of the QR-based VBLAST ZF-DFE ordering can be calculated as:

$$C_{\text{VB}}^{\text{QR}} = \sum_{k=2}^{2n_T} \left[\frac{k^3}{2} + \frac{25}{2}k^2 + 12kn_T + 8k \right] + 4n_T^2 \cong 2n_T^4 + \mathcal{O}(n_T^3). \quad (3.7)$$

As (3.7) shows, the proposed QR-based implementation reduces the complexity from $\mathcal{O}(n_T^5)$ to $\mathcal{O}(n_T^4)$.

3.2.4 Performance Evaluation

The peprocessing techniques under study were combined with the K-Best tree-search detector. Fig. 3.1 illustrates the BER performance versus SNR for the 3-Best and 5-Best detectors without preprocessing, compared to the performance of those algorithms when the VBLAST ZF-DFE preprocessing has been previously used, for a 4×4 MIMO system with a 16-QAM alphabet. Note that the VBLAST ZF-DFE preprocessing substantially improves the performance of the K-Best detector with different values of K for high values of the SNR. In fact, note that the combination of VBLAST ZF-DFE with the 5-Best detector achieves almost ML performance.

3.3 Lattice-Reduction Algorithms

Any lattice can be generated from many possible bases [16]. Given a basis $\mathbf{B}$, many other generator bases can be constructed as $\mathbf{B}' = \mathbf{BP}$ for any matrix $\mathbf{P}$ such that both $\mathbf{P}$ and $\mathbf{P}^{-1}$ have integer entries. If the columns of the channel matrix $\mathbf{H}$ are considered to be the basis of a lattice, LR strategies [21] can be used to preprocess and transform the channel matrix $\mathbf{H}$ into a new channel matrix $\widetilde{\mathbf{H}} = \mathbf{HT}$ with less correlated columns [17]. In this case, the transformation matrix $\mathbf{T}$ is unimodular (i.e. $\det(\mathbf{T}) = \pm 1$).

The basic idea behind using LR prior to traditional low-complexity detectors is to operate in a chosen lattice basis that is optimized for those detectors, as shown in Fig. 3.2. Note that transmitting vector $\mathbf{s}$ through the channel $\mathbf{H}$ is equivalent to transmitting $\mathbf{z} = \mathbf{T}^{-1}\mathbf{s}$ through $\widetilde{\mathbf{H}}$. At the

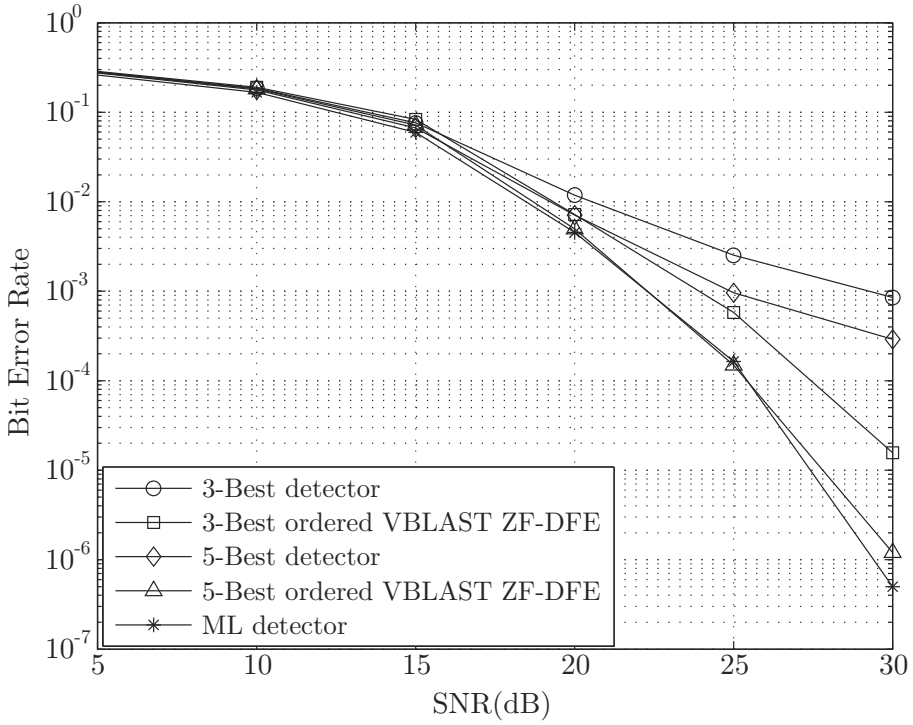


Figure 3.1. BER curves of the K-Best detector with two different values of K (3 and 5) in a 4×4 MIMO system using 16-QAM, both compared to the same algorithms after VBLAST ZF-DFE preprocessing.

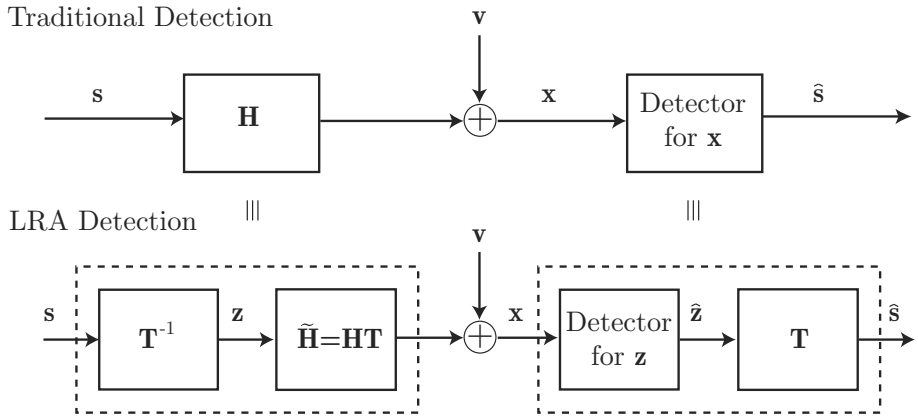


Figure 3.2. Comparison between traditional detection and LR-aided detection

receiver, the detector must operate over $\mathbf{z}$ and multiply the result by $\mathbf{T}$ to give the detected vector $\hat{\mathbf{s}}$.

Among many LR algorithms [29], in this thesis we focused mainly on the most employed one in the context of MIMO communications systems: the LLL method [21]. A brief description of Seysen's algorithm [65] is also included.

3.3.1 LLL and Seysen's Algorithms

The LLL method is a meaningful LR algorithm which was firstly proposed in [21]. This method iterates between a weak Gram-Schmidt decomposition and column-swapping in order to achieve more orthogonal columns of the final LLL-reduced channel matrix. Although this method is suboptimal, it has been shown to give acceptable results when used previously to a linear detection [16][17][18] or previously to a tree search detection [60]. The LLL method starts from the QR decomposition of matrix $\mathbf{H}$ and size-reduces matrix $\mathbf{R}$ when the following condition is not satisfied:

$$|R_{l,k}| \leq \frac{1}{2}|R_{l,l}|, \tag{3.8}$$

Algorithm 4 LLL Algorithm

Input: $\mathbf{Q}, \mathbf{R}$, δ
Output: $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{R}}$, $\mathbf{T}$

$\widetilde{\mathbf{Q}} = \mathbf{Q}$, $\widetilde{\mathbf{R}} = \mathbf{R}$, $\mathbf{T} = \mathbf{I}_{2n_T}$, $m = 2$
while $m \leq 2n_T$ **do**
  for $l = m-1, \ldots, 1$ **do**
    $\lambda = \text{round}(\widetilde{R}_{l,m}/\widetilde{R}_{l,l})$
    if $\lambda \neq 0$ **then**
      $\widetilde{R}_{1:l,m} = \widetilde{R}_{1:l,m} - \lambda \widetilde{R}_{1:l,l}$
      $T_{:,m} = T_{:,m} - \lambda T_{:,l}$
    end if
  end for
  if $\delta \widetilde{R}^2_{m-1,m-1} > \widetilde{R}^2_{m,m} + \widetilde{R}^2_{m-1,m}$ **then**
    $\widetilde{R}_{:,m-1} \leftrightarrow \widetilde{R}_{:,m}$
    $T_{:,m-1} \leftrightarrow T_{:,m}$
    $\Theta = \begin{bmatrix} \widetilde{R}_{m-1,m-1} & \widetilde{R}_{m,m-1} \\ -\widetilde{R}_{m,m-1} & \widetilde{R}_{m-1,m-1} \end{bmatrix}$
    $\Theta = \Theta / \|\widetilde{R}_{m-1:m,m-1}\|$
    $\widetilde{R}_{m-1:m,m-1:2n_T} = \Theta \widetilde{R}_{m-1:m,m-1:2n_T}$
    $\widetilde{Q}_{:,m-1:m} = \widetilde{Q}_{:,m-1:m} \Theta^T$
    $m = \max\{m-1, 2\}$
  else
    $m = m + 1$
  end if
end while

where $1 \leq l < k \leq 2n_T$. In addition, two column vectors are exchanged if (3.9) is not satisfied for a given δ parameter:

$$\delta R^2_{k-1,k-1} \leq R^2_{k,k} + R^2_{k-1,k}, \tag{3.9}$$

with $1/4 < \delta \leq 1$ and for $k = 2, \ldots, 2n_T$. A typical value for δ is $3/4$ [21], which will be considered in this thesis.

In order to calculate the computational cost of the algorithm, it is useful to observe the steps of the LLL algorithm, which are detailed in Algorithm 4.

Algorithm 5 Fixed-Complexity LLL Algorithm

Input: $\mathbf{Q}, \mathbf{R}, \delta, Y$
Output: $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{R}}$, $\mathbf{T}$

1: $\widetilde{\mathbf{Q}} = \mathbf{Q}$, $\widetilde{\mathbf{R}} = \mathbf{R}$, $\mathbf{T} = \mathbf{I}_{2n_T}$, $m = 2$
2: **for** $y = 1, \ldots, Y$ **do**
3: **while** $m \leq 2n_T$ **do**
4: **for** $l = m-1, \ldots, 1$ **do**
5: $\lambda = \text{round}(\widetilde{R}_{l,m}/\widetilde{R}_{l,l})$
6: **if** $\lambda \neq 0$ **then**
7: $\widetilde{R}_{1:l,m} = \widetilde{R}_{1:l,m} - \lambda \widetilde{R}_{1:l,l}$
8: $T_{:,m} = T_{:,m} - \lambda T_{:,l}$
9: **end if**
10: **end for**
11: **if** $\delta \widetilde{R}^2_{m-1,m-1} > \widetilde{R}^2_{m,m} + \widetilde{R}^2_{m-1,m}$ **then**
12: $\widetilde{R}_{:,m-1} \leftrightarrow \widetilde{R}_{:,m}$
13: $T_{:,m-1} \leftrightarrow T_{:,m}$
14: QR update
15: **end if**
16: $m = m + 1$
17: **end while**
18: **end for**

One of the main drawbacks of the LLL algorithm is that its number of operations cannot be easily determined in advance, since the value of m in the *while* loop that begins in step 2) of Algorithm 4 is not always incremented, but it can be either incremented or decremented depending on the condition in step 10). This is due to the *if-else* sentence contained in steps 10) to 20).

In [66], a modified LLL algorithm that can eventually guarantee a maximum number of loops inside the LLL algorithm was proposed (and thus called fixed complexity LLL (fcLLL)). Since this algorithm can be more suitable for real-time applications, we included it in our study of performance and complexity. The steps of the fcLLL are included in Algorithm 5.

Basically, the fcLLL algorithm removes step 17) from Algorithm 4, which is the main responsible for the variable complexity, and includes an outer *for* loop that fixes the number of times that the steps from 3) to

17) are carried out, which is called LLL loop. The number of LLL loops, L, can be set by means of an input value, Y, that ensures a number of $L = Y(2n_T - 1)$ loops.

Another LR method of recent interest is Seysen's algorithm [65]. This method reduces not only the lattice basis $\mathbf{H}$ but also its dual lattice basis $\mathbf{H}^{\#}$ at the same time. The algorithm aims to minimize the Seysen's orthogonality measure defined as

$$S(\widetilde{\mathbf{H}}) = \sum_{i=1}^{n_T} \|\mathbf{H}_{:,i}\|^2 \|\mathbf{H}_{:,i}^{\#}\|^2, \tag{3.10}$$

where $\mathbf{H}_{:,i}$ denotes the ith column of the channel matrix $\mathbf{H}$. Further details concerning SA can be found in [65] and its performance with linear detectors in several works [23].

3.3.2 Complexity Analysis

As said above, the total number of operations required by the LLL algorithm cannot be easily determined in advance due to the condition in step 10). Moreover, the fulfillment of the condition in step 5) can also vary the number of operations.

Therefore, as a first approach, the number of operations inside both conditions are obtained as follows:

a) Inside the condition in step 5), from now on called *condition 1*, a maximum number of $4n_T$ flops are required for the linear combination of two columns.

b) Inside the condition in step 10), in the sequel called *condition 2*, a maximum number of $24n_T$ flops is needed for the QR update of a matrix of maximum size $2n_T \times 2n_T$. Actually, $12n_T$ flops are necessary to update matrix $\mathbf{R}$ and other $12n_T$ flops to update matrix $\mathbf{Q}$. Note that matrix Θ is a 2×2 matrix which only modifies two columns of matrix $\mathbf{Q}$ when it post-multiplies it. In the same way, Θ modifies only two rows of matrix $\mathbf{R}$ when it pre-multiplies it.

Next, the number of times that each condition is satisfied was approximated in order to compute the overall complexity. From now on, the num-

Table 3.1. Number of times that the conditions inside the LLL algorithm are fulfilled.

	γ_1(LLL)		γ_2(LLL)	
Complex MIMO	Avg	Max	Avg	Max
2×2	3.53	23	2.41	16
3×3	9.97	47	6.97	33
4×4	19.16	91	13.17	68
6×6	46.62	214	27.88	131
8×8	85.05	410	42.76	205

ber of times that condition 1 and condition 2 are fulfilled, which depend on the current channel realization, are denoted as γ_1 and γ_2, respectively. Considering this, the computational cost of the LLL algorithm for a given channel can be calculated as:

$$C_{\text{LLL}} \cong \gamma_1 \cdot 4n_T + \gamma_2 \cdot 24n_T. \tag{3.11}$$

To approximate γ_1 and γ_2, a wide range of MIMO channel matrices were generated (10^6 matrices having i.i.d. entries $\sim \mathcal{CN}(0,1)$) to be processed by the LLL method. The number of times conditions 1 and 2 were fulfilled was recorded and the maximum and average value of those parameters were extracted. These values for different channel matrix sizes are collected in Table 3.1.

As it was shown in [66], the fcLLL algorithm has marginal loss of performance in comparison to the LLL when a value of $Y = 5$ is considered, for a 4×4 complex MIMO system. Therefore, this value of Y was selected for our complexity evaluation. The same value of Y will be assumed for all the system sizes as an approximation. Further work includes a more extensive analysis to determine the most convenient value of Y depending on the channel matrix size, since the way the Y parameter affects the complexity for higher systems is not investigated in [66].

Table 3.2. Number of times that the conditions inside the fixed complexity LLL algorithm with $Y = 5$ are fulfilled.

	γ_1(fcLLL)		γ_2(fcLLL)	
Complex MIMO	Avg	Max	Avg	Max
2×2	3.50	19	2.40	13
3×3	10.58	41	6.85	23
4×4	21.12	74	12.13	31
6×6	49.97	132	21.62	46
8×8	86.81	210	29.95	57

The fcLLL is claimed to have fixed complexity [66], since this algorithm restricts the maximum number of executed LLL loops. However, inside each LLL loop, the fulfillments of conditions 1 and 2 are dependent on the channel realization to be processed, as in the case of the original implementation of the LLL method. Therefore, it could be said that the complexity of the method is somewhat bounded but it is not always exactly the same. In the end, the computational cost of the method is given again by (3.11).

Table 3.2 presents the average and maximum γ_1 and γ_2 of the fcLLL algorithm for $Y = 5$, using this Y value for several channel sizes.

Note that computer simulations could provide the average complexity of both the LLL and fcLLL algorithms directly. However, the reason for including (3.11) was to emphasize which contributions to the average complexity of the LLL algorithm are predictable (number of operations inside each condition), and which ones are dependent on the current channel realization (number of times that each condition is fulfilled).

3.3.3 Performance Evaluation

The use of channel matrix ordering techniques such as the VBLAST ZF-DFE to improve MIMO detection are easy to combine with any detector

just keeping in mind to reorder the detected components to undo the effects of the preprocessing. However, there are other preprocessing techniques, such as the LR, the use of which with tree-search detectors is not straightforward and requires extra computational cost. Nevertheless, in a later section we will propose an approach to allow the use of LR with the K-Best detector, which was employed in this thesis for all the performance evaluations involving lattice-reduction-aided (LRA) K-Best detection.

Fig. 3.3 illustrates the performance of the 3-Best and 5-Best detectors compared to the performance of those algorithms aided by LLL preprocessing. As in the case with the VBLAST ZF-DFE preprocessing, the LLL can improve the performance of the K-Best detector, but this time it is only worth for high values of the SNR. In fact, for SNR values lower than 22 dB, the LLL preprocessing can even slightly worsen the performance of the K-Best.

3.4 Complexity and Performance Comparison

The complexities, in number of flops, of the VBLAST ZF-DFE, LLL and fcLLL preprocessing algorithms for different dimensions of the complex MIMO system are displayed in Fig. 3.4. For the VBLAST ZF-DFE method, the efficient implementation proposed in this thesis has been considered. Also note that for both LLL algorithms, the maximum and average complexities have been calculated, considering as values for the number of times that the conditions inside the algorithm are fulfilled those from Tables 3.1 and 3.2. As Fig. 3.4 shows, the proposed low-complexity implementation of the VBLAST ZF-DFE preprocessing requires a higher number of flops than the LLL and fcLLL with $Y = 5$ for the average case. In fact, the fcLLL remains less costly than the VBLAST ZF-DFE even for the worst-case (maximum number of flops). Therefore, the LLL method is the algorithm with the highest worst-case computational cost.

In Fig. 3.5 it is depicted the comparison between the BER curves of the K-Best detector with $K = 3$ and the performance of the same algorithm when either the LLL or the VBLAST ZF-DFE preprocessing algorithms

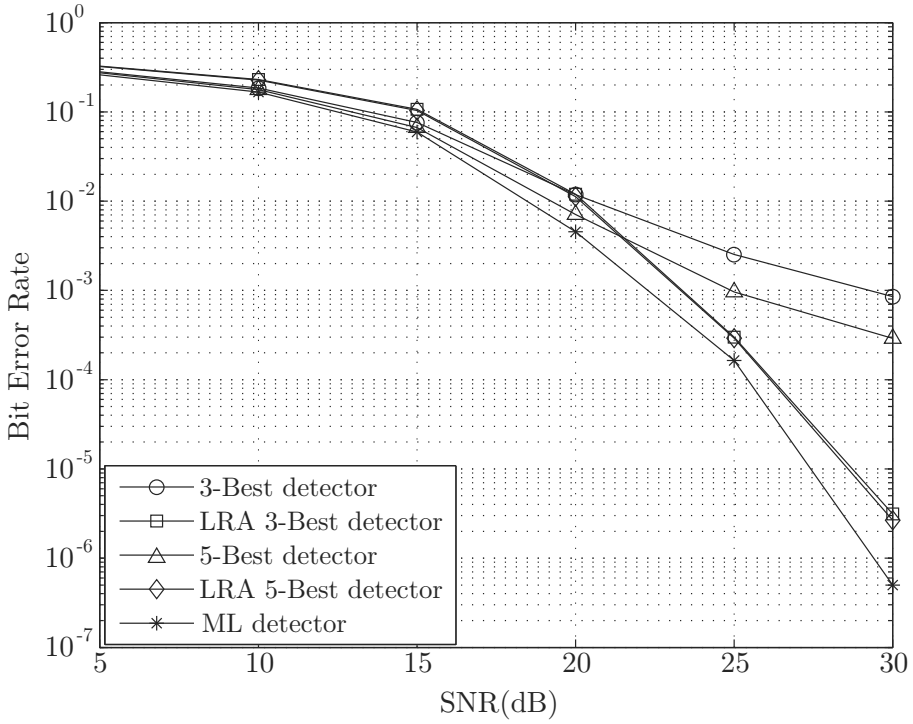


Figure 3.3. BER curves of the K-Best detector with two different values of K (3 and 5) in a 4×4 MIMO system using 16-QAM, both compared to the same algorithms when the LLL preprocessing is applied.

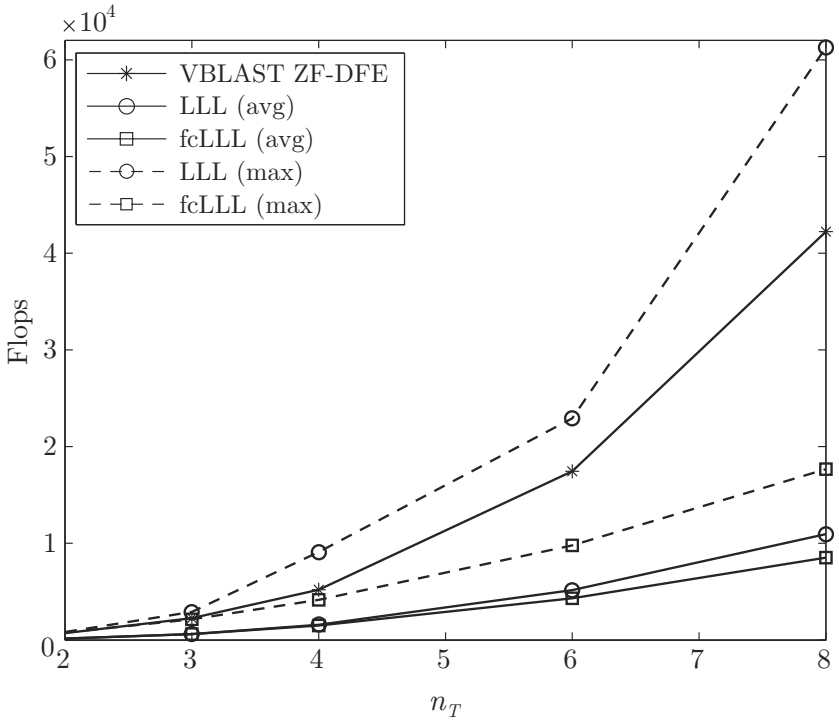


Figure 3.4. Number of flops for the VBLAST ZF-DFE, LLL and fixed complexity LLL with $Y = 5$ methods.

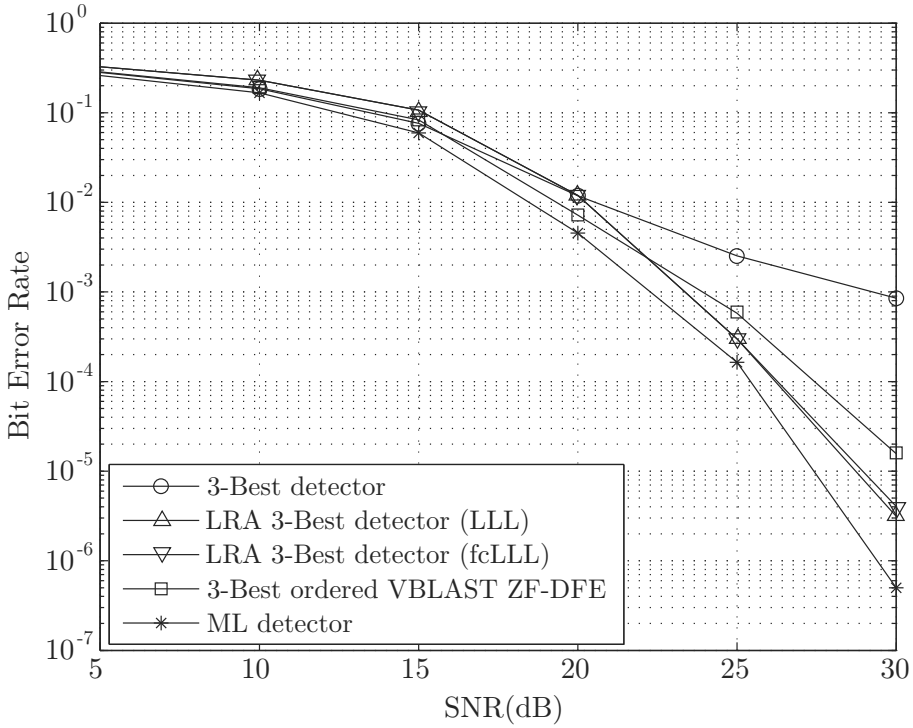


Figure 3.5. BER curves of the K-Best detector with $K = 3$ without preprocessing and when the LLL, fcLLL and VBLAST-DFE preprocessing algorithms are applied, in a 4×4 MIMO system using 16-QAM.

are applied. The BER curve of the fcLLL method with $Y = 5$ has also been displayed and shows that this method has similar performance to the LLL method. In this case, the VBLAST ZF-DFE preprocessing achieves slightly better detection results than the LLL method only for SNR $<$ 22 dB. However, for the 5-Best detector, Figs. 3.1 and 3.3 reveal that the VBLAST ZF-DFE performs better for every SNR. After observing that a higher average number of operations is needed to run the VBLAST ZF-DFE ordering than to run the LLL algorithm, providing a closed rule to choose between both algorithms is not possible. In the end, the most suitable preprocessing should be chosen according to the application needs, and the SNR and available system resources.

3.5 Conclusion

This chapter presented a performance and a complexity study of the application of preprocessing techniques to the K-Best tree search MIMO detector. Both, a complexity and a performance comparison between two LR algorithms, the LLL and fcLLL, and a column ordering strategy based on the channel matrix, the VBLAST ZF-DFE, were performed. Also, an efficient QR-based implementation of the latter was proposed to reduce the complexity from $\mathcal{O}(n_T^5)$ to $\mathcal{O}(n_T^4)$. It has been shown that the VBLAST ZF-DFE ordering exhibits fixed complexity for a given MIMO channel size, which can be more advisable for practical implementations that require predictable complexity, whereas the LLL algorithms have variable overall complexity, depending both on the size and values of the channel matrix entries. Therefore, our comparison included the average and maximum number of flops for the LLL algorithms, which were partially estimated by simulations.

The results showed that the VBLAST ZF-DFE preprocessing algorithm requires a higher number of operations than the fcLLL and LLL algorithms for the average case. If the worst-case is considered, however, the complexity of VBLAST ZF-DFE is in between the fcLLL and the LLL, being the LLL the most costly algorithm. On the other hand, for high SNR, al-

though the LLL preprocessing achieves the best detection performance for the 3-Best detector, the VBLAST ZF-DFE outperforms the LLL for the 5-Best case. Therefore, the most suitable preprocessing should be chosen according to the application needs and the available resources.

The main contributions of this chapter were published in [67].

4 Efficient Hard-Output Detection

4 Efficient Hard-Output Detection

Previous works have shown that the performance of MIMO detectors is highly influenced by the MIMO channel matrix condition number. In this chapter, the impact of the 2-norm channel matrix condition number in data detection is exploited in order to decrease the complexity of already proposed detection schemes. First, a variable-breadth K-Best detector is developed, where the value of its K parameter is varied depending on the channel matrix condition number. The proposed approach includes a low-complexity condition number estimator stage and a threshold selection method. The results show that the proposed scheme has lower average complexity than a fixed-breadth K-Best detector of similar performance. In addition, a second detection scheme to be used with LR techniques is proposed, which employs the idea of condition number thresholding to avoid carrying out a LR stage when the channel has already good condition number. This way, a high number of LR calls is avoided while keeping good detection performance.

4.1 Introduction

Experiments in [68] show that the channel matrix condition number is strongly related to the performance of suboptimal detection schemes, since it is a measure of how the symbols belonging to the original constellation get distorted by the channel. It can also be shown that the condition number generally increases with the size of the channel matrix [69], thus, MIMO systems with a high number of antennas can suffer from a stronger detection degradation.

The influence of the condition number over data detection motivates the design of MIMO detection schemes based on this parameter. Authors of [70] developed MIMO detectors based on condition number thresholding, for instance, combining the use of ML and ZF detectors. However, the use of the ML detector either by exhaustive search or by optimal sphere decoding leads to a generally non-fixed complexity in the resulting detector structure. Keeping in mind the potential practical implementation of the detector, these approaches require two or more different algorithms to be implemented, which can be an additional drawback.

In this chapter, the impact of the 2-norm condition number in data detection is exploited in order to either improve the performance of already proposed algorithms or to decrease their complexity while keeping acceptable performance. Two different efficient detection schemes based on condition number are described. The first of them is exclusively based on the K-Best tree-search detector while the second one combines the K-Best detector with the use of LR.

The chapter is organized as follows. Section 4.2 describes the proposed variable-breadth K-Best detection method. In Section 4.3, low-complexity channel matrix condition number estimation is described and the details concerning threshold condition number selection are pointed out in Section 4.4. Finally, Section 4.6 addresses some results of the variable-breadth K-Best approach and Section 4.5 contains the main aspects of the proposed LRA K-Best detector based on condition number.

4.2 Variable-Breadth K-Best Detector

The sensitivity of the solution of a non-singular system of linear equations $\mathbf{Ax} = \mathbf{b}$ with respect to perturbations of the matrix $\mathbf{A}$ is directly related to its condition number [64], which is defined as

$$\kappa_p(\mathbf{A}) = \|\mathbf{A}\|_p \|\mathbf{A}^{-1}\|_p. \tag{4.1}$$

Note that $\kappa_p(\mathbf{A})$ is a function of the norm (denoted by $\|\cdot\|_p$), being norms 1, 2 and ∞ generally the only used in practice. The variation of $\kappa_p(\mathbf{A})$ with p can be somewhat predicted, since on a finite dimensional vector space, all norms are related [71]. For instance, the 1 and 2-norms are related in $\mathbb{R}^n$ and the relationship among their respective condition numbers is given by

$$\frac{1}{n}\kappa_2(\mathbf{A}) \leq \kappa_1(\mathbf{A}) \leq n\kappa_2(\mathbf{A}), \tag{4.2}$$

where the size of matrix $\mathbf{A}$ is assumed as $n \times n$.

Among the three condition numbers mentioned above, $\kappa_2(\mathbf{A})$ was selected for the design of the algorithms presented in this chapter. This was motivated by some special properties of this condition number, which will be introduced in a later section of this chapter. For the sake of simplicity, in the sequel κ will denote κ_2 and $\|\cdot\|_2$ the 2-norm.

It is useful to mention that the 2-norm condition number of a matrix $\mathbf{A}$ can be equivalently calculated as [64]

$$\kappa(\mathbf{A}) = \frac{\sigma_{max}}{\sigma_{min}}, \tag{4.3}$$

being σ_{max} and σ_{min} the maximum and minimum singular values of $\mathbf{A}$, respectively. According to (4.3), the condition number ranges between 1 and ∞. A matrix with a low condition number (close to 1) is said to be *well-conditioned*, while a matrix with a high condition number is said to be *ill-conditioned*.

The probability density function (pdf) of κ for a Gaussian MIMO channel matrix with $n_T = n_R$ is given by [72]:

$$f_\kappa(\kappa) = \frac{8{n_T}^3}{\kappa^3} \exp\left(-\frac{4{n_T}^2}{\kappa^2}\right). \tag{4.4}$$

Note that since (4.4) decays polynomially, the probability of κ taking high values cannot be neglected. As many experiments show, these high values often worsen detectors performance.

This fact can also be observed in Fig. 4.1, which shows the degradation performance of some MIMO detectors with the channel matrix condition number for SNR= 20 dB. The selected algorithms were the optimal ML detector and the K-Best tree-search detector with two different values of K (5 and 8). On the one hand, it can be seen that, for channel matrix condition number values lower than 8, the performance of the 5-Best is almost equal to the one of the 8-Best or even to the ML. On the other hand, as the condition number gets higher, the performance of 8-Best is not as much degraded as the 5-Best. Considering this, it seems reasonable to think that a suitable combination between 5-Best and 8-Best based on the channel condition number could have a quasi-ML behavior for this value of SNR. This is the idea behind the detection scheme developed in this chapter.

The proposed approach performs always a K-Best detection but with the possibility of tuning the breadth of the detection tree (given by K) among a set of possible values. The most suitable K value is chosen depending on the channel matrix condition number. This method was called variable-breadth K-Best detector (VB K-Best).

Fig. 4.2 shows the flow diagram of the implementation of the VB K-Best detector with only two considered condition number regions (i.e. setting only one threshold). Note that a low value of K (K_1) is used while working with well-conditioned channels and a higher value of K (K_2) when the channel is ill-conditioned. This way, a greater number of tree paths is considered in the K-Best detector, making the ML solution less likely to be discarded in the second case.

Note that, although the proposed scheme adapts its complexity depending on the current channel matrix, in block fading channels the channel is assumed constant during a whole block transmission. Thus, the new condition number is only computed once per block transmission and the system observes most of the time fixed detection complexity. Moreover, when K is

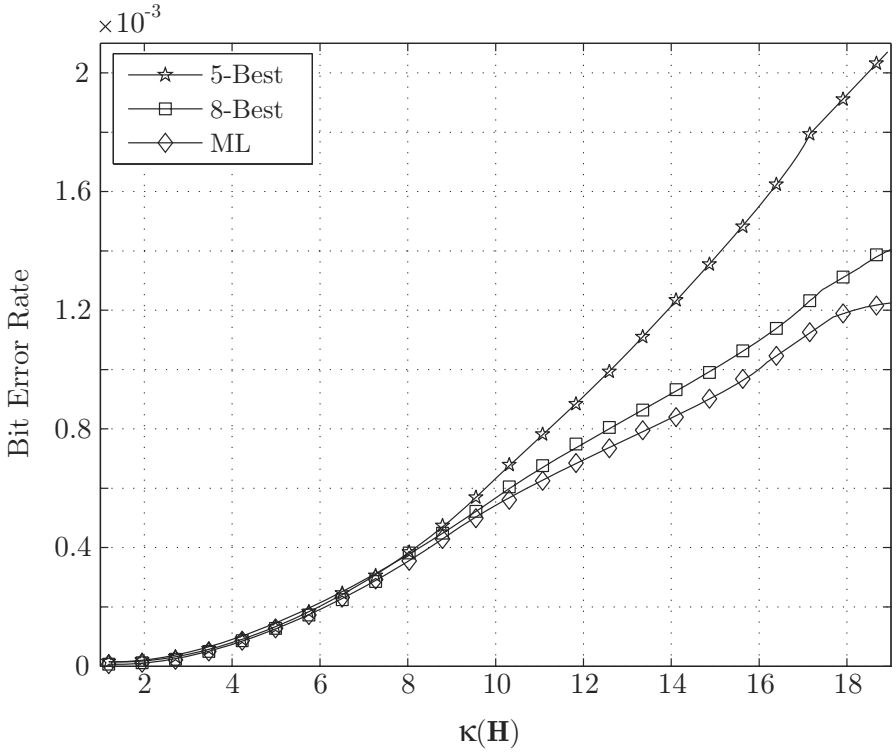


Figure 4.1. BER of different MIMO detectors with a 16-QAM constellation in a 4×4 MIMO channel and a SNR of 20 dB, as a function of the channel matrix condition number $\kappa(\mathbf{H})$.

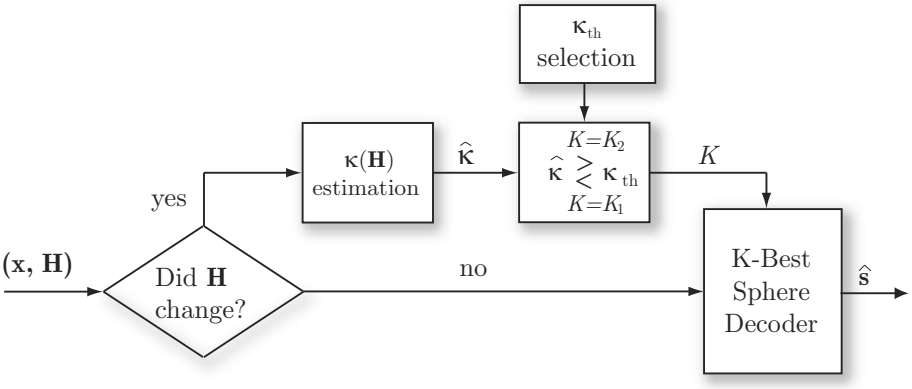


Figure 4.2. Flow diagram of the VB K-Best detector with threshold selection and channel matrix condition number estimation.

changed, the complexity switches to another value that is also fixed, thus the main implementation advantage of the K-Best detector is conserved.

Recall that, as said before, an estimator of the condition number together with a threshold condition number, denoted by κ_{th}, are needed to classify the channels and subsequently adjust the K parameter. The implementation details of this auxiliary stages are described next.

4.3 Channel Matrix Condition Number Estimator

Focusing on the practical implementation of the proposed approach, it is essential to have a reliable and low cost estimator of $\kappa(\mathbf{H})$. As explained in Section 2.3, tree-search detectors (without being the K-Best an exception) require a factorization of the form $\mathbf{H} = \mathbf{QR}$. Making use of (4.1) and only in case of working with the 2-norm, the following equality holds

$$\kappa(\mathbf{H}) = \kappa(\mathbf{R}) = \|\mathbf{R}\|\|\mathbf{R}^{-1}\|, \tag{4.5}$$

since $\mathbf{Q}$ is a unitary matrix with $\|\mathbf{Q}\| = 1$. Note that $\kappa(\mathbf{R})$ can be calculated faster than $\kappa(\mathbf{H})$, due to the fact that $\mathbf{R}$ is triangular.

The proposed condition number estimator is composed by the product of two independent estimators that calculate σ_{max} and $1/\sigma_{min}$, respectively. First, to efficiently compute $\|\mathbf{R}\| = \sigma_{max}$, the *Power Method* (PM) [64] is employed. Then, to compute $\|\mathbf{R}^{-1}\| = 1/\sigma_{min}$, a low complexity method, which was firstly developed in [73], is used. In what follows, each independent estimator and the joint proposed one are described.

4.3.1 The Power Method to Estimate σ_{max}.

The Power Method (PM) is an iterative algorithm that obtains the eigenvalue with the largest absolute value of a given $n \times n$ diagonalizable matrix $\mathbf{A}$. The method starts with a unit 2-norm vector $\mathbf{q}^{(0)} \in \mathbb{R}^n$ as an initial approximation of one of the dominant eigenvectors. At each iteration i, it updates $\mathbf{q}^{(i)}$ in two steps. First, an auxiliary vector $\mathbf{z}^{(i)}$ is calculated from vector $\mathbf{q}$ in the previous iteration and, after this, $\mathbf{z}^{(i)}$ is normalized:

$$\mathbf{z}^{(i)} = \mathbf{A}\mathbf{q}^{(i-1)}, \tag{4.6}$$

$$\mathbf{q}^{(i)} = \mathbf{z}^{(i)}/\|\mathbf{z}^{(i)}\|. \tag{4.7}$$

After the last iteration of the process, the maximum eigenvalue of $\mathbf{A}$ can be computed as

$$\lambda_{max} = [\mathbf{q}^{(i)}]^{\mathrm{T}}\mathbf{A}\mathbf{q}^{(i)}. \tag{4.8}$$

In our work, the PM is proposed for calculating $\|\mathbf{R}\| = \sigma_{max}$. To get this, the maximum eigenvalue of $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ is obtained, which corresponds to σ^2_{max}. Obviously, the method can be applied to $\mathbf{R}^T\mathbf{R}$ without computing explicitly this product.

Considering that the maximum size of the real channel matrices in practical MIMO systems is usually up to 16×16, a maximum number of 10 iterations for running the PM gets quite accurate results. Thus, the number of flops of the PM for a square matrix of size n equals $21n^2 + 22n$.

4.3.2 Estimator of σ_{min}^{-1}.

This method was firstly developed in [73] and is based on solving two triangular systems. The first system to be solved is

$$\mathbf{R}^T\mathbf{x} = \mathbf{b}, \tag{4.9}$$

where the right-hand-side $\mathbf{b}$ has to be chosen so that the solution of the system, $\hat{\mathbf{x}}$, makes $\|\hat{\mathbf{x}}\|/\|\mathbf{b}\|$ as large as possible. This is achieved after a n-step iterative process, considering the size of $\mathbf{R}^{-1}$ is n. At each step i, the component b_i is chosen between $+1$ and -1 in order to maximize $\hat{x}_i$, which will be computed as

$$\hat{x}_i = \frac{b_i - (R_{1i}\hat{x}_i + \cdots + R_{i-1i}\hat{x}_{i-1})}{R_{ii}}. \tag{4.10}$$

The second system to solve is

$$\mathbf{R}\mathbf{y} = \hat{\mathbf{x}}. \tag{4.11}$$

Once its solution $\hat{\mathbf{y}}$ is obtained, the estimation is given by

$$\|\mathbf{R}^{-1}\| = \frac{1}{\sigma_{min}} = \frac{\|\hat{\mathbf{y}}\|}{\|\hat{\mathbf{x}}\|}. \tag{4.12}$$

The number of flops of this estimator is only $2n^2 + 6n$.

4.3.3 Joint Estimator of $\kappa(\mathbf{H})$.

As soon as σ_{max} and $1/\sigma_{min}$ are computed via the estimators described above, $\kappa(\mathbf{R})$ is simply calculated as the product of both values.

In [70] the authors estimated both σ_{max} and $1/\sigma_{min}$ by means of the PM, what led to $42n^2+44n+2$ total flops. On the other hand, the proposed combined estimator exploits better the QR factorization used in the K-Best algorithms thus requiring only $23n^2 + 28n + 2$ flops.

Table 4.1 shows that the complexity of the proposed estimator of $\kappa(\mathbf{R})$ measured in number of flops is almost half the complexity of the estimator of $\kappa(\mathbf{R})$ that employs exclusively the PM to compute both singular values.

Table 4.1. Complexity in number of flops for the $\kappa(\mathbf{R})$ estimators of Gaussian MIMO channel matrices.

Real-valued channel	Proposed estimator	Power Method
4×4	482	850
8×8	1698	3042
16×16	6338	11458

Focusing not only on the complexity but also on the accuracy of the estimators, the relative estimation error and the magnitude error were calculated. The relative estimation error was obtained as

$$E_R = \frac{|\hat{\kappa}(\mathbf{H}) - \kappa(\mathbf{H})|}{\kappa(\mathbf{H})} \times 100 \tag{4.13}$$

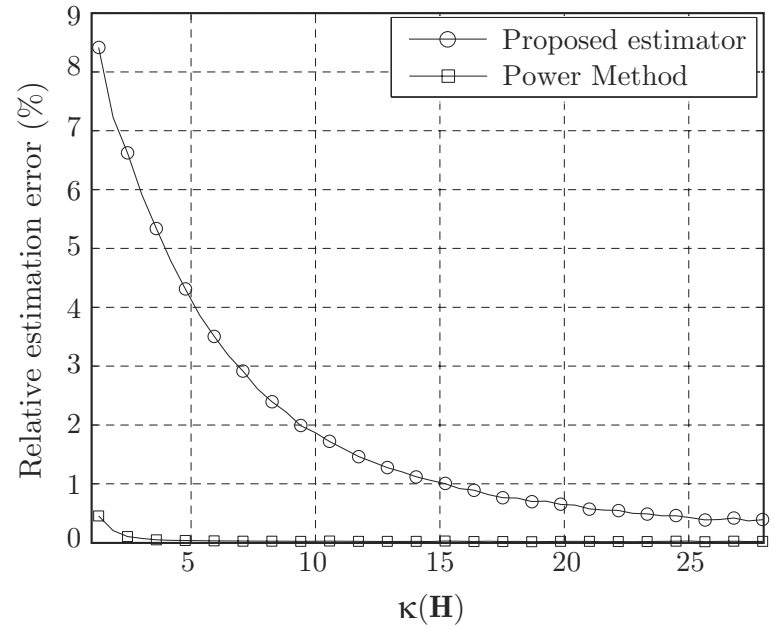
and the magnitude error was simply measured as

$$E_M = |\hat{\kappa}(\mathbf{H}) - \kappa(\mathbf{H})|. \tag{4.14}$$

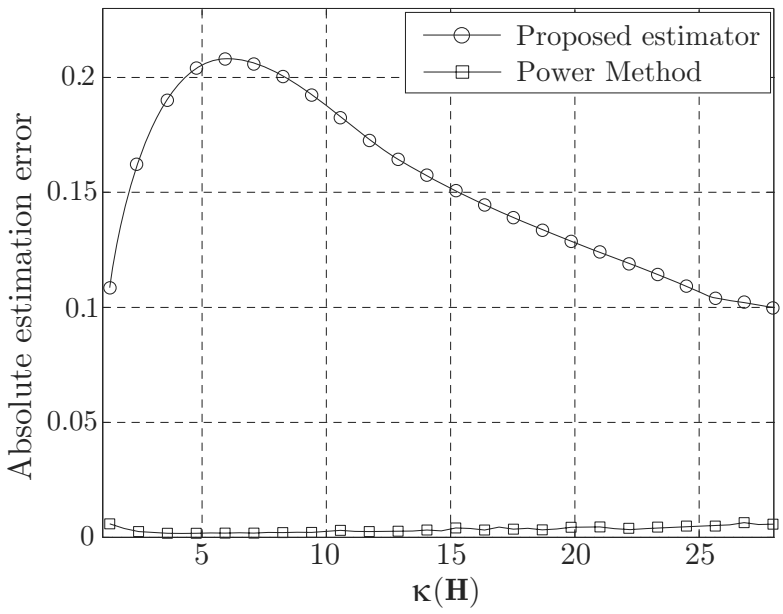
Fig. 4.3(a) shows that the relative estimation error of the proposed joint estimator for $\kappa(\mathbf{R})$ is higher than the one of the estimator only based on the PM. However, as can be seen in Fig. 4.3(b), the error magnitude is not very significant, making this estimator a useful choice for the VB K-best detector. In fact, the relative estimation error of the proposed estimator always remains below the 8% for the typical case of 4×4 Gaussian MIMO channel matrices. As it will be shown in the BER curves, this accuracy is already good enough for our application.

4.4 Threshold Selection

This section addresses a meaningful way to choose the threshold condition number κ_{th} necessary for the VB K-Best detector. The complexity of the proposed K-Best detector depends on the chosen threshold κ_{th}, which is related to the cumulative density function (cdf) of the pdf (4.4). The cdf



(a)



(b)

Figure 4.3. Error of our proposed estimator compared with the error of the Power Method for computing the whole condition number: (a) Relative, (b) Absolute.

stands for the probability of having $\kappa \leq \kappa_{\text{th}}$, which equivalently gives the probability to choose the lower value of K. For the Gaussian MIMO channel such cdf can be obtained as follows

$$F_\kappa(\kappa_{\text{th}}) = \int_1^{\kappa_{\text{th}}} f_\kappa(\kappa) d\kappa \approx \exp\left(-\frac{4{n_T}^2}{\kappa_{\text{th}}^2}\right), \tag{4.15}$$

where, for the sake of simplicity, the approximation for high values of n_T proposed in [72] was used.

As said in Section 2.3, the complexity of tree-search detectors is commonly measured in number of expanded nodes, since this allows a fair comparison among different algorithms. Considering a constellation of size M, the number of expanded nodes for a K-Best algorithm working with the real-valued system (2.7) can be easily calculated as follows:

$$n_K = \sum_{l=1}^{2n_T} n_K^{(l)}, \tag{4.16}$$

where the $n_K^{(l)}$ coefficients contain the number of expanded nodes at level l. For the conventional K-Best detector, the number of expanded nodes can be recursively calculated as

$$n_K^{(l)} = \min\left(K, n_K^{(l-1)} M\right), \tag{4.17}$$

with $n_K^{(1)} = \min\left(K, M\right)$.

For instance, considering a VB K-Best detector that switches between two K values (K_1 and K_2, with $K_1 < K_2$), for a threshold value of κ_{th}, the resulting average number of expanded nodes $n_{\kappa_{\text{th}}}$ is given by

$$n_{\kappa_{\text{th}}} = F_\kappa(\kappa_{\text{th}}) \cdot n_{K_1} + (1 - F_\kappa(\kappa_{\text{th}})) \cdot n_{K_2}, \tag{4.18}$$

where n_{K_i} (for $i = 1, 2$) stands for the number of expanded nodes in a K_i-Best algorithm.

Finally, after combining equations (4.15) and (4.18), an expression for the threshold that provides the desired average number of expanded nodes $n_{\kappa_{\text{th}}}$ is:

$$\kappa_{\text{th}} = 2n_T \sqrt{\log\left(\frac{n_{K_1} - n_{K_2}}{n_{\kappa_{\text{th}}} - n_{K_2}}\right)}. \tag{4.19}$$

It can be noted that the desired average number of expanded nodes determines the threshold value. In the same way, for a given threshold, the average number of expanded nodes can be straightforwardly predicted.

The generalization of the proposed idea for more than two regions of the channel matrix condition number is next described. Considering the combination of a set of N K-Best detectors ($N > 2$) with parameters ranging from K_1 to K_N ($K_1 < K_2 < \ldots < K_N$) and a set of thresholds $\kappa_{\text{th}(1)} < \kappa_{\text{th}(2)} < \ldots < \kappa_{th(N-1)}$, the number of expanded nodes for the resulting VB K-Best detector is

$$\begin{aligned} n_{\kappa_{\text{th}(1)-(N-1)}} &= F_\kappa(\kappa_{\text{th}(1)}) \cdot n_{K_1} + \sum_{i=2}^{N-1} (F_\kappa(\kappa_{\text{th}(i)}) - F_\kappa(\kappa_{\text{th}(i-1)}) \cdot n_{K_i} \\ &+ (1 - F_\kappa(\kappa_{\text{th}(N-1)})) \cdot n_{K_N}. \end{aligned} \tag{4.20}$$

Note that in this case there exist multiple combinations among the threshold values $\kappa_{\text{th}(i)}$ and number of nodes n_{K_i} that fulfill (4.20). Thus, the set of thresholds and detectors should be selected in a meaningful manner to adjust the complexity at a desired value.

4.5 LRA K-Best Detector Based on Condition Number

In Section 2.2.4 the use of channel matrix preprocessing to improve MIMO detection was introduced and in Chapter 3 two preprocessing methods were described and evaluated in terms of complexity and performance. It was shown that the use of LR methods such as the LLL improve the performance of the K-Best detector at the expense of increasing the preprocessing complexity.

It is important the fact that the LR stage has a direct effect on the channel matrix condition number. Fig. 4.4 shows the pdf of the condition number of a Gaussian 4×4 channel matrix, obtained by simulation, before and after the LLL-reduction. It can be noted that, after the LR transformation, low condition number values become more likely than higher values. Thus, the LR step improves the condition number most of the times. From

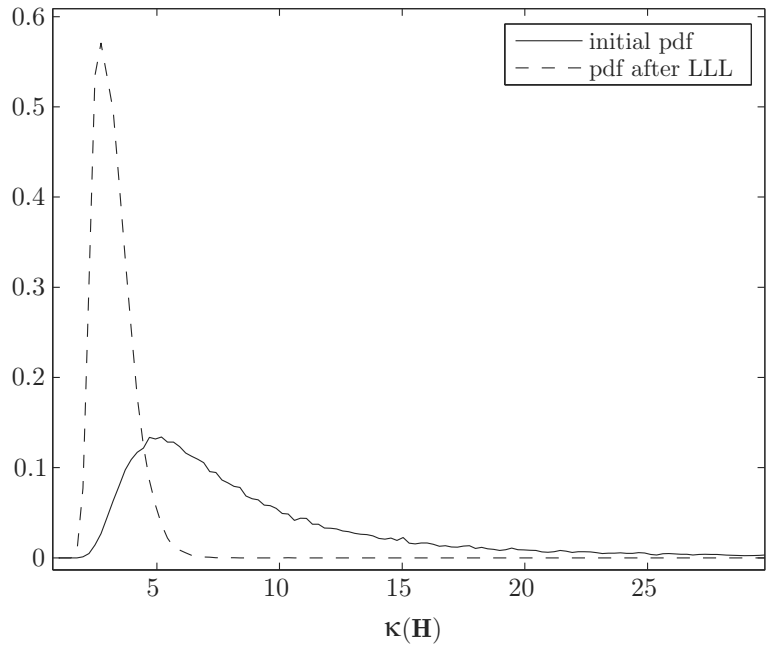


Figure 4.4. Probability density function of the condition number of a 4×4 channel matrix with Gaussian entries, before and after the LLL-reduction.

some of the reasonings in Section 4.2, it is obvious that MIMO detectors will perform better after the LR preprocessing of the channel matrix.

A criterion to measure the reduction of the condition number achieved by the LR method is the ratio between the initial matrix condition number $\kappa(\mathbf{H})$ and the matrix condition number after applying the LR algorithm $\kappa(\widetilde{\mathbf{H}})$. In what follows, it will be denoted by RR (reduction ratio) and it is easily calculated as:

$$\mathrm{RR}(\mathbf{H}) = \frac{\kappa(\mathbf{H})}{\kappa(\widetilde{\mathbf{H}})}. \tag{4.21}$$

Fig. 4.5 shows the average RR obtained by the LLL method for a 4×4 MIMO channel with Gaussian entries, as a function of the initial condition number. It can be seen that the RR increases nearly linearly with κ. Since higher condition number values experience a higher reduction, it seems to

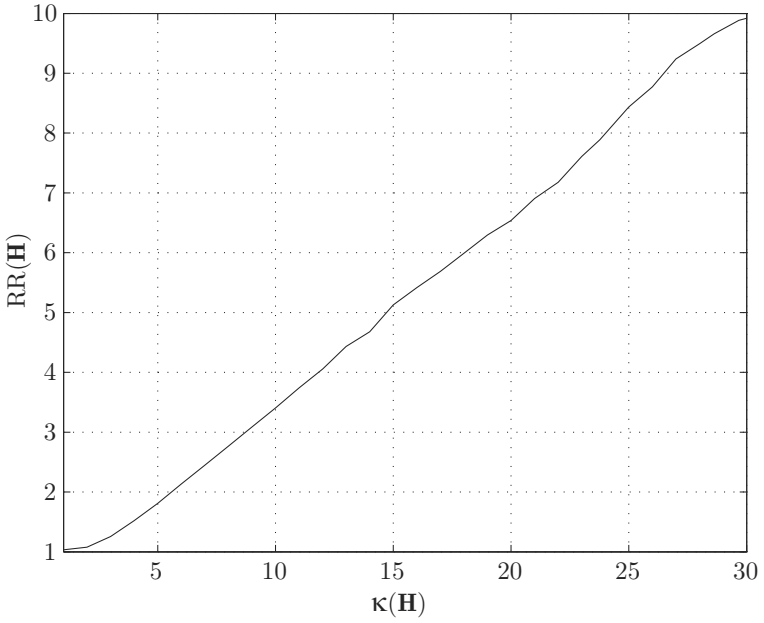


Figure 4.5. Average reduction ratio vs initial condition number for Gaussian 4×4 channel matrices.

be more worth to run the LLL routine for higher condition number matrices. This leads again to the idea of condition number thresholding.

A meaningful approach to reduce the complexity of the LRA K-Best is to perform the LR step only for ill-conditioned channel matrices and to avoid doing it when the condition number is already low. The proposed approach is referred to as LRA K-Best detector based on condition number. Its flow diagram is shown in Fig. 4.6.

In this case, only two condition number regions are needed, associated to switching between using LR or not. Nevertheless, further work could combine this approach with the VB K-Best strategy in order to define more thresholds and K values in the high condition number region. The combined approach could offer more possibilities to tune the performance and complexity.

The preprocessing complexity reduction of the LRA K-Best depends

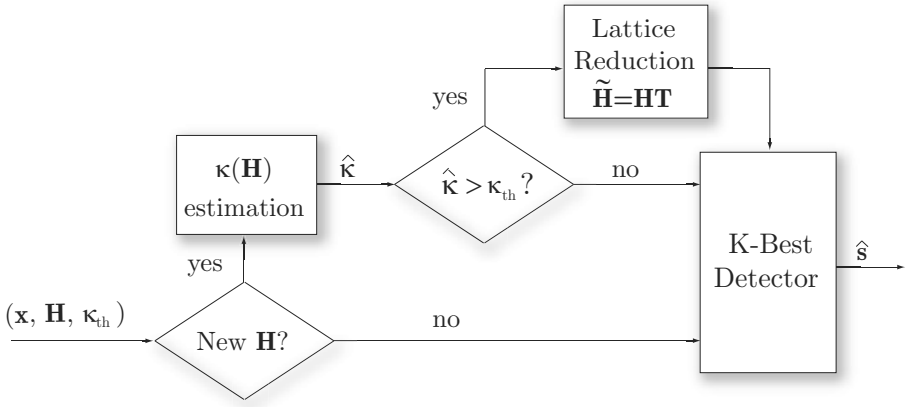


Figure 4.6. Flow diagram of a LRA K-Best Detector based on condition number with channel condition number estimation.

on the chosen threshold κ_{th} and it is related to the function $F_\kappa(\kappa_{\text{th}})$ (4.15), as in the case of the VB K-Best detector. The function $F_\kappa(\kappa_{\text{th}})$, which stands for the probability of having $\kappa \leq \kappa_{\text{th}}$, gives the probability of not performing a previous LR. In the next subsection some examples showing this complexity reduction will be described.

4.6 Results

4.6.1 VB K-Best Detector

First, a 4×4 MIMO system with 16-QAM symbols was considered to simulate the performance of a VB K-Best detector with $K_1 = 2$ and $K_2 = 12$, i.e., combining the 2-Best and the 12-Best detectors. The BER curves of the conventional (fixed-breadth) K-Best detectors with $K = 2$ and $K = 12$ together with the BER curve of the optimal ML detector were also included for reference. The threshold values for the VB K-Best detector, which was chosen following the previously proposed threshold selection method, were $\kappa_{\text{th}} = 10$ and $\kappa_{\text{th}} = 20$. The algorithm was tested first using the exact condition number and afterwards using the proposed QR-based condition

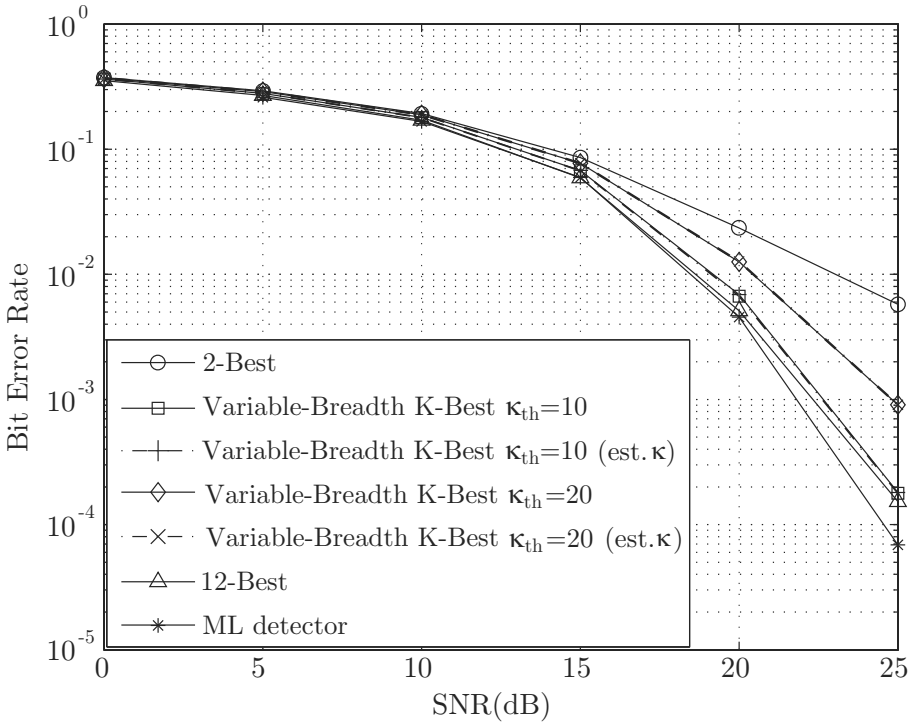


Figure 4.7. BER curves of the proposed VB K-Best detector with two different thresholds on a 4×4 MIMO system using 16-QAM, working with either the exact or estimated 2-norm condition number, all compared to conventional 2-Best, 12-Best and ML detectors.

number estimator.

Fig. 4.7 shows that, as expected, the performance gets worse as the threshold increases. It can be also observed that the slightly imprecise knowledge of the condition number does not modify the performance of the proposed detector at all, since in Fig. 4.7 the discontinuous curves (VB K-Best with estimated κ) match perfectly the continuous ones (VB K-Best with exact κ). Therefore, this estimator is useful for this application despite its magnitude error, which was shown in Fig. 4.3(b).

Table 4.2 includes the average complexity in number of expanded nodes

Table 4.2. Average number of expanded nodes n_K for the VB K-Best detector in a 4×4 MIMO system using 16-QAM.

2-Best	12-Best	$\kappa^s_{\text{th}} = 10$	$\kappa^a_{\text{th}} = 10$	$\kappa^s_{\text{th}} = 20$	$\kappa^a_{\text{th}} = 20$
16	88	37.25	50.03	21.89	26.64

n_K of the considered VB K-Best detectors. Note that the entries labelled as κ^s_{th} denote the average number of expanded nodes calculated by means of simulations, where a high number of Gaussian channel matrices were generated to test their condition numbers. The entries labelled as κ^a_{th} contain the average number of expanded nodes calculated by means of (4.18). It is observed that the analytical values of the number of expanded nodes are more conservative than the simulated ones. Obviously all the n_K values for the VB detector remain between the n_K values of the 2-Best and 12-Best detectors.

Next, the two cases of VB K-Best detector are compared to the conventional K-Best detectors that have similar complexity. The VB K-Best detector with $\kappa^s_{\text{th}} = 10$ can be reasonably compared to a 5-Best detector, which expands 40 nodes in average, and the case with $\kappa^s_{\text{th}} = 20$ is equivalent to a 3-Best detector, which expands 24 nodes. This comparison is quite fair, since the low-complexity calculation of the condition number is carried out only once per channel realization within the preprocessing stage and its complexity gets somewhat masked by that of the node expansion.

It can be seen in Fig. 4.8 that the VB K-Best detectors outperform the conventional ones for a given average complexity, so it is in fact worth to exploit the knowledge of the channel matrix condition number to adjust the breadth of the decoding tree.

Finally, the performance of the VB K-Best detector was tested in the case of working with 64-QAM constellation symbols. Since higher order constellations need more paths to be expanded in order to achieve better performance, for this simulation the 2-Best detector was combined with the

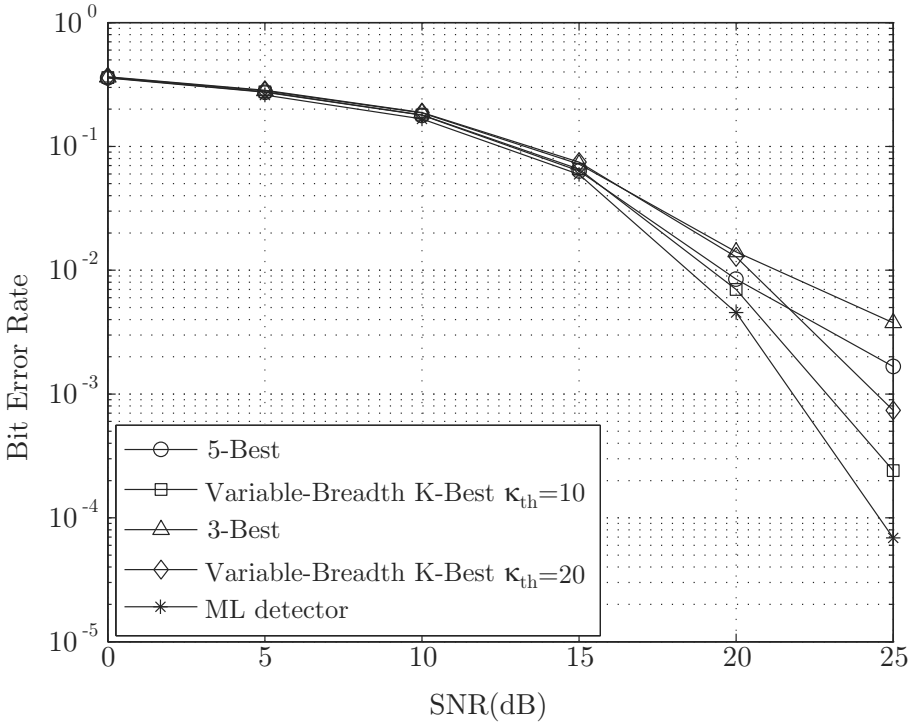


Figure 4.8. BER curves of the proposed VB K-Best detector with two different thresholds on a 4×4 MIMO using 16-QAM, both compared to the conventional K-Best detectors with equivalent complexity (3-Best and 5-Best) and to the ML detector.

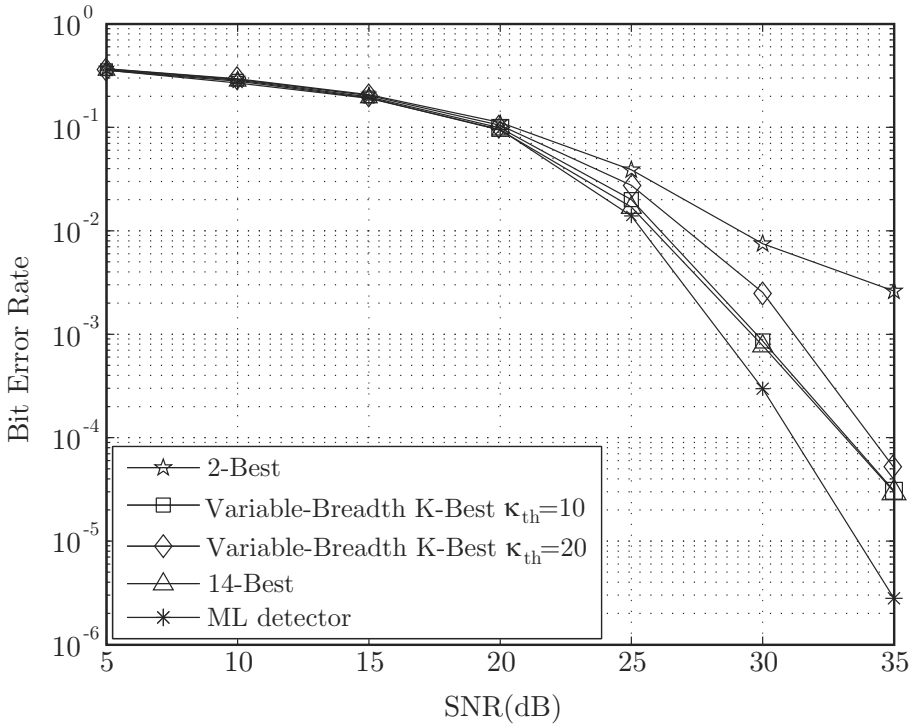


Figure 4.9. BER curves of the proposed VB K-Best detector with two different thresholds on a 4×4 MIMO using 64-QAM, all compared to conventional 2-Best, 14-Best and ML detectors.

14-Best detector instead of with 12-Best. Fig. 4.9 reveals that the selected VB K-Best detector shows a very similar behavior for the 64-QAM case and the same conclusions can be extracted from it. It is interesting to see that the BER curve for the VB K-Best detector with $\kappa_{\text{th}}^{s} = 10$ overlaps the curve of the 14-Best. The performance of the estimator has not been evaluated for this case, since the estimator only depends on the channel matrix statistics and size, and these two features were kept unchanged for this simulation.

Table 4.3 includes the average complexity in number of expanded nodes n_K of the VB K-Best detectors for the 64-QAM case. The notation equals

Table 4.3. Average number of expanded nodes n_K for the VB K-Best Detector in a 4×4 MIMO system using 64-QAM.

2-Best	14-Best	$\kappa^s_{\text{th}} = 10$	$\kappa^a_{\text{th}} = 10$	$\kappa^s_{\text{th}} = 20$	$\kappa^a_{\text{th}} = 20$
16	106	37.39	58.54	23.47	29.30

the one in Table 4.2. The values of n_K for the 2-Best and 14-Best detectors were also included for the sake of comparison and, obviously, the number of expanded nodes of the VB K-Best detectors remains between them. Note that the number of expanded nodes of the VB K-Best detector with $\kappa^s_{\text{th}} = 10$ is much lower than the one of the 14-Best (almost the half that in the analytical case). This result again shows how useful the VB K-Best detector is to reduce the computational cost of the conventional K-Best, sometimes without losing performance and, in the worst case, with only a slight performance loss.

4.6.2 LRA K-Best Based on Condition Number

In this section we evaluate the performance of the proposed LRA K-Best detector based on condition number by means of BER simulations in a 4×4 MIMO system with 16-QAM symbols. The detector was initially formed by the 2-Best and the LLL algorithm. The threshold values to test were $\kappa_{\text{th}} = 10$ and $\kappa_{\text{th}} = 20$, following the threshold selection method in Section 4.4.

Fig. 4.10 illustrates the performance of the LRA K-Best detector based on condition number with both selected thresholds. The two LRA detectors are compared to the conventional K-Best with $K = 2$ (2-Best) and the LRA 2-Best detectors. The ML detection curve is included as a reference. As expected, the performance gets worse as the threshold increases. A meaningful result is that the LRA detector based on condition number with $\kappa_{\text{th}} = 10$ achieves the same performance as the LRA K-Best detector without threshold.

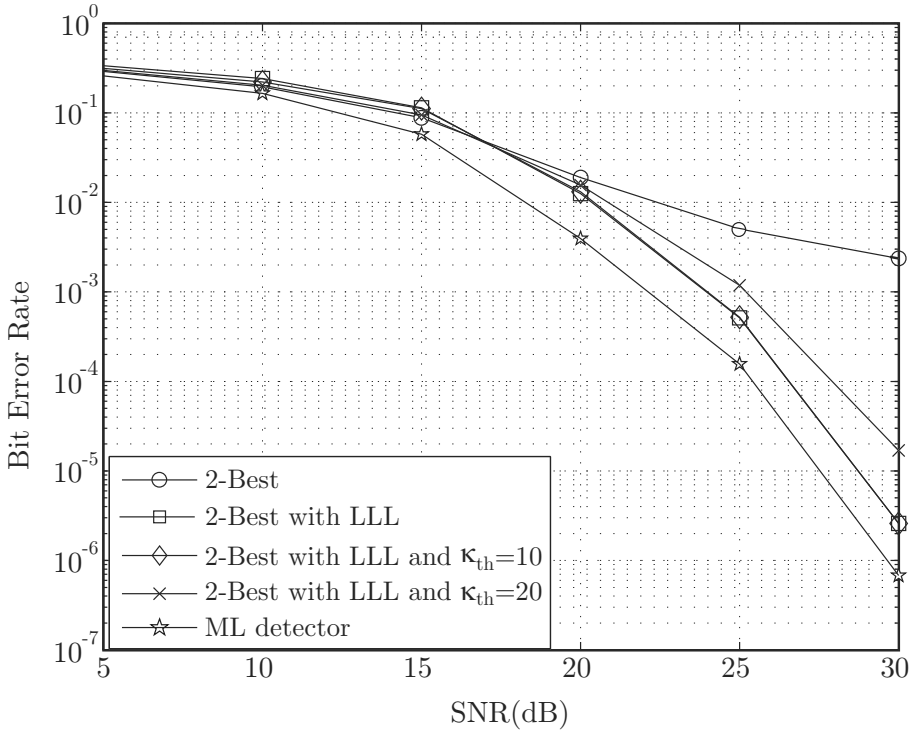


Figure 4.10. BER curves of the LRA K-Best detector based on condition number with two different thresholds on a 4×4 MIMO using 16-QAM, both compared to conventional 2-Best and 2-Best with LLL detectors.

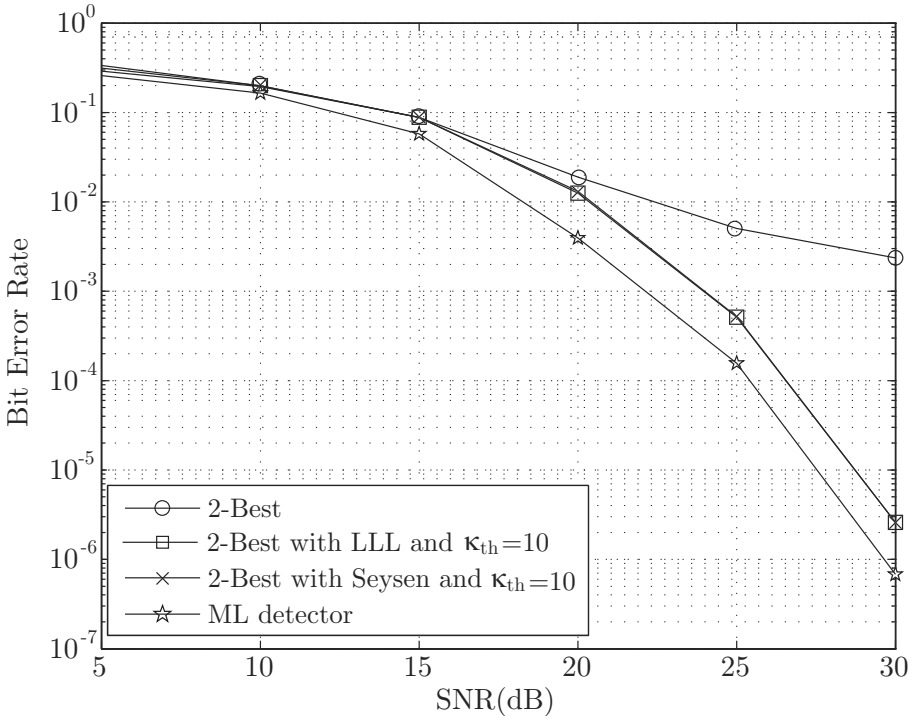


Figure 4.11. BER curves of the proposed LRA K-Best detector on a 4×4 MIMO using 16-QAM, in the two cases of performing the lattice reduction with LLL or Seysen.

Regarding complexity, the opposite event probability of (4.15), i.e. $1 - F_\kappa(\kappa_{\text{th}})$, can be used to obtain the number of LR calls that are performed when a certain threshold κ_{th} is selected. Basic calculations show that for a threshold value of $\kappa_{\text{th}} = 10$, only 47% LR calls are necessary and, as said above, the performance remains the same as in the case with 100% LR calls. For $\kappa_{\text{th}} = 20$, the number of LR calls is reduced to just 14%, at the expense of only a $2 - 3$ dB loss at high SNR regimes.

The BER of the proposed LRA detector with $\kappa_{\text{th}} = 10$ using Seysen's algorithm [65] for LR was also evaluated. The comparison with the use of LLL in Fig. 4.11 shows that both LR algorithms achieve the same performance when used in the condition-number-based scheme. It was shown in [23] that Seysen's algorithm has higher per-iteration complexity than LLL, although it requires a lower number of iterations. In practice, as it was suggested there, parallel hardware structures can be used to compute several independent operations existent in Seysen's algorithm, in order to make this algorithm more efficient than LLL. Therefore, in the end it is up to the user to select the most suitable LR algorithm for the proposed approach depending on the available resources.

4.7 Conclusion

Throughout the first section of this chapter it was developed a variable-breadth K-Best detector that uses two different values of the parameter K in the K-Best tree-search algorithm depending on the 2-norm condition number of the MIMO channel matrix. It was also proposed a condition number estimator that reuses the computation of the QR decomposition of the channel matrix, which is always a previous step before tree-search detection. In addition, a meaningful way of determining the threshold condition number was proposed. The proposed detection approach is built from fixed-complexity parts, which are advised for efficient practical implementations.

Some performance and complexity results showed that the proposed scheme achieves the same performance as other conventional K-Best detec-

tors but with the advantage of having lower average complexity. Thus, it can be said that this scheme offers average power saving. Regarding the condition number estimator, it does not degrade the performance of the VB K-Best detector at all and it has lower complexity than the estimator based exclusively on the PM.

In the second part of the chapter, a LRA K-Best detector based on condition number developed in this thesis was described. The idea behind it is to perform a LR of the channel matrix when its condition number exceeds a given threshold. Results showed that the proposed detector exhibits the same performance as already proposed LRA K-Best methods, with the advantage of not carrying out the LR step for every channel, thus reducing the percent of preprocessing calls to just 47% or even 14%. Moreover, this second proposal was tested in the cases of working with the LLL algorithm and with Seysen's algorithm, without showing noticeable differences between their performances. Therefore, the chosen LR algorithm should finally be determined depending on other practical reasons, for instance on the available hardware resources or complexity restrictions.

The main contributions described throughout this chapter were published in the next cited journal and conference papers. The first version of the VB K-Best detector was included in [74] and an improved version of it in [75]. The proposed method to estimate the channel matrix condition number was proposed in [76], together with the threshold selection method also described in this chapter. Finally, the LRA K-Best detector based on condition number was presented in [77].

5 Efficient Lattice-Reduction-Aided Algorithms

5 Efficient Lattice-Reduction-Aided Algorithms

Lattice-reduction (LR) techniques have been widely employed for different applications such as the data detection in MIMO systems or the signal precoding in MU-MIMO systems. In this chapter, several contributions which involve the use of LR methods are presented. First, the combination of LR with the K-Best algorithm is investigated and alternative implementations that outperform previous proposals are developed. An extended LLL algorithm for LR is proposed to assist the preprocessing part of some LRA K-Best schemes. In the last part of the chapter, this extended LLL algorithm is exploited to decrease the computational cost of several LRA precoding methods. In addition, the most employed signal precoding approaches are evaluated and compared in terms of both computational cost and performance.

5.1 Introduction

In Chapter 3, the use of channel matrix preprocessing before MIMO detection was evaluated considering the VBLAST ZF-DFE and LLL preprocessings before K-Best detection. As explained in Section 2.2.4, preprocessing

can be seen as transforming the MIMO channel matrix $\mathbf{H}$ into a new matrix $\widetilde{\mathbf{H}} = \mathbf{HP}$. After this, the transmitted symbol-vector $\mathbf{s}$ turns into $\mathbf{z} = \mathbf{P}^{-1}\mathbf{s}$.

When matrix $\mathbf{P}$ causes a permutation of the channel matrix columns, the counter permutation is needed by the detected components to make them match the order of the input data. This is the only thing to consider when using preprocessing methods such as the VBLAST ZF-DFE ordering. However, when a previous LR preprocessing has been applied to the channel matrix, the vector to be detected becomes $\mathbf{z} = \mathbf{T}^{-1}\mathbf{s}$. Since $\mathbf{T}^{-1}$ involves a more complicated transformation, the range of values expanded by $\mathbf{z}$ is no longer the same as the one of $\mathbf{s}$. This idea is next explained graphically.

Fig. 5.1(a) shows the initial lattice points existing in an example symbol-constellation $\Omega = \{-1, 0, 1, 2\}$, which is assumed to be transmitted through a particular 2×2 real-valued MIMO system. For instance, if the system is transformed by the matrix $\mathbf{T} = [1, 1; -1, 0]$, with inverse matrix $\mathbf{T}^{-1} = [0, -1; 1, 1]$, the transformed lattice points are those depicted in Fig. 5.1(b). This example shows that the elements of $\mathbf{z}$ do not belong to Ω in all cases. Moreover, each of the components of vector $\mathbf{z}$ may expand a different range of possible values, which will depend on the $\mathbf{T}^{-1}$ matrix, as the comparison between z_1 and z_2 reveals.

A straightforward way of determining all the possible $\mathbf{z}$ values would be to obtain all the $\mathbf{s} \in \Omega^{n_T}$ and, afterwards, transform them with $\mathbf{T}^{-1}$. Note that this requires similar complexity to exhaustive search detection and, hence, it is not useful in our context.

The above described problem led to discard the idea of combining LR with tree-search detectors [32]. Nevertheless, the authors in [60] proposed an implementation of a LR-aided (LRA) K-Best detector that slightly leaves aside the strategy of conventional tree-search detectors (searching within a finite and *a priori* known set of points). Before introducing our proposed approaches, the method in [60] will be described in what follows.

The structure of this chapter is the following. First, Sections 5.2 and 5.3 describe an existing methodology (the LRA-SIC K-Best detector) and a novel scheme for LRA K-Best detection, respectively. The insights of boundary calculation of the transformed lattice points together with an ex-

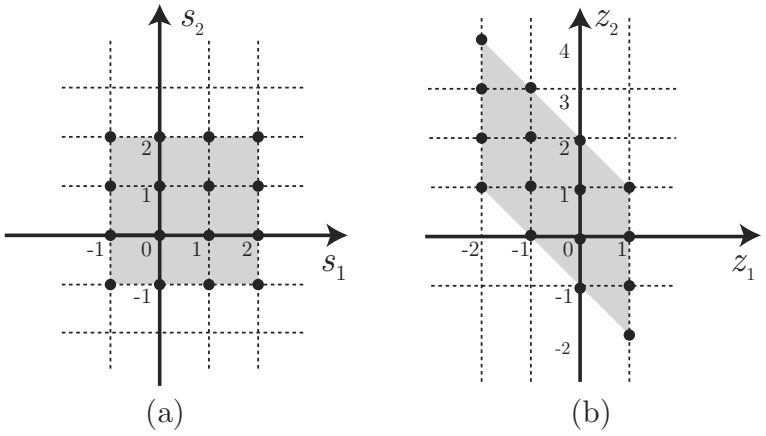


Figure 5.1. (a) Initial lattice points and (b) transformed lattice points.

tended LLL algorithm are also presented in Section 5.3. In Section 5.4, the boundary calculation is exploited to decrease the complexity of previously proposed LRA K-Best detectors. Finally, Section 5.6 addresses the use of the extended LLL method to perform efficient multiuser signal precoding. A comparison among several precoding methods in terms of performance and computational cost is also included.

5.2 LRA-SIC K-Best detection

Considering the real-valued system model (2.7), the application of LR techniques before data detection requires a set of consecutive integer constellation symbols [78]. Therefore, it is necessary to introduce a displacement vector $\mathbf{d}_{2n_T\times 1} = [1,\ldots,1]^T$ to shift and scale the original constellation symbols as

$$\widetilde{\mathbf{x}} = \frac{(\mathbf{x}+\mathbf{H}\mathbf{d})}{2} = \mathbf{H}\left[\frac{(\mathbf{s}+\mathbf{d})}{2}\right] + \frac{\mathbf{v}}{2} = \widetilde{\mathbf{H}}\widetilde{\mathbf{z}} + \frac{\mathbf{v}}{2}, \tag{5.1}$$

where $\widetilde{\mathbf{s}} = (\mathbf{s}+\mathbf{d})/2$ and $\widetilde{\mathbf{z}} = \mathbf{T}^{-1}\widetilde{\mathbf{s}}$. After this transformation, $\widetilde{\mathbf{x}}$ is used to carry out the detection.

Once the symbols are scaled and shifted, the QR factorization of the LR channel matrix is computed ($\widetilde{\mathbf{H}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$) and (5.1) is pre-multiplied by $\widetilde{\mathbf{Q}}^T$. Then, the system becomes:

$$\widetilde{\mathbf{x}}' = \widetilde{\mathbf{R}}\widetilde{\mathbf{z}} + \frac{\widetilde{\mathbf{Q}}^T\mathbf{v}}{2}. \tag{5.2}$$

The steps proposed in [60] to carry out the detection over system (5.2) are described in Fig. 5.2. First, an initial estimate of the solution is calculated at each level of the tree through SIC detection (see Section 2.2.3). Due to its implicit SIC detection, we will call this method *LRA-SIC K-Best*. Second, a neighborhood of points is expanded around this first estimation in order to approximate the set of transformed lattice-points. Note that this method considers only a subset of transformed lattice points to avoid the computation of the complete set.

The LRA-SIC K-Best detector was shown to be meaningful mainly to provide soft information about encoded bits since it outperforms conventional K-Best detector with lower K values. However, the LRA-SIC K-Best detector does not follow the usual implementation of tree-search methods (searching among a finite and *a priori* known set of points). Therefore, most of the existing approaches to optimize the practical implementation of the K-Best algorithm cannot be straightforwardly applied. Moreover, the displacements to grow the neighborhood and the value of N are selected in an ad-hoc manner, which increases the dependency of the algorithm on the actual system parameters.

In [79] a strategy to avoid the predetermined set of displacements required in steps 2) and 5) of the LRA-SIC K-Best was proposed. The method also selected the first child node as in step 5) (solving the SIC problem) but, after that, it followed the Schnorr-Euchner enumeration to find the next neighbor-points by taking zig-zag moves around it. Despite reducing the complexity of the neighborhood calculation, this method still has the drawback of depending on an initial SIC calculation. Thus, it alters the usual structure of K-Best methods. Note also that no justified boundary control rule is provided.

1. Calculate symbol estimate at layer $l = 2n_T$ using a SIC procedure:

$$\hat{z}_{2n_T} = \left\lceil \frac{\widetilde{x}'_{2n_T}}{\widetilde{R}_{2n_T,2n_T}} \right\rfloor,$$

   where $\lceil \cdot \rfloor$ rounds to the nearest integer.

2. Generate a $N(> K)$-point neighborhood around $\hat{z}_{2n_T}$ at the $2n_T$ layer and calculate the Euclidean distance of each point to $\widetilde{x}'_{2n_T}$.

3. Select the K candidates with the lowest Euclidean distances and store them.

4. Decrease $l = l - 1$. For each of the K best paths $\widetilde{z}^i_{l+1:2n_T}$ that were stored in level $l + 1$, generate the symbol estimate at layer l using the SIC procedure:

$$\hat{z}^i_l = \left\lceil \frac{\widetilde{x}'_l - \widetilde{R}_{l,l+1:2n_T}\widetilde{z}}{\widetilde{R}_{l,l}} \right\rfloor, \quad i \in 1, \ldots, K.$$

5. Generate a $N(> K)$-point neighborhood of $\hat{z}^i_l$ at the l layer using a set of predetermined integer displacements and calculate their accumulated Euclidean distances to $\hat{x}^i_l$.

6. Select the K candidates with the lowest distances and store them.

7. If the iteration arrives at level 1 of the tree, stop the algorithm and select the best path as $\hat{\mathbf{z}}$. Transform it into $\hat{\mathbf{s}} = 2\mathbf{T}\hat{\mathbf{z}} - \mathbf{d}$, quantize the value of $\hat{\mathbf{s}}$ if it is outside the initial lattice and give the result as an output. Otherwise go to step 4.

Figure 5.2. LRA-SIC K-Best detector.

5.3 LRA K-Best Detector

In this thesis, we developed an alternative approach to implement LRA K-Best detection that conserves the initial strategy of conventional tree search methods. Previous proposals were based on modifying the original structure of the K-Best algorithm through adapting it to the currently received signal vector (SIC stage). Thus, extra complexity was required at every symbol interval. However, we proposed an scheme based on a low-complexity preprocessing stage that is totally independent of the K-best detector. This means that it is carried out only once per different channel realization, as other stages inside the receiver such as the channel estimation and the LLL lattice-reduction. Hence, the proposed scheme allows for a disjoint implementation of the LR and K-Best detection stages, which makes the algorithm as versatile and modular as required for practical implementations.

Fig. 5.3 shows the MIMO system model described in Section 2.1 including the proposed approach to carry out an efficient data detection. Note that the receiver is based on a conventional K-Best detector which, at every symbol period, needs the following inputs: the received signal vector, the matrices from the QR decomposition, the transformation matrix from the LLL algorithm and the set of points among which the K-Best search must be performed. These stages that are only carried out once per different channel realization appear shadowed in the figure. Note that the set of points that enter the K-Best detector (denoted by Φ) is calculated from the initial constellation set and from the inverse of the channel transformation matrix. The calculation of Φ will be detailed later.

The steps to look for the solution in the proposed LRA K-Best algorithm are described in Fig. 5.4. Note that the proposed LRA K-Best algorithm has exactly the same structure as the conventional K-Best detector without LR. The only difference is step 1), which is performed once per channel realization and can be reused for a long number of runs of the K-Best detection part. The following section is devoted to describe how to compute Φ.

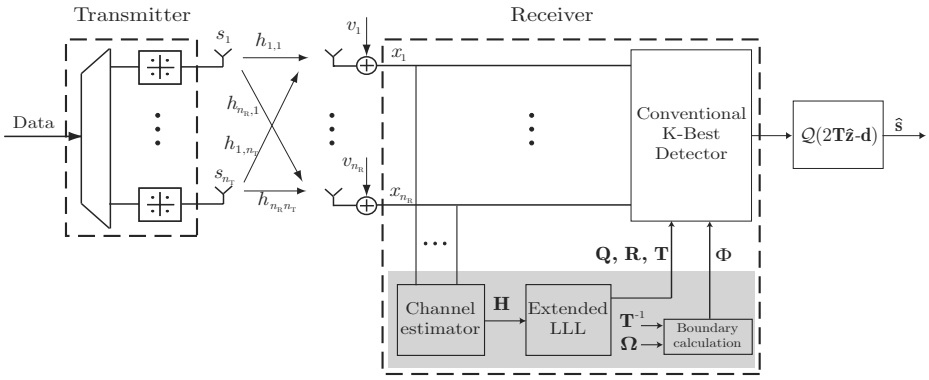


Figure 5.3. Block diagram of a MIMO-BLAST system model including the proposed LRA K-Best detector.

1. Compute all the sets of candidates and store them into Φ.
2. At level $l = 2n_T + 1$, initialize one path with accumulated PED equal to zero.
3. Decrease l and calculate the children nodes of the K best paths that were stored in level $l+1$, considering vector $\phi^{(l)}$. Update the accumulated PED for each path with (2.26).
4. Sort the candidate paths according to their accumulated PEDs and select the K best paths to be expanded in the next level.
5. If $l = 1$, stop the algorithm and select the best path as $\hat{\mathbf{z}}$. Transform it into $\hat{\mathbf{s}} = 2\mathbf{T}\hat{\mathbf{z}} - \mathbf{d}$, quantize the value of $\hat{\mathbf{s}}$ if it is outside the initial lattice and give the result as an output. Otherwise go to step 3.

Figure 5.4. Proposed LRA K-Best detector

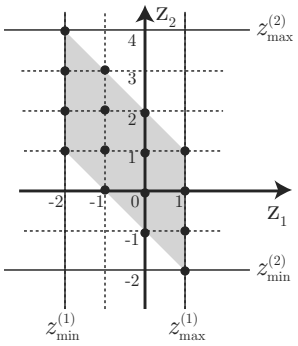


Figure 5.5. Transformed lattice points and boundaries of the transformed lattice.

5.3.1 Boundary Calculation of the Transformed Lattice Points

As said in the introduction, the straight calculation of all the transformed lattice points requires similar complexity to exhaustive search detection. We propose a low-complexity approach that determines only the minimum and maximum integer value of each of the components of vector $\mathbf{z}$, denoted as ${z_{\min}}^{(l)}$ and ${z_{\max}}^{(l)}$ with $l \in \{1, \ldots, n_T\}$. These values provide the boundaries of the transformed lattice at each level of the detection tree. As it can be seen in the example of Fig. 5.5, the values within ${z_{\min}}^{(l)}$ and ${z_{\max}}^{(l)}$, with $l \in \{1, 2\}$, provide a set of integers where the solution at each level is certain to be found. Thus, the elements in such set can be used as the candidate solutions at each level of the tree in LRA MIMO detectors.

Although several extra points are explored in those cases where the original channel matrix is highly skewed, this disadvantage also exists in the other LRA K-Best proposals such as the LRA-SIC K-Best in [60]. Unfortunately, this is the price to pay to avoid an exhaustive candidate calculation.

If matrix $\mathbf{T}^{-1}$ is available for each channel realization, the maximum and minimum $\mathbf{z}$ at levels $l \in \{1, \ldots, n_T\}$ can be obtained from the maximum and minimum value of Ω as follows

$$
\begin{aligned}
{z_{\max}}^{(l)} &= \Omega_{\max} \sum_{j \in P^{(l)}} T_{lj}^{-1} + \Omega_{\min} \sum_{j \in N^{(l)}} T_{lj}^{-1}, \\
{z_{\min}}^{(l)} &= \Omega_{\min} \sum_{j \in P^{(l)}} T_{lj}^{-1} + \Omega_{\max} \sum_{j \in N^{(l)}} T_{lj}^{-1},
\end{aligned} \tag{5.3}
$$

where $P^{(l)}$ stands for the set of j indexes where $T_{lj}^{-1} > 0$ and $N^{(l)}$ stands for the set of j indexes where $T_{lj}^{-1} < 0$.

Once ${z_{\min}}^{(l)}$ and ${z_{\max}}^{(l)}$ have been calculated, the candidates of each level l are obtained as

$$
\phi^{(l)} = \left[{z_{\min}}^{(l)}, {z_{\min}}^{(l)} + 1, \ldots, {z_{\max}}^{(l)} - 1, {z_{\max}}^{(l)} \right]^T. \tag{5.4}
$$

The boundary vectors are also stored in the following $2 \times 2n_T$ matrix:

$$
\mathbf{\Phi} = \begin{bmatrix} {z_{\min}}^{(1)} & {z_{\min}}^{(2)} & \cdots & {z_{\min}}^{(2n_T)} \\ {z_{\max}}^{(1)} & {z_{\max}}^{(2)} & \cdots & {z_{\max}}^{(2n_T)} \end{bmatrix}, \tag{5.5}
$$

which will be an input of the proposed LRA K-Best algorithm.

Even though the proposed boundary calculation needs the inverse of the transformation matrix, $\mathbf{T}^{-1}$, LR algorithms such as LLL [21] only provide matrix $\mathbf{T}$ as an output. The computational cost for inverting matrix $\mathbf{T}$ is cubic with the number of transmitting antennas. In order to fulfill this requirement with low cost, we developed an extended LLL algorithm that efficiently provides matrix $\mathbf{T}^{-1}$, which is described in the following subsection.

5.3.2 Extended LLL Algorithm

In Section 3.3.1 a thorough description of the LLL algorithm was presented. The steps of this algorithm were described in Algorithm 4. Thus, this section will only focus on the additional steps to be included in the original algorithm.

Recall that, in the LLL algorithm, all the transformations performed on matrix $\mathbf{R}$ are stored in matrix $\mathbf{T}$ when the algorithm ends. At the

beginning, matrix $\mathbf{T}$ is initialized to the identity matrix and then two basic operations are performed over it. The first one is a linear combination of columns in order to fulfill condition (3.8), which is done in step 7) of Algorithm 4. The second operation is a column exchange that aims at the fulfillment of condition (3.9) and is addressed in step 12) of Algorithm 4.

We developed an efficient way to calculate matrix $\mathbf{T}^{-1}$ inside the LLL algorithm. It is useful to note that the result of the linear combination of columns in step 7) can be also represented by a matrix $\mathbf{C}$ that postmultiplies $\mathbf{T}$. For instance, the matrix that adds the product of $-\lambda$ by the column l to the column m is simply an identity matrix with its (l, m)-entry replaced by $-\lambda$. As an example with $l = 2$, $m = 3$ and a 3×3 $\mathbf{T}$ matrix:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\lambda \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.6}$$

Since the equality $\mathbf{T}\mathbf{T}^{-1} = \mathbf{I}$ holds, it is also true that $\mathbf{T}\mathbf{C}\mathbf{C}^{-1}\mathbf{T}^{-1} = \mathbf{I}$. If we calculate the matrix $\mathbf{C}^{-1}$ and premultiply $\mathbf{T}^{-1}$ by it, we get that this operation corresponds to the following linear combination of rows in $\mathbf{T}^{-1}$:

$$T^{-1}_{l,:} = T^{-1}_{l,:} + \lambda T^{-1}_{m,:}. \tag{5.7}$$

In a similar manner, the column exchange performed in step 12) can be directly represented by a permutation matrix $\mathbf{P}$ that again postmultiplies $\mathbf{T}$. The matrix $\mathbf{P}$ is built after swapping the columns we need to exchange in $\mathbf{T}$ (column $m-1$ by column m) in an identity matrix. Since for permutation matrices it is true that $\mathbf{P}\mathbf{P}^T = \mathbf{I}$, the premultiplication of $\mathbf{T}^{-1}$ by $\mathbf{P}$ is only this row exchange

$$T^{-1}_{m-1,:} \leftrightarrow T^{-1}_{m,:}. \tag{5.8}$$

Therefore, the extended LLL algorithm provides the QR decomposition matrices of the lattice-reduced channel matrix, together with the transformation matrix and its inverse, as claimed before.

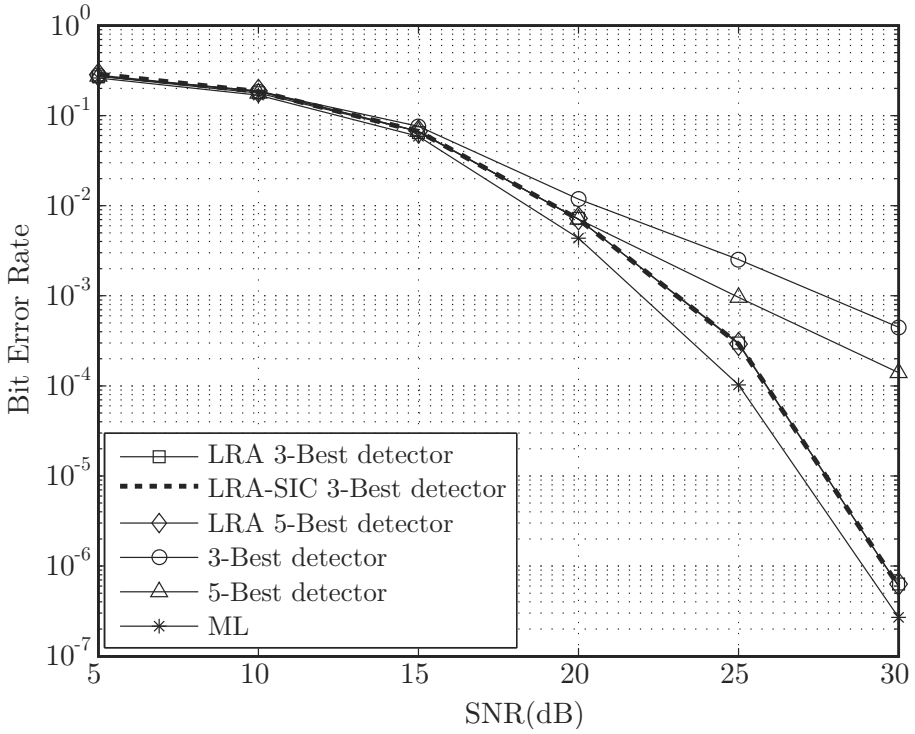


Figure 5.6. BER curves of the proposed LRA K-Best detector with two different values of K (3 and 5) on a 4×4 MIMO system using 16-QAM, both compared to conventional 3-Best, 5-Best and optimal detectors.

5.3.3 Performance Evaluation

A 4×4 MIMO system with a 16-QAM alphabet is here considered. Fig. 5.6 shows the BER performance versus SNR for the proposed LRA K-Best detector, the conventional K-Best detector with $K = 3$ and $K = 5$ and the LRA-SIC K-Best proposed in [60]. For the LRA schemes, the K parameter was set to 3. It can be seen that our proposal performs exactly in the same way as the LRA-SIC K-Best (for the rest of K values and constellations, the same results were obtained).

As Fig. 5.6 shows, the proposed LRA detector outperforms the plain

K-Best detector with the same value of K at high SNR values. The BER degradation compared to optimal ML detection is quite low ($\simeq$ 1 dB). Moreover, the LRA 3-Best detector performs exactly the same as the LRA 5-Best detector; thus, the proposed approach is very suitable for low K values.

Focusing on a particular BER value, e.g. 25 dB, the plain K-Best needs a value of $K = 9$ to achieve the same performance as the LRA K-Best detector with $K = 3$. Therefore, the LRA K-Best detector can achieve the same results as the plain K-Best with a much lower K value.

5.3.4 Computational Cost Analysis

The proposed LRA K-Best detector can be divided in two stages. On the one hand, the preprocessing stage, which includes the LLL and the boundary calculation, is carried out only when the channel changes and, thus, it can be performed off-line. On the other hand, the K-Best detector is run when a new symbol vector is received, which happens many times for a given realization of a block-fading channel.

The complexity of the LLL was previously analyzed in Chapter 3. Note that the computational cost of the extended LLL algorithm is almost the same as the one of the LLL but including $2n_T$ sums and $2n_T$ products when one of the conditions of the LLL is fulfilled.

Regarding the boundary calculation, it can be observed that, according to (5.3), it is performed with very simple mathematical calculations. The upper bound for the cost of the proposed preprocessing stage corresponds to $2n_T$ products and $2n_T - 1$ additions. Generally, this computational cost becomes even lower in practice since the unimodular matrix $\mathbf{T}^{-1}$ usually contains several entries that are equal to 0, -1 and 1, which avoids several products and additions.

As discussed in Section 2.3, the complexity of tree-search detectors is commonly measured in number of expanded nodes since this allows a fair comparison among different algorithms. The number of expanded nodes for any K-Best algorithm can be easily calculated as $n_K = \sum_{l=1}^{n_T} n_K^{(l)}$, where the $n_K^{(l)}$ coefficients contain the number of expanded nodes at level l (see

Table 5.1. Stages of the proposed LRA K-Best and the LRA-SIC K-Best schemes.

	Proposed LRA K-Best	LRA-SIC K-Best
Preprocessing cost	Extended LLL Boundary calculation	LLL
Per-symbol-vector cost	$n_T K$ visited nodes	$n_T K$ visited nodes SIC calculation

Section 4.4). For the proposed LRA K-Best detector, the coefficients $n_K^{(l)}$ can be recursively calculated as

$$n_K^{(l)} = \min\left(K, n_K^{(l-1)}(z_{\max}^{(l)} - z_{\min}^{(l)} + 1)\right), \tag{5.9}$$

with $n_K^{(0)} = 1$. Note that the complexity is upper-bounded by $n_T K$, as in the conventional K-Best algorithm.

Table 5.1 collects the contributions to the preprocessing and per-symbol-vector computational costs for the proposed LRA K-Best detector and the LRA-SIC K-Best detector [60]. Taking a look at the differences between our proposal and the LRA-SIC K-Best detector, the main advantage of our scheme is that the computational cost of the boundary calculation stage ($2n_T$ products and $2n_T - 1$ additions) is much lower than the one required for calculating the SIC solution of each group of n_T symbols (with $\mathcal{O}(n_T^2)$). Moreover, unlike the SIC computation, the boundary calculation is carried out only at the preprocessing, which reduces the complexity even more.

Next, when comparing the computational cost of our proposal with that of the conventional K-Best, both have the same worst-case number of expanded nodes $n_T K$. However, as the LRA K-Best achieves better performance than the K-Best, a fair comparison requires to increase the K parameter in the conventional K-Best to provide the same performance as the LRA K-Best does. This fact leads to an increase in the per-symbol-vector cost, which is the main contribution to the overall cost of the algorithm, as

discussed above.

5.4 Complexity Reduction Using Boundaries

In this section, we propose several alternative applications for the boundaries of the transformed lattice. Note that, once these boundaries are available, they can be used to reduce the search for the solution at each level of the decoding tree of the LRA-SIC K-Best detector described in Section 5.2. By limiting the search in this way, the number of explored candidate solutions is substantially reduced. This first strategy is called *LRA K-Best with Candidate Limitation (LRA-CL)*. The second strategy is based on the dynamic K-Best detection approach (see Section 2.3.3) and it led to two different detection proposals: the *Dynamic-K LRA K-Best* and the *Dynamic-N LRA K-Best*.

5.4.1 LRA K-Best with Candidate Limitation

Considering that the boundaries of the transformed lattice have been calculated as presented in Section 5.3.1, the only steps of the LRA-SIC K-Best detector that have to be modified in order to discard erroneous solutions are steps 3) and 6) (see Fig. 5.2). Step 6) is reformulated as:

6) Pick $N > K$ integer values around $\hat{z}_l^i$ at the l layer following a zig-zag strategy around the SIC solution. In case of reaching the boundaries, follow the same direction and reduce N to the value of $N = \lceil (N/2) \rceil$, where $\lceil \cdot \rceil$ denotes rounding to the higher integer. If the other boundary is also reached, stop exploring candidates. Finally calculate the Euclidean distances of the candidates to $\hat{x}_l^i$.

Fig. 5.7 represents an example of the candidate selection following a zig-zag strategy around the SIC solution. It will be considered for this case a value of $N = 8$ and a solution from the SIC detector equal to -2, labelled in the figure as SIC. The first row of numbers surrounded by circles represent the order in which each candidate is explored when there is no candidate limitation. However, when the candidate limitation is applied, if a value for the lower boundary of $z_{\min} = -3$ is supposed, the value

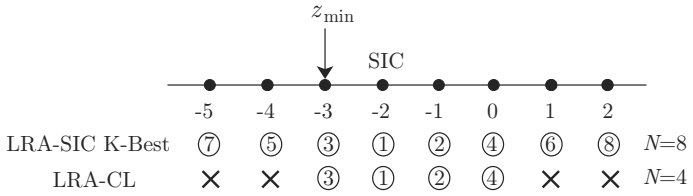


Figure 5.7. Representation of the order in which candidate points are explored in the LRA-SIC K-Best scheme and in the LRA-CL scheme.

of N is reduced to $N = 4$ and the selected candidates points now range from -3 to 0. Therefore, the proposed candidate limitation can reduce the number of candidates without decreasing the detection performance, as will be discussed below.

The modifications over step 3) can be straightforwardly predicted.

Results and Complexity Discussion

The computational complexity comparison has been performed for a 4×4 MIMO system working with 16-QAM and 64-QAM constellations. For the sake of simplicity, the simulations were carried out using the real representation of the MIMO system and an uncoded scheme, although the proposed technique can be easily adapted to a complex system. The LLL algorithm [21] was employed for the LR operation.

For a $2n_T \times 2n_T$ MIMO system, it can be easily seen that the number of explored paths at each run of the original LRA K-Best algorithm equals $N+(2n_T-1)NK$, since every level needs to explore NK candidates except the first level, where only N are explored. In the case considered for our simulations, the number of levels of the decoding tree is 8, and thus, the number of explored paths at each run of the algorithm equals $N + 7NK$. From now on, it will be considered that $N = K + 1$, which leads to a number of explored candidates of $7K^2 + 8K + 1$. Note that the complexity analysis has been performed under the assumption of working with a real representation of the MIMO system.

Table 5.2. Average number of expanded candidates for the LRA-SIC and LRA-CL detectors (16-QAM case).

LRA-SIC 3-Best			LRA-SIC 5-Best		
[60]	LRA-CL	Reduction (%)	[60]	LRA-CL	Reduction (%)
88	71	19	216	148	31.5

Table 5.3. Average number of expanded candidates for the LRA-SIC and LRA-CL detectors (64-QAM case).

LRA-SIC 5-Best			LRA-SIC 10-Best		
[60]	LRA-CL	Reduction (%)	[60]	LRA-CL	Reduction (%)
216	185	14.4	781	570	27

Tables 5.2 and 5.3 show the average number of expanded nodes for the LRA-SIC K-Best and LRA-CL algorithms and the achieved corresponding reduction using the LRA-CL strategy. For the 16-QAM case, simulations were run with the LRA-SIC 3-Best and LRA-SIC 5-Best detectors. As Table 5.2 shows, the LRA-CL scheme decreases the number of explored candidates of the LRA-SIC 3-Best in a 19%. For the LRA-SIC 5-Best case, the complexity is reduced in a 31.5%.

The results for a 64-QAM constellation are collected in Table 5.3, considering the LRA-SIC 5-Best and LRA-SIC 10-Best detectors. It can be seen that, for the 5-Best scheme, the number of candidates is decreased in a 14.4%. For the LRA-SIC 10-Best case, the reduction is 27% of the candidates. Therefore, it can be concluded that the higher the value of K is, the higher the percentage of reduction can be achieved.

5.4.2 Dynamic-K and Dynamic-N LRA K-Best Detectors

In this subsection, a dynamic selection of the parameters of the LRA-SIC K-Best detector is proposed in order to decrease its average complexity. Note that the complexity of the LRA-SIC K-Best detector is subject to the

two parameters K and N, with $N > K$. Parameter K is responsible for the number of stored paths per level and N for the number of candidate solutions that are explored before choosing the K best ones.

As it can be seen from the proposed boundary calculation, vector $\phi^{(l)}$ allows to determine the number of valid candidate solutions at each level l of the decoding tree. It seems reasonable that levels with higher number of candidate solutions should not discard as many paths as levels with lower number of candidates. Therefore, the computational complexity of the algorithm can be decreased either by having both a different K_l and N_l value at each level (later called as *Dynamic-K* scheme), or keeping the number of stored paths K unaltered and only assigning different N_l values per level (*Dynamic-N* scheme). The first approach leads to both a variable number of explored paths $K_{l-1}N_l$ and a variable number of stored paths K_l at each level, whereas the second approach explores KN_l paths and stores K paths (which is a fixed value).

Following the above mentioned strategies, we propose two different dynamic algorithms. In the Dynamic-K algorithm (Dyn-K) a non-linear distribution of the K_l values at each level of the tree is suggested. A generalized logistic function [80] rounded to the higher integer by the operator $\lceil \cdot \rceil$ is used. The resulting K_l values are calculated according to the number of candidates at level l and also to the K value of a same performance fixed K-Best algorithm, as follows:

$$K_l = \left\lceil 1 + \frac{(K-1)}{(1 + 0.5 \exp\left(-(L_l - M)\right))^2} \right\rceil, \tag{5.10}$$

where L_l stands for the number of candidate points at level l:

$$L_l = (\phi_2^{(l)} - \phi_1^{(l)}) + 1 \tag{5.11}$$

and M denotes the central point of the range of values expanded by $\mathbf{L} = [L_1, \ldots, L_l, \ldots, L_{2n_T}]$

$$M = (\max_{\forall l}(\mathbf{L}) - \min_{\forall l}(\mathbf{L}))/2. \tag{5.12}$$

Let this distribution of K_l values be clarified by means of an example. Considering a 4×4 complex MIMO system (8×8 in its real-valued representation), working with a 16-QAM constellation, 8 detection levels are

Table 5.4. Example of assigned K_l values in a 8×8 real-valued MIMO system, associated to the set of number of candidates L_l for each of the 8 detection levels.

Level (l)	Candidates (L_l)	Assigned K (K_l)
1	3	3
2	2	2
3	4	4
4	6	5
5	3	3
6	3	3
7	1	1
8	8	5

necessary to obtain the whole detected vector. At each level, the number of candidate points (L_l) for a given channel realization can be calculated by using (5.11) and previously related equations. A set of possible values of L_l is given as an example in Table 5.4 for each level l. The next step for the calculation of the K_l values is the calculation of M, which can be easily performed using (5.12) as $M = (8-1)/2 = 3.5$.

Once the values of L_l and M are available, a fixed K-Best algorithm to be compared has to be chosen, for instance 5-Best ($K = 5$). Inserting L_l, M and K into (5.10), the values shown in the third column of Table 5.4, labelled as K_l, are achieved. It can be easily checked that the average K value is now 3.62, which is lower than the K value of the fixed 5-Best for this particular case. The smallest value for vector N at each level was considered, being calculated as $N_l = K_l + 1$.

In the Dyn-K scheme, the fact of working with different K_l values affects steps 4) and 7) of the above described LRA-CL scheme. These steps should be modified in order to operate with the K_l best paths instead of with the K best paths. In the same way, steps 3) and 6) of the above described LRA-CL scheme should be modified in order to operate with the N_l integer values around the SIC solution instead of with the N values.

Table 5.5. Average number of expanded candidates for the LRA-CL, Dyn-K and Dyn-N detectors (16-QAM case).

LRA-SIC 3-Best			LRA-SIC 5-Best		
LRA-CL	Dyn-K	Dyn-N	LRA-CL	Dyn-K	Dyn-N
71	66.2	54.2	148	131.8	124.3

This strategy has been shown to reduce the average number of stored paths (K) and the average number of expanded nodes of the LRA-CL algorithm, without decreasing the performance, as will be presented in the next section. Therefore, this first dynamic approach saves average power consumption in practical implementations.

In the second dynamic scheme, called Dynamic-N algorithm (Dyn-N), the value of K remains constant and the N_l values are now calculated as follows:

$$N_l = \left\lceil 1 + \frac{(K-1)}{(1+0.5\exp\left(-(L_l - M)\right))^2} \right\rceil . \tag{5.13}$$

In this case, only steps 3) and 6) of the above described LRA-CL scheme should be modified in order to operate with the N_l integer values around the SIC solution instead of with the N values. This strategy has been shown to reduce the average number of explored paths and to keep the average number of stored paths fixed, since K is the same for all levels.

Tables 5.5 and 5.6 show the average number of expanded candidates for the LRA-CL algorithm and for the two proposed dynamic versions of this method, labelled as Dyn-K and Dyn-N. It can be seen that dynamic schemes can further reduce the complexity.

Table 5.7 shows the comparison between the average K values for some fixed K-Best detectors and the average K values for the Dyn-K detection scheme that achieves the same performance. Also, the relative reduction of K in % obtained with the dynamic scheme is presented. It can be observed that the Dyn-K detector reduces the average K value and, thus, the average number of stored paths of the algorithm.

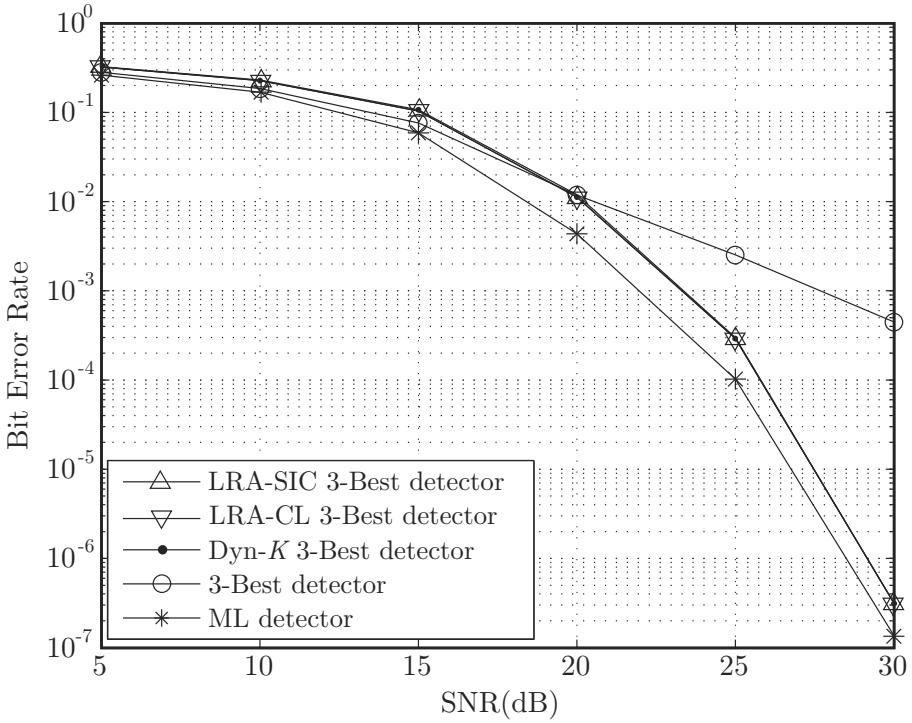


Figure 5.8. BER curves of the proposed LRA K-Best schemes (LRA-CL and Dyn-K) on a 4×4 MIMO system using 16-QAM, compared to conventional 3-Best, LRA 3-Best and ML detectors.

Table 5.6. Average number of expanded candidates for the LRA-CL, Dyn-K and Dyn-N detectors (64-QAM case).

LRA-SIC 5-Best			LRA-SIC 10-Best		
LRA-CL	Dyn-K	Dyn-N	LRA-CL	Dyn-K	Dyn-N
185	155	143	570	473	486.7

Table 5.7. Reduction of the average K values with a Dyn-K detector.

16-QAM			64-QAM		
Fixed	Dyn-K	Reduction (%)	Fixed	Dyn-K	Reduction (%)
3	2.8	6.7	5	4.3	14
5	4.4	12	10	8.1	19

Finally, Fig. 5.8 shows the performance of the LRA-CL and Dyn-K schemes with $K = 3$, in a 4×4 MIMO system using 16-QAM, compared to conventional 3-Best, LRA-SIC 3-Best and ML detectors. It can be seen that the BER curves of the proposed low-complexity detectors overlap the BER curve of the LRA-SIC 3-Best detector, and all of them improve the performance of conventional 3-Best detector for SNR $\geq$ 20 dB. Therefore, the proposed schemes can decrease the average complexity without modifying the performance. It has been checked that the BER curve for the Dyn-N algorithm also overlaps the curves of LRA-SIC K-Best, LRA-CL and Dyn-K.

5.5 Relationship among the proposed hard-output schemes

To further clarify the relationship among the suboptimal techniques proposed in Chapter 4 and so far in Chapter 5, we show how these techniques

compare among each other in Fig. 5.9. The first level contains the basic preprocessing stages that, when combined with the K-Best detector, result in the proposed approaches. These basic stages are the condition number calculation and thresholding, the LR stage and the boundary calculation of the transformed lattice.

The second level includes the VB-K-Best detector, which uses condition number thresholding in conjunction with K-Best detection. If LR is used instead, there are two possible alternatives: either the proposed LRA K-Best (with boundary calculation) or the existing LRA-SIC K-Best (with a SIC calculation per received symbol-vector).

At the third level of the figure there are three detection strategies which are built from further modifications of the detectors in the second level. These detectors are the LRA K-Best detector based on condition number, which includes condition number thresholding and LR, the LRA-SIC K-Best detector with candidate limitation, which uses the boundaries of the transformed lattice to limit the search in the LRA-SIC K-Best, and the Dynamic-K or Dynamic-N LRA-SIC schemes, which exploit the knowledge about the number of candidates among the boundaries to set different values of either the K or the N parameter.

General guidelines on the suitability of each proposed technique for a certain application are hard to be given. Nevertheless, we found useful to describe at least some considerations to be taken into account when using these techniques.

In general, all the proposed approaches are intended to reduce the complexity of the detection stage at the expense of including extra preprocessing complexity. The practical cases when this strategy is worth, however, will depend on the time the channel remains constant. If the channel has a fast variation in time, those algorithms with a low-complexity preprocessing stage are more suitable, for instance the VB K-Best of the LRA-SIC K-Best detectors. On the other hand, if the channel remains constant during the transmission of many symbol-vectors, the preprocessing complexity has a small contribution to the total complexity of the algorithm. For the latter case, algorithms such as the LRA K-Best of LRA K-Best based on condition

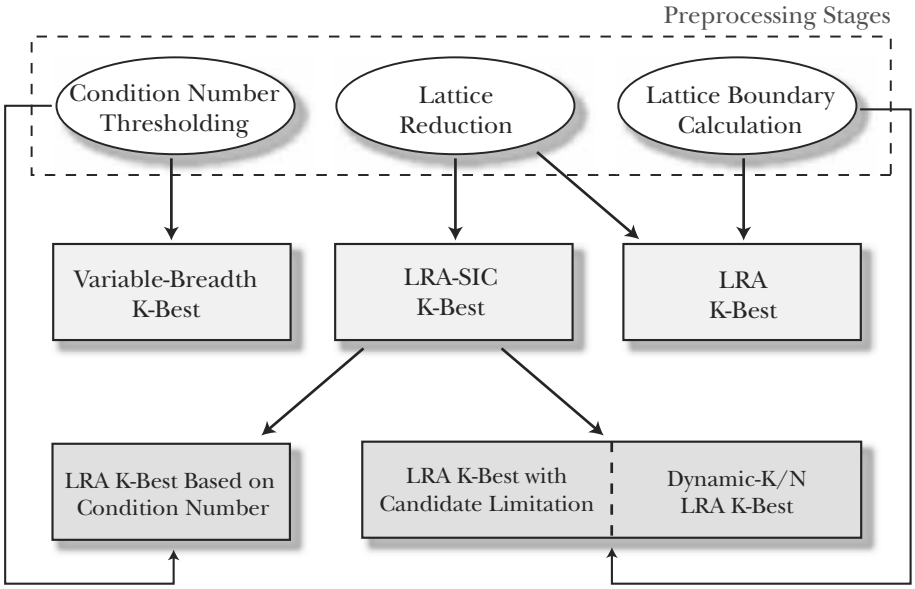


Figure 5.9. Relationship among all the hard-output strategies proposed in Chapters 4 and 5.

number may be more appropriate. Finally, if the practical application has fixed-complexity requirements, algorithms such as the LRA-SIC K-Best CL or Dyn-K and Dyn-N LRA-SIC schemes would be less suitable, due to their variable complexity. If there are no fixed-complexity requirements, then the latter algorithms are more advisable than the plain LRA-SIC K-Best.

5.6 Multiuser Precoding using the Extended LLL Method

As shown in Section 2.5, there is a close relationship between the MIMO point-to-point system model and the one of multiuser systems. In fact, there are several MIMO detection algorithms that can be also used to perform multiuser signal precoding.

In Section 2.5, some well-known precoding techniques were introduced, such as the vector perturbation, zero-forcing and Tomlinson-Harashima

approaches. LR algorithms have also been widely employed for signal precoding, some examples are the schemes proposed in [16] and [81].

In this section, we present several LRA precoding techniques and propose their combination with the extended LLL algorithm. Our aim is to save computational cost when the inverse of the transformation matrix is required by the LRA precoding algorithms. The preprocessing and per-symbol-vector computational cost of all the multiuser precoding algorithms presented in this thesis are both evaluated. Then, the impact that the extended LLL has in the overall cost of those LRA algorithms that employ the inverse of the transformation matrix is investigated. Finally, all the algorithms are compared in terms of both computational cost and performance in order to determine the algorithm with the best trade-off.

5.6.1 Lattice-Reduction-Aided Precoding

In [81] it was proposed a simple and efficient way to approximate the perturbation vector (2.42) based on LR. A LR stage is applied to the columns of the pseudoinverse matrix $\mathbf{H}^{+}$, giving

$$\mathbf{H}_{\mathrm{R}}^{+} = \mathbf{H}^{+}\mathbf{T}, \tag{5.14}$$

where $\mathbf{H}_{\mathrm{R}}^{+}$ is the new precoding matrix with less correlated columns. After this, the perturbation vector is approximated as:

$$\mathbf{p}_{app} = -\mathbf{T}\mathcal{Q}\{\mathbf{T}^{-1}\mathbf{s}'\}, \tag{5.15}$$

where $\mathcal{Q}\{.\}$ denotes the componentwise quantization of a $2K$-dimensional vector to the scaled integer lattice $M\mathbb{Z}$.

Finally, the precoded signal is

$$\mathbf{s} = \mathbf{H}^{+}(\mathbf{s}' + \mathbf{p}_{app}), \tag{5.16}$$

as displayed in Fig. 5.10. Note that both $\mathbf{T}$ and $\mathbf{T}^{-1}$ are needed by the algorithm.

5.6.2 Enhanced Lattice-Reduction-Aided Precoding

There is another variant of this algorithm which is based on SIC [81], which will be called as the LRA-SIC precoding method. Here the matrices $\mathbf{Q}$ and

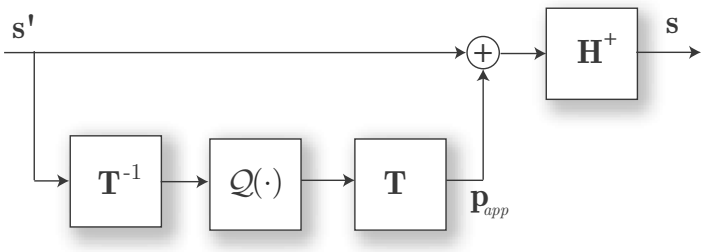


Figure 5.10. Block diagram of the Lattice-Reduction-Aided precoding.

$\mathbf{L}$ are calculated with a QR-type decomposition of the LR matrix $\mathbf{H}_{\mathrm{R}}^{+}$ to obtain:

$$\mathbf{Q}\mathbf{H}_{\mathrm{R}}^{+} = \mathbf{L}, \tag{5.17}$$

where $\mathbf{Q}$ contains orthogonal rows and $\mathbf{L}$ is a $2K \times 2K$ lower triangular matrix. It can be observed in Fig. 5.11 that the first step of the algorithm builds the vector

$$\mathbf{q} = -\mathbf{Q}\mathbf{H}^{+}\mathbf{s}'. \tag{5.18}$$

Next, the components of $\widetilde{\mathbf{q}}$ are calculated as follows

$$\widetilde{q}_k = \mathcal{Q}\{q_k - \sum_{l=1}^{k-1} L_{k,l}\widetilde{q}_l\}, \qquad k = 2, \ldots, 2K. \tag{5.19}$$

Finally, the perturbation vector is calculated as

$$\mathbf{p}_{app} = \mathbf{T}^{-1}\widetilde{\mathbf{q}}. \tag{5.20}$$

This algorithm requires again the matrices $\mathbf{T}$ and $\mathbf{T}^{-1}$ together with the QR decomposition of $\mathbf{H}_{\mathrm{R}}^{+}$.

5.6.3 Lattice-Reduction-Aided Tomlinson-Harashima Precoding

The THP strategy, which was introduced in Section 2.5, can be also performed after a LR of the channel matrix [82], leading to the LRA-THP precoding approach. Now the components of vector $\mathbf{s}'$ in (2.45) must be

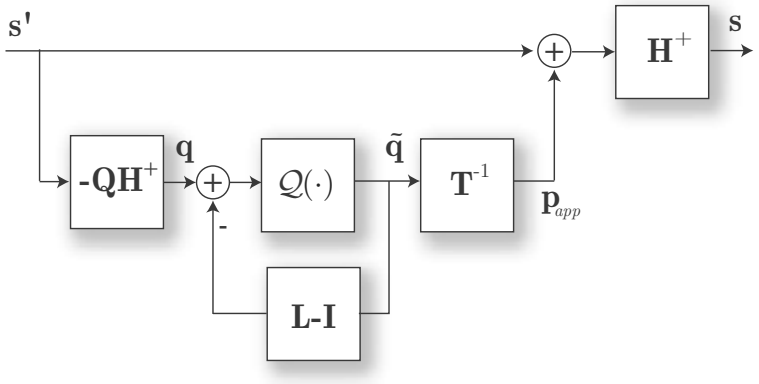


Figure 5.11. Block diagram of the Enhanced Lattice-Reduction-Aided precoding.

replaced by the components of a new vector $\mathbf{d} = \mathbf{T}^{-1}\mathbf{s}'$ (see Fig. 5.12). In this case, the QR is performed over the lattice-reduced channel matrix. This method also employs $\mathbf{T}$ and $\mathbf{T}^{-1}$.

5.6.4 Performance Comparison

A brief performance comparison among the precoding algorithms under study was included for the sake of completeness. The performance of all the methods was analyzed by showing the achieved BER within a range of E_b/N_0 values. A system with $N = 4$ transmit antennas and $K = 4$ users with a QPSK constellation was considered. To average the results, 10^6 channel matrices with Gaussian entries of zero mean and unit variance were generated.

Fig. 5.13 shows that the precoding methods employing LR (labelled as LRA, LRA-SIC and LRA-THP) outperform the ZF and THP algorithms and attain full receive diversity. It is interesting to see that, although the LRA-THP is not a vector perturbation method, it improves the performance of the LRA-SIC method and almost reaches the exhaustive estimation of the perturbation vector obtained through optimum sphere-encoding (labelled as VP(SE)).

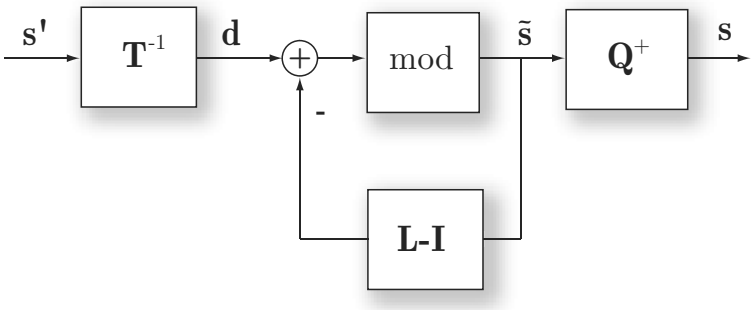


Figure 5.12. Block diagram of the Lattice-Reduction-Aided Tomlinson-Harashima precoding.

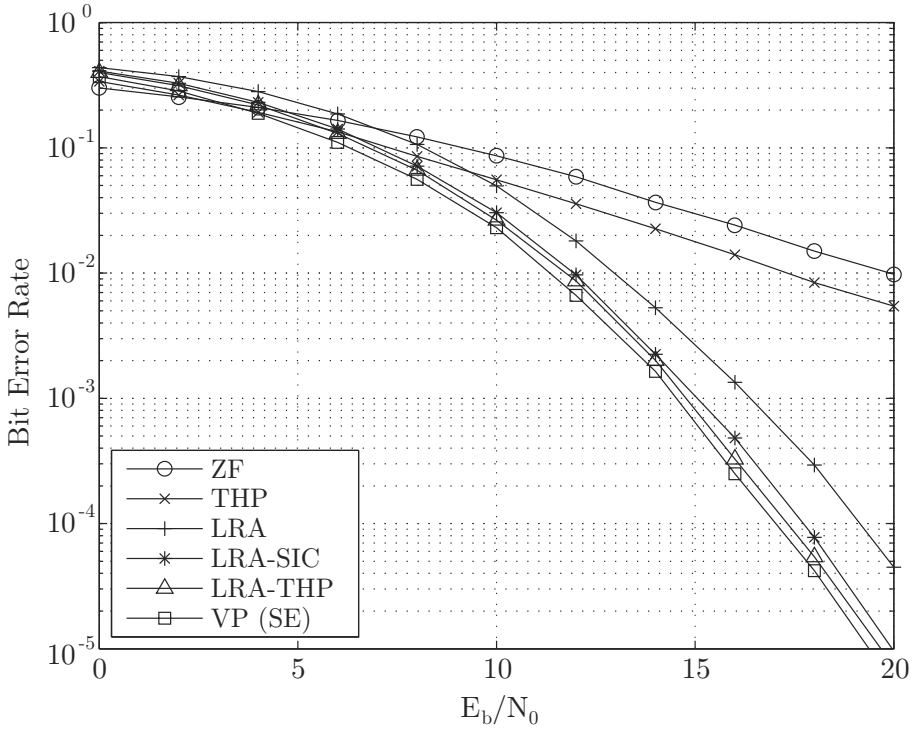


Figure 5.13. BER curves of the five precoding algorithms under study in a system with $N = 4$ transmitter antennas and $K = 4$ users using QPSK symbols.

5.6.5 Computational Cost Analysis and Comparison

In practical applications, the computational complexity restrictions can help to decide which is the most suitable algorithm. Thus, it is important to analyze the different precoding schemes from a computational point of view. An interesting fact is that the overall computational cost of the above described precoding algorithms comes from two different parts that must be analyzed independently. On the one hand, the preprocessing stages (QR decomposition, LLL and inversion of matrices) together with some products matrix by matrix that do not depend on the symbol-vector to transmit are only carried out when the channel changes, thus they can be done off-line. On the other hand, the remaining calculations are performed every time that a new symbol vector is transmitted, which happens several times for a given realization of a block-fading channel.

Per-symbol-vector computational cost

Table 5.8 collects the per-symbol-vector number of sums and products for the five precoding algorithms under study, which will be next explained. The per-symbol-vector arithmetic operations of the ZF precoding come from the product $\mathbf{H}^{+}\mathbf{s}'$. The per-symbol-vector cost of the LRA precoding is due to (5.15) and (5.16), and the one of LRA-SIC precoding to (5.18-5.20) and (5.16). Finally, the cost of the THP relies mainly on (2.45) and the product by matrix $\mathbf{Q}^{+}$ before transmission. The increased cost of the LRA-THP with respect to the conventional THP is due to the product of the initial data vector by $\mathbf{T}^{-1}$. Both THP and LRA-THP also require $2K$ modulo operations.

Preprocessing computational cost

Table 5.9 shows which of the precoding algorithms under study need any of the following preprocessing stages: QR decomposition, LLL lattice reduction and matrix inversion (either $(\mathbf{H}\mathbf{H}^{H})^{-1}$ or $\mathbf{T}^{-1}$).

The number of arithmetic operations of the QR factorization is $2(2K)^2(2N-2K/3)$ for a $2N \times 2K$ matrix, assuming that it has been computed through Householder reflections [64]. On the other hand, the inversion of a $2K \times 2K$ matrix through Gaussian elimination requires an amount of $K(2K+1)$ divi-

Table 5.8. Per-symbol-vector cost of precoding algorithms.

	Sums	Products
ZF	$2K^2 + 4KN - 3K - 2N$	$2K^2 + 4KN - K$
LRA	$8K^2 + 4KN - 2K - 2N$	$8K^2 + 4KN$
LRA-SIC	$10K^2 + 4KN - 3K - 2N$	$10K^2 + 4KN - K$
THP	$2K^2 + 4KN - K - 2N$	$2K^2 + 4KN - K$
LRA-THP	$6K^2 + 4KN - 3K - 2N$	$6K^2 + 4KN - K$

Table 5.9. Main preprocessing stages of precoding algorithms.

	QR	LLL	$\mathbf{T}^{-1}$	$(\mathbf{H}\mathbf{H}^H)^{-1}$
ZF	Yes	No	No	No
LRA	No	Yes	Yes	Yes
LRA-SIC	Yes	Yes	Yes	Yes
THP	Yes	No	No	No
LRA-THP	Yes	Yes	Yes	No

Table 5.10. Additional preprocessing cost of precoding algorithms.

	Sums	Products
ZF-SIC	-	$4KN + 2K$
LRA	$16K^2N - 4K^2 - 4KN$	$16K^2N$
LRA-SIC	$24K^2N - 8K^2 - 4KN$	$24K^2N$
THP	-	$4KN + 2K$
LRA-THP	-	$4KN + 2K$

sions, $(2(2K)^3+3(2K)^2-10K)/6$ products and $(2(2K)^3+3(2K)^2-10K)/6$ sums.

The number of sums and products of any other preprocessing calculations apart from the ones discussed in Table 5.9 were included in Table 5.10. In most of the algorithms, the calculations are needed to perform the pseudoinverse of the channel matrix.

Note that the total number of arithmetic operations required by the LLL and fcLLL algorithms (already presented in Chapter 3) were directly considered for the present complexity evaluation.

Overall computational cost

The overall computational cost depends on the number of symbol vectors that can be transmitted through the same channel without needing to estimate it again, i.e. the time that the channel remains unchanged. Hence, the overall cost of the precoding of a symbol vector equals

$$C_{\text{tot}} = C_{\text{psv}} + \frac{C_{\text{pre}}}{L_{\text{ch}}}, \tag{5.21}$$

where it has been considered that the preprocessing cost (C_{pre}) is shared among L_{ch} transmitted symbol vectors.

For the computational cost comparison, it is useful to know from which value of L_{ch} the per-symbol-vector cost (C_{psv}) exceeds $C_{\text{pre}}/L_{\text{ch}}$ and thus, decreasing the preprocessing cost is not so necessary. Table 5.11 shows

Table 5.11. Minimum value of L_{ch} for $C_{\text{psv}} > (C_{\text{pre}}/L_{\text{ch}})$.

	ZF	LLL	LLL-SIC	THP	THP-LLL
L_{ch}	5	9	14	5	9

these L_{ch} values for the five precoding algorithms under study in a 4×4 system. Note that for values of $L_{\text{ch}} \leq 5$, the per-symbol-vector costs of all the algorithms are lower than their respective preprocessing costs. When $L_{\text{ch}} > 14$, the opposite thing happens.

Taking into account the results above, the total number of arithmetic operations of the precoding algorithms under study were depicted in Fig. 5.14, for $N = 4$ and different values of K. For the LRA, LRA-SIC and LRA-THP precoding methods, the computational cost with the conventional calculation of the inverse of the transformation matrix is displayed in continuous line and the one with the extended fcLLL algorithm in dashed line. Figs. 5.14(a) and 5.14(b) show the cost when $L_{\text{ch}} = 5$ and $L_{\text{ch}} = 20$, respectively. Note that in the first case, the extended fcLLL algorithm decreases slightly the overall cost of the methods, whereas in the second case, the cost decrease is hardly noticeable. The reason is that, for all the precoding algorithms, $(C_{\text{pre}}/L_{\text{ch}}) > C_{\text{psv}}$ when $L_{\text{ch}} = 5$, so a decrease in the preprocessing cost is worthwhile here. On the other hand, for $L_{\text{ch}} = 20$ the per-symbol-vector cost is predominant and masks the preprocessing cost reduction.

Focusing now on the comparison among algorithms, in Fig. 5.13 it was observed that the LRA-THP and LRA-SIC precoding methods achieved the lowest BER. In Fig. 5.14, the cost of the LRA-THP is lower than the one of the LRA-SIC. The reason is that, whereas the THP schemes directly generate the precoded vector to transmit, the vector perturbation methods require extra calculations to generate the precoded vector after knowing the perturbation vector. Therefore, the LRA-THP method seems to be a better choice than the LRA-SIC. Nevertheless, in case of tighter computational requirements, either THP or ZF without LR should be employed instead.

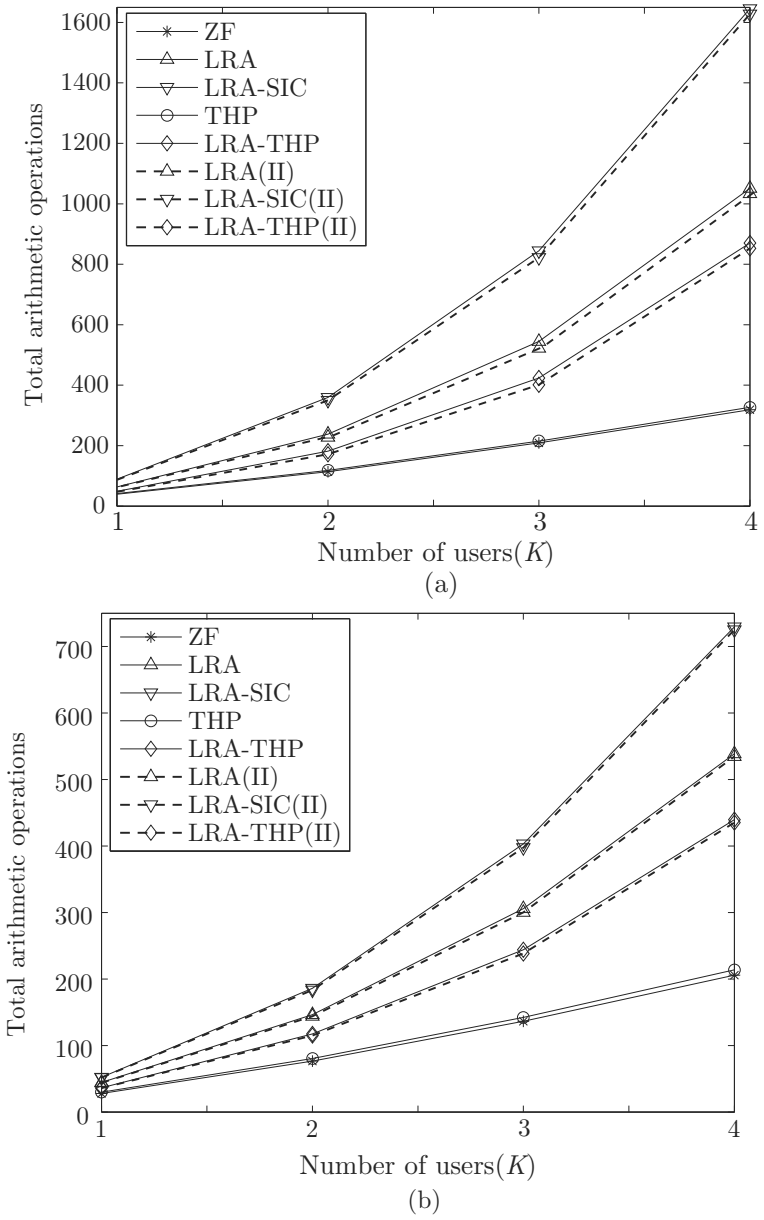


Figure 5.14. Total number of arithmetic operations of the five precoding algorithms under study for a system with $N = 4$. (a) $L_{\text{ch}} = 5$. (b) $L_{\text{ch}} = 20$.

5.7 Conclusion

This chapter introduced the problem of using LR methods with K-Best tree-search detectors. After describing a previously proposed approach to combine both techniques (the LRA-SIC K-Best), a novel LRA K-Best detector was presented. The proposed approach performs a LR of the channel matrix using an extended version of the LLL algorithm, which provides the inverse of the channel transformation matrix. After the LR step, the candidate points for each level of the detection are computed via an efficient boundary calculation and the detection is carried out with any conventional K-Best detector. The simulations revealed that the proposed scheme substantially improves the performance of conventional K-Best detectors at high SNR. The main advantage of this LRA K-Best detector is that it does not need an initial estimate of the solution at each level. Therefore, it can be easily merged with already existing K-Best schemes.

Making use of the proposed lattice boundary calculation and extended LLL algorithm, several schemes to decrease the computational complexity of already existing LRA K-Best detectors were proposed. The first one was called as LRA K-Best scheme with candidate limitation (LRA-CL). It employed the boundaries of the transformed lattice to discard in the LRA K-Best scheme those candidates that are certainly outside the transformed lattice. In addition, two schemes with a dynamic distribution of the parameters K and/or N based on the boundaries were proposed. The computational complexity was further decreased by means of these last schemes without performance loss.

In the last section of this chapter, the extended LLL method was exploited to decrease the computational cost of some LRA precoding schemes for multiuser communications. The performance and computational cost of five widely employed precoding algorithms were compared. It was shown that the overall cost depended on two different terms: the preprocessing cost and the per-symbol-vector cost. The impact that each term had over the total cost depends on the time that the channel remained unchanged (L_{ch}). It was observed that for $L_{\text{ch}} = 5$, a reduction in the preprocessing cost decreased the overall cost. On the other hand, for $L_{\text{ch}} = 20$ the per-

symbol-vector cost was predominant and masked any preprocessing cost reduction. Thus, this parameter should be taken into account to direct our efforts towards reducing the term that is predominant in the overall cost.

Finally, a performance comparison revealed that the LRA-THP and the LRA-SIC precoding methods got the lowest BER. Regarding computational cost, the LRA-THP was more efficient than the LRA-SIC method, making the LRA-THP the precoding option with the best tradeoff.

The following publications contain the main contributions described in this chapter. The extended LLL algorithm was published in [83]. The different LRA K-Best schemes with reduced complexity were presented in [84] and the complexity comparison among the most employed preprocessing techniques is contained in [85].

6 Efficient Soft-Output Detection

6 Efficient Soft-Output Detection

The use of soft detection (demodulation) in MIMO-BICM systems can substantially improve their performance with respect to the use of hard detection. In practical MIMO-BICM systems, demodulators have to deliver finite word-length (quantized) log-likelihood-ratios (LLR). In this chapter, we propose an efficient modification of the fixed-complexity sphere decoder for MIMO-BICM systems working with quantized LLRs. Our approach reduces the complexity of previously proposed schemes via pruning strategies that exploit the clipping and quantization of LLR. Numerical results confirm that our scheme achieves a significant complexity reduction (by 37% for the case of 2 bits per LLR and by 31% for the case of 3 bits per LLR) with negligible degradation in bit error rate performance.

6.1 Introduction

As it was already presented in Section 2.4, which was devoted to MIMO-BICM, the demodulation (or soft detection) and channel decoding are not performed jointly at the receiver of these systems, but in two differentiated stages. First, the demodulator provides reliability information (soft

outputs) about the transmitted coded bits in the form of real-valued log-likelihood ratios (LLRs). Next, these values are used by the channel decoder to make final decisions on the transmitted coded bits.

Some remarkable demodulators were also mentioned in Section 2.4, especially those based on SD or tree-search. Among the cited schemes, the soft-output fixed-complexity SD (SFSD) proposed in [55] exhibits better performance than the list-based SD described in [52] while keeping fixed complexity. Nevertheless, its complexity can be still to high for some practical applications.

Channel codes used in current wireless communications (LDPC, turbo codes, etc.) work with bit sequences of thousands to ten thousands of bits. Hence, saving the LLRs delivered by the soft MIMO detector before passing them to the channel decoder requires a lot of memory and, consequently, large chip sizes. This requirement is translated into both higher power consumption and cost. Due to this, in practical systems, real numbers are usually represented with a finite word-length. Thus, LLR quantization is being paid increasing attention recently [86][87]. Since meaningful guidelines on the necessary number of bits for efficient LLR quantization together with quantization design rules are well detailed in [86], the design of LLR quantizers will be out of the scope of the present thesis. On the other hand, we will focus on the use of LLR quantization to reduce the complexity of the soft MIMO detection stage.

In this chapter we propose an efficient soft detection scheme with fixed complexity for MIMO-BICM systems working with quantized LLRs, which is based on the SFSD. This approach makes use of the fact that the quantized LLRs belong to an *a priori* known discrete set of values (quantization levels) to avoid some of the computations carried out by the soft detector. In addition, LLR clipping is also included to save complexity when a certain LLR value exceeds a previously selected clipping level. Simulations to evaluate the performance and complexity of the efficient SFSD scheme are presented, showing the usefulness of the method.

The chapter is structured as follows. Section 6.2 describes the MIMO-BICM system model with turbo decoding used in this chapter. In Sec-

tion 6.3 a soft-output detection method based on fixed-complexity tree-search is presented. A proposed approach to perform soft detection delivering quantized outputs is detailed in Section 6.4, where a novel pruning based on quantization together with clipping-based pruning are introduced. Finally, Section 6.5 is devoted to the performance and complexity results of the proposed approach and conclusions are presented in Section 6.6.

6.2 System model

In Section 2.4 of Chapter 2, a generic MIMO-BICM system with n_T transmit antennas, n_R receive antennas ($n_R \geq n_T$) and a certain SNR was described. Since Fig. 2.16 shows a general MIMO-BICM system where neither the channel encoder nor the channel decoder are specified, the specific channel coding considered in this chapter is described next.

We consider the MIMO-BICM system depicted in Fig. 6.1, where two parallel concatenated convolutional codes are used at the transmitter. The coded bits are mapped to symbols which are split into the n_T antennas and transmitted simultaneously. A block Rayleigh fading channel constant during L_{ch} time intervals is assumed. At the receiver, the soft information is decoded by a turbo decoder. Note that no information is exchanged between the soft detector and the turbo decoder.

As seen in Section 2.4, the max-log approximated LLRs calculated by the soft MIMO detector are:

$$L_{j,b} = \frac{1}{\sigma^2}\left[\min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(0)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 - \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(1)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2\right], \tag{6.1}$$

where $\mathcal{X}_{j,b}^{(0)}$ and $\mathcal{X}_{j,b}^{(1)}$ denote the sets of symbol vectors containing the bth bit of the jth symbol equal to 0 and 1, respectively, and $x_{j,b}$ is the bth bit in the bit label of symbol s_j.

Once the LLRs are available, they are split into two sets (L_{in_1} and L_{in_2}) and enter the turbo decoder, which has to update them iteratively. At each iteration, the ith BCJR decoder obtains the a posteriori information, L_{d_i}, using the *a priori* information, L_{a_i}, and the information provided by the

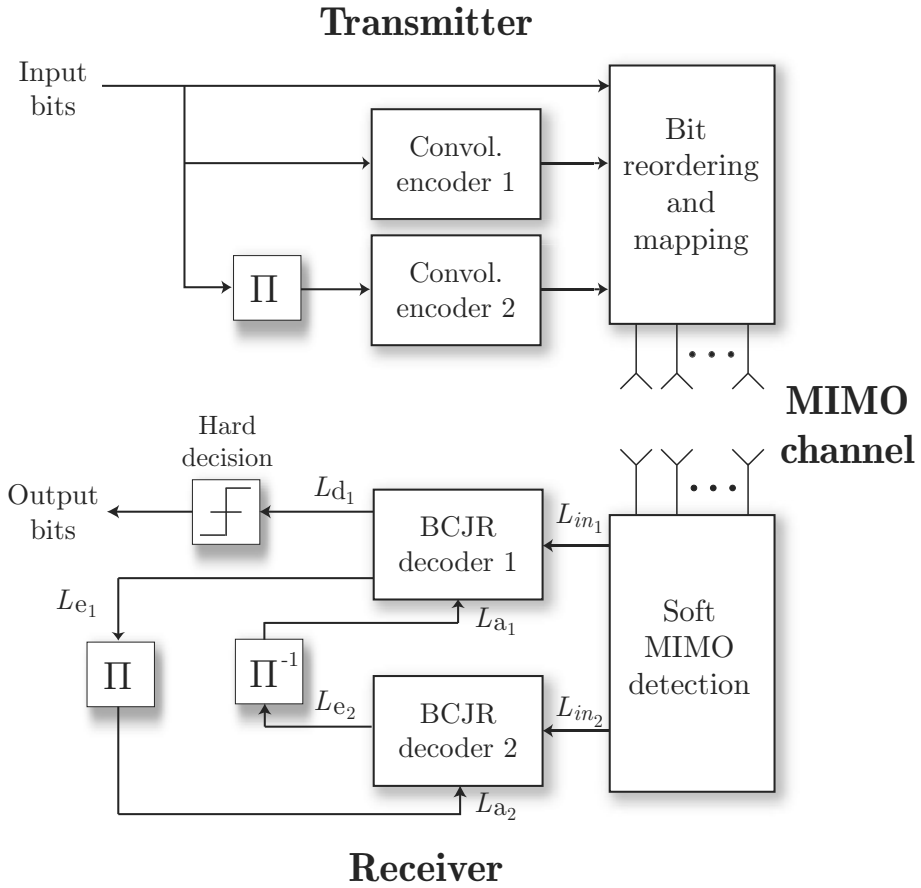


Figure 6.1. MIMO-BICM block diagram with parallel concatenated convolutional encoding and turbo decoding.

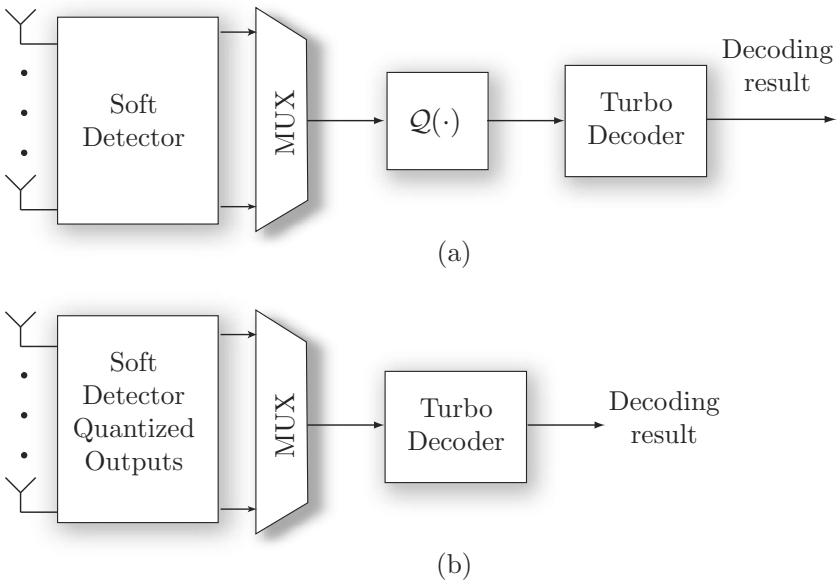


Figure 6.2. MIMO-BICM receiver: (a) Using soft detection and LLR quantization, (b) Using soft detection with quantized outputs.

soft MIMO detector, L_{in_i}. Next, this information is converted into extrinsic information, L_{e_i}, and it is deinterleaved and passed to the other BCJR decoder as *a priori* information. In the final iteration, the first decoder generates a posteriori information about the uncoded bits, L_{d_i}, which is used to obtain the estimated transmitted bitstream.

When the MIMO-BICM system with turbo decoding is assumed to work with LLR quantization, the receiver block diagram turns into the one in Fig. 6.2(a), where a quantization step is inserted between the soft detection and the turbo decoding. This receiver would be equivalently implemented as Fig. 6.2(b) shows, since, in this case, the soft detector directly delivers quantized LLRs. This consideration will be taken into account next for the design of efficient soft detectors delivering quantized LLRs.

6.3 Soft-output Fixed-complexity Detection

In Section 2.3.5, the FSD was described as an efficient method to achieve quasi-ML hard detection performance. The method consisted of a preprocessing stage followed by a predetermined tree-search with fixed complexity. An interesting strategy to provide soft information after the FSD search was proposed in [55]: the soft-output FSD (SFSD). The two main stages of the SFSD method are depicted in Fig. 6.3 for an example search-tree with $n_T = 4$, QPSK symbols and $T = 1$.

First of all, a conventional hard-output FSD tree-search is carried out (Fig. 6.3(a)). This search gives as a result 4 SIC branches of 4 symbols each, from where the path with the minimum accumulated PED is selected as the ML solution. The distance associated to this ML path approximates one of the two minima in (6.1):

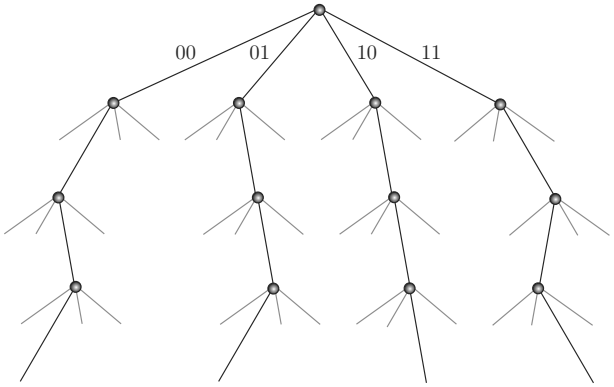
$$d^{\mathrm{ML}} = \|\mathbf{y} - \mathbf{H}\mathbf{s}^{\mathrm{ML}}\|^2. \tag{6.2}$$

For each j and b, the second minimum in (6.1) can be computed as

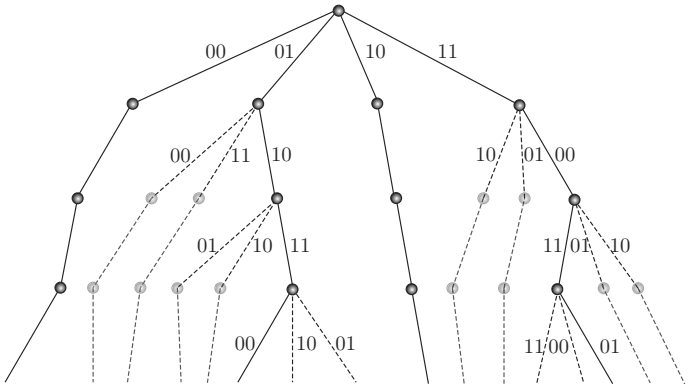
$$\bar{d}_{j,b} = \min_{\mathbf{s} \in \mathcal{X}_{j,b}^{(\overline{x}_{j,b}^{\mathrm{ML}})}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2, \tag{6.3}$$

where $\overline{x}$ denotes the complement of bit x. Recall that $\mathbf{s} \in \mathcal{X}_{j,b}^{(\overline{x}_{j,b}^{\mathrm{ML}})}$ represents the counter-hypothesis to the ML solution for bit b in layer j.

The SFSD [55] extends the FSD tree search in order to obtain the minimum distances in (6.3) after (6.2) has been solved. The SFSD starts from the candidate list obtained by the hard-output FSD (4 SIC branches of 4 symbols each) and adds new candidates for the counter-hypotheses (see Fig. 6.3(b)). Since the first level of the hard-output FSD tree is already totally expanded, all the necessary values to compute the LLRs of the symbol bits in the first level are available. To begin the list extension, the best N_{iter} paths are selected from the initial hard-output FSD list (in this example, $N_{\mathrm{iter}} = 2$). This is motivated by the heuristics that the lowest-distance paths may be candidates differing from the best paths in only a few bits. The symbols belonging to these N_{iter} paths are picked



(a)



(b)

Figure 6.3. Decoding trees of the SFSD algorithm for a 4×4 MIMO system with QPSK symbols and $N_{iter} = 2$: (a) Hard-output stage and (b) Soft-output extension.

up from the root up to a certain level l; at level $l-1$, additional $\log_2 M$ branches are explored, each of them having one of the bits of the initial path symbol negated. Afterwards, these new partial paths are completed using the SIC path, as done in the hard-output FSD scheme. The same operation is repeated until the lowest level of the tree is reached. Note that N_{iter} depends on the symbol constellation.

The values that achieved almost max-log performance for a 4×4 system in [55] were $N_{\text{iter}} = \{2, 4, 6\}$ for QPSK, 16-QAM, and 64-QAM, respectively. In the remainder of this thesis we will consider these values.

6.4 Proposed SFSD with quantized outputs

Quantizer designs for the LLRs delivered by the demodulator of a MIMO-BICM system have been introduced in [86]. A q-bit quantizer uses $K = 2^q$ quantization intervals $\{I_1, I_2, ..., I_K\}$. These intervals are specified by $I_k = [i_{k-1}, i_k]$ where $\{i_0, i_1, i_2, ..., i_K\}$ are the quantizer thresholds ($i_0 = -\infty$ and $i_K = \infty$ by convention). The quantized LLR value $\Lambda_{j,b}$ is given by

$$\Lambda_{j,b} = \mathcal{Q}(L_{j,b}) = \lambda_k, \quad \text{if} \quad L_{j,b} \in I_k, \tag{6.4}$$

where λ_k is the kth quantization level. For fixed q, the ideal quantizer maximizes the mutual information between the input code bits and the quantized LLRs [88]. Since the design of this ideal quantizer is difficult in practice, [86] proposed an alternative quantizer design that maximizes the mutual information between the LLRs before and after quantization [86], i.e.,

$$\{i_k^{\text{opt}}\}_{k=1}^{K-1} = \arg \max_{\{i_k\}_{k=1}^{K-1}} I\left(L_{j,b}; \Lambda_{j,b}\right). \tag{6.5}$$

Following [86], we restrict to even K and symmetric quantizers in which case $\{i_{K/2}^{\text{opt}}\} = 0$ and $i_k = -i_{K-k+1}$.

If the MIMO-BICM system uses quantized LLRs, the final demodulator output is known in advance to belong to the finite set $\{\lambda_1, \ldots, \lambda_K\}$. This fact allows to reduce the size of the candidate list used by the SFSD to refine the initial LLRs.

6.4.1 Quantization-based Pruning

As described above, once d^{ML} has been found at the hard-output FSD stage, the SFSD adds $\log_2 M$ new tree paths to the candidate list. Slightly abusing notation, we denote by $\bar{d}_{j,b}$ the current estimate of the minimum distance associated with the counter-hypothesis for bit b at level j. Every new path starting from level j is intended to update $\bar{d}_{j,b}$ by negating the bth bit. In fact, any refinement of the distances in the candidate list can only modify $\bar{d}_{j,b}$ by a smaller value than the current one. The proposed approach decides whether a certain $\bar{d}_{j,b}$ needs to be updated or not by testing whether the magnitude of the associated LLR satisfies the following condition:

$$|L_{j,b}| = \frac{1}{\sigma^2}\left|d^{\mathrm{ML}} - \bar{d}_{j,b}\right| \leq i^{\mathrm{opt}}_{K/2+1}. \tag{6.6}$$

This condition holds for those LLR values that lie within either $I_{K/2} = [i_{K/2-1}, i_K/2]$ or $I_{K/2+1} = [i_{K/2}, i_K/2+1]$. Since d^{ML} and $\bar{d}_{j,b}$ both are greater than zero and $d^{\mathrm{ML}} \leq \bar{d}_{j,b}$ for all $\{j, b\}$, (6.6) is equivalent to

$$\bar{d}_{j,b} \leq (d^{\mathrm{ML}} + \sigma^2 i^{\mathrm{opt}}_{K/2+1}). \tag{6.7}$$

This condition can be easily tested in the SFSD path extension stage to avoid unnecessary updates of $\bar{d}_{j,b}$.

6.4.2 Clipping-based Pruning

In order to further reduce the number of path extensions in the SFSD, we propose to combine the previously described pruning with LLR clipping. The use of LLR clipping has been widely adopted as a mechanism to reduce the complexity of tree-search detectors [52][54]. LLR clipping imposes a constraint on the dynamic range of the LLRs to enable fixed-point implementations. In our quantization context this reads

$$L_{j,b} = \begin{cases} L_{j,b}, & \text{if} \quad |L_{j,b}| \leq L_{\mathrm{clip}}, \\ L_{\max}\,\mathrm{sign}(L_{j,b}), & \text{else.} \end{cases} \tag{6.8}$$

Here, $L_{\max}$ is the largest admissible LLR value and L_{clip} is the associate LLR clipping threshold. Similar to (6.7), the clipping can be exploited in

the SFSD search to avoid unnecessary updates of $\bar{d}_{j,b}$ whenever the current LLR estimate $L_{j,b}$ satisfies $|L_{j,b}| > L_{\text{clip}}$, which is equivalent to

$$\bar{d}_{j,b} > (d^{\text{ML}} + \sigma^2 L_{\text{clip}}). \tag{6.9}$$

Note that when doing LLR quantization, we have $L_{\max} = \lambda_K = -\lambda_1$ and $L_{\text{clip}} = i_{K-1}^{\text{opt}} = -i_1^{\text{opt}}$.

6.5 Results

We consider a 4×4 MIMO-BICM system with a 16-QAM symbol alphabet. The channel code is a parallel concatenated turbo code with rate=1/2 and a block length of 6144 bits. The code polynomials in octal notation are $[13, 15]$ and rate-1/2 is achieved after puncturing the parity bits of the code. The MIMO channel was i.i.d. block Rayleigh fading (staying constant for 16 MIMO symbols). Fig. 6.4 shows the BER versus SNR for max-log demodulation, the plain SFSD, SFSD with clipping only, and SFSD with 2-bit and 3-bit LLR quantization. For the demodulator using LLR clipping only, we chose $L_{clip} = 8$ as in [52]. For the quantization-based demodulator, the LLR quantizer was designed as proposed in [86].

First of all, it can be observed that the plain SFSD achieves max-log performance. With 3-bit quantization, the BER degradation compared to the non-quantized case is negligible ($\simeq 0.1$ dB). SFSD with 2-bit quantization performs slightly worse ($\simeq 0.5$ dB gap to non-quantized case). Note that quantization-based demodulation inherently performs LLR clipping as well.

Next, we study the complexity savings achieved by the proposed approach in terms of the average number of visited nodes, which is directly related to the number of partial distance calculations in the tree. Fig. 6.5 shows the average number of visited nodes for the SFSD scheme with quantization-based pruning only and with quantization- and clipping-based pruning. The complexity of plain SFSD detection without pruning is included in the figure as a baseline reference. It can be seen that the quantization-based pruning is more significant at lower SNR whereas the

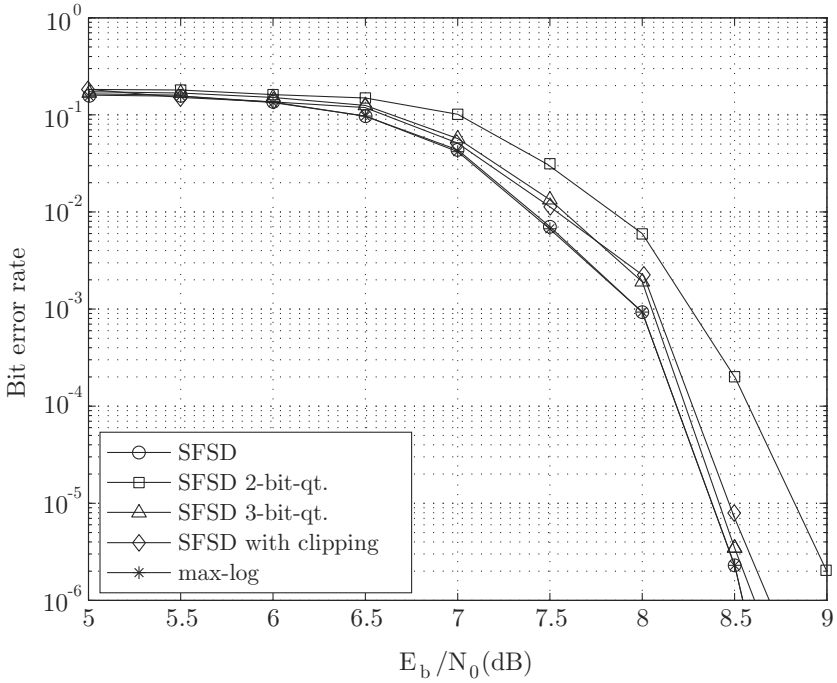


Figure 6.4. BER curves for different SFSD schemes and for max-log demodulation, all with a rate-1/2 turbo code in a 4×4 MIMO-BICM system using 16-QAM.

complexity reduction due to clipping-based pruning increases as the SNR grows. Moreover, as expected, 2-bit quantization results in larger complexity savings than 3-bit quantization. Clearly, the complexity savings achieved by the combined pruning strategy are much larger than those achieved by either of the two pruning techniques alone, with complexity reductions of 37% and 31% for 2-bit- and 3-bit quantization, respectively, at an SNR of 9 dB. Since quantization-based pruning yields more pronounced savings at low SNR and clipping-based pruning yields larger savings at high SNR, their combination provides a more uniform complexity over the SNR range considered.

Furthermore, the average number of candidates contained in the final list of each SFSD scheme was also recorded during the soft detection stage.

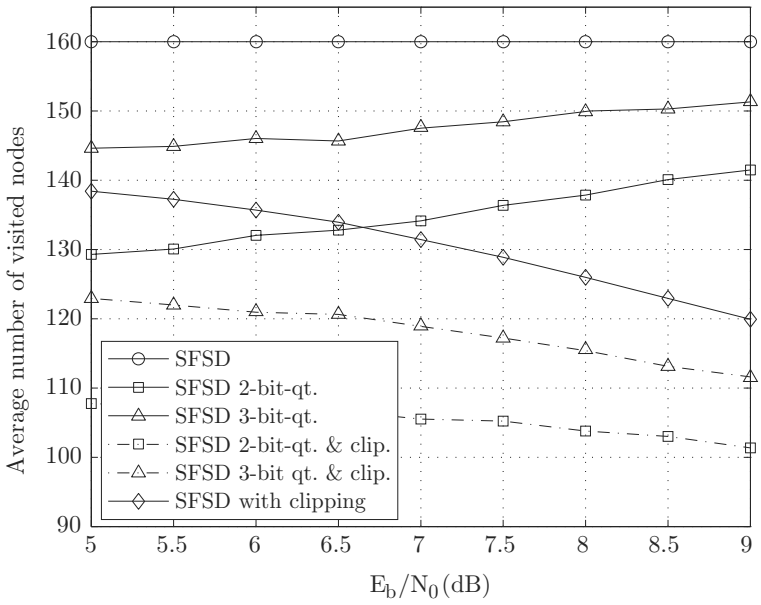


Figure 6.5. Average number of nodes visited by the different SFSD schemes in a 4×4 MIMO-BICM system using 16-QAM.

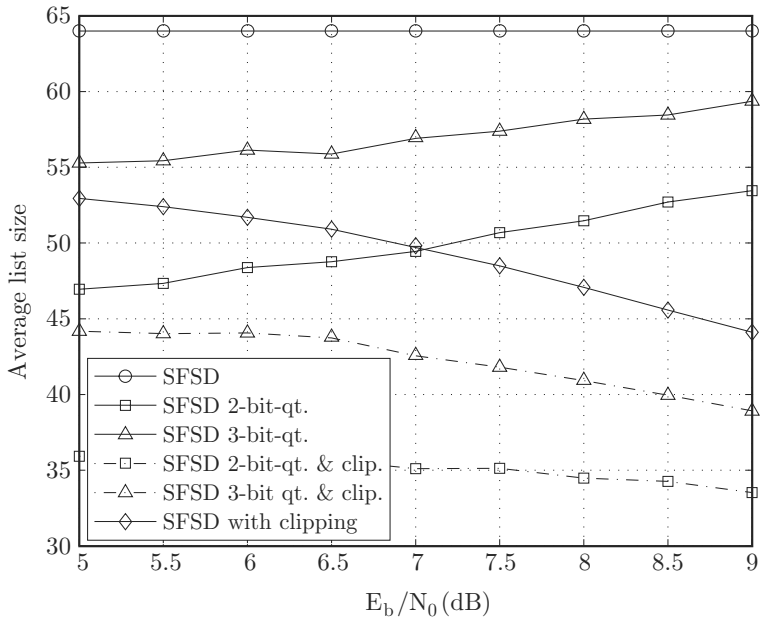


Figure 6.6. Average list size of the different SFSD schemes under study in a 4×4 MIMO-BICM system using 16-QAM symbols.

Note that this parameter gives an idea about the storage requirements, the necessary memory accesses and also the complexity to sort the list and to subsequently search for the minimum paths. Fig. 6.6 shows the average list size for the SFSD scheme with LLR quantization pruning, with and without clipping. The upper bound for this parameter is also included. It can be seen that both the list size reduction due to quantization-based pruning and the one due to clipping follow the same tendency as the average number of visited nodes reduction shown in Fig. 6.5. Hence, the 2-bit quantization strategy decreases the average list size more than the 3-bit-quantization and, when clipping is included additionally to quantization, further reduction is achieved than those of the independent pruning techniques. The percentages of list size reduction reach 39% for 2-bit-quantization and 47% for 3-bit-quantization, both for the highest E_b/N_0 in the considered range.

6.6 Conclusion

In this chapter we considered MIMO-BICM systems working with quantized soft information. Some guidelines on how to embed the LLR quantization stage into the demodulator to directly deliver quantized soft outputs were given. As an example, the well-known SFSD scheme was modified to take into account the subsequent quantization stage in order to prune in advance those tree-paths useless for LLR refinement. In addition, LLR clipping was considered and efficiently combined with quantization-based pruning to further reduce the average number of visited nodes in the SFSD tree-search.

Several performance results were obtained for the typically used quantization lengths: 2 and 3 bits. The complexity reduction was assessed by measuring the average number of visited nodes. Furthermore, the average candidates list size was obtained in order to assess the storage requirements, the necessary memory accesses and also the complexity to sort the list and to subsequently search for the minimum paths.

Results showed that while the complexity reduction due to quantization was mainly important at low SNR, the complexity reduction due to clipping

increased at higher SNR. This complementary behavior justifies the combination of both pruning techniques to provide nearly uniform complexity reduction along the SNR range without noticeably degrading the results after quantization.

The efficient SFSD with quantized outputs is currently under consideration in [89].

7 GPU Implementation of MIMO Detectors

7 GPU Implementation of MIMO Detectors

The use of many-core processors such as general purpose Graphic Processing Units (GPU) has recently become attractive for the efficient implementation of signal processing algorithms for communication systems. This is due to the cost-effectiveness of GPU together with their potential capability of parallel processing. This chapter presents efficient GPU implementations of several hard- and soft-output detection schemes, which allow to considerably decrease the computational time required for the data detection stage in MIMO systems. Moreover, a fully-parallel soft-output scheme with a GPU-aware preprocessing stage is developed. All the implementations are evaluated for different system configurations: changing the constellation used, the system size or the number of subcarriers. In addition, the computational times of the proposed GPU implementations are compared with their execution times on a high performance CPU. Furthermore, the throughputs of all the algorithms are computed, showing to outperform other recent implementations while ensuring nearly-optimal detection performance.

7.1 Introduction

As claimed throughout this thesis, the use of MIMO communication systems complicates the receiver stage, which has the task of processing the received mixture of signals affected by the channel in order to recover the transmitted data with the highest reliability. If nearly optimal detection is desired, this stage becomes often the most computationally expensive within a MIMO system and, thus, the search for high-throughput receiver implementations is imperative. Furthermore, scalability in the number of subcarriers per MIMO-symbol and in the system size are key factors in LTE and 4G wireless standards [3].

Recently, the use of many-core processors such as GPU has become attractive for the efficient implementation of signal processing algorithms with high computation requirements [90][91]. Although multi-core CPU implementation could also replace the traditional use of DSP and field programmable gate arrays (FPGA), this option would interfere with the execution of the tasks assigned to the CPU of the computer, possibly causing speed decrease. Therefore, since the GPU is more rarely used than the CPU in conventional applications, its use as a co-processor in signal processing systems is very promising. Several recent works show the growing interest of GPU for the development of reconfigurable software-defined-radio platforms [92][93]. Signal processing for MIMO wireless communication systems is indeed a field which requires high computation capabilities, thus, GPU are becoming very useful to implement high-throughput schemes such as the MIMO detectors proposed in [20][94][95] or the fast decoding schemes for LDPC codes presented in [19] and [96]. This is due to the cost-effectiveness of GPU together with their huge capability of parallel processing.

Since most of the existing MIMO detection algorithms were not specifically designed to exploit the GPU resources efficiently, the algorithms that best match a certain GPU architecture must be carefully selected and, if necessary, be improved before facing implementation. Note that the latter task is not straightforward at all.

In this chapter, the different parts that constitute GPU are described together with some general rules on how to best exploit their capabilities.

After this, we propose GPU implementations for several schemes based on the FSD proposed in [46], which ensure nearly optimal detection performance in MIMO-BICM systems with high throughput. Both hard-output and soft-output algorithms are implemented. In addition, a fully parallel soft-output scheme is proposed with a GPU-aware designed preprocessing stage. Focusing on real-time applications, the execution times of the algorithms are also evaluated and compared to those of some other recent implementations in the literature. Furthermore, the achieved throughputs are compared with the minimum one required by current wireless standards and the transmission configurations supported by the proposed GPU implementations are discussed.

The chapter is structured as follows. Section 7.2 gives an overview of compute unified device achitecture (CUDA) programming and GPU architectures and describes the features of the GPU employed for the implementations. In Section 7.3, the motivation for choosing the algorithms selected in this thesis is described. In addition, a fully parallel soft detection algorithm, which is suitable for GPU implementation, is proposed. Section 7.5 addresses the configuration parameters and performance measures considered for the evaluation of the proposed GPU implementations. Finally, Section 7.6 contains the conclusions of this chapter.

7.2 GPU and CUDA

Compute Unified Device Architecture (CUDA) [97] is a software programming model that exploits the massive computation potential offered by GPU. In what follows, the CUDA environment and many insights about GPU architectures are described.

7.2.1 CUDA Programming Model

CUDA technology is an environment based on C language which allows the development of software intended to solve high complexity computational problems efficiently. This software takes profit from the high amount of execution threads which are available in GPU. CUDA programming is per-

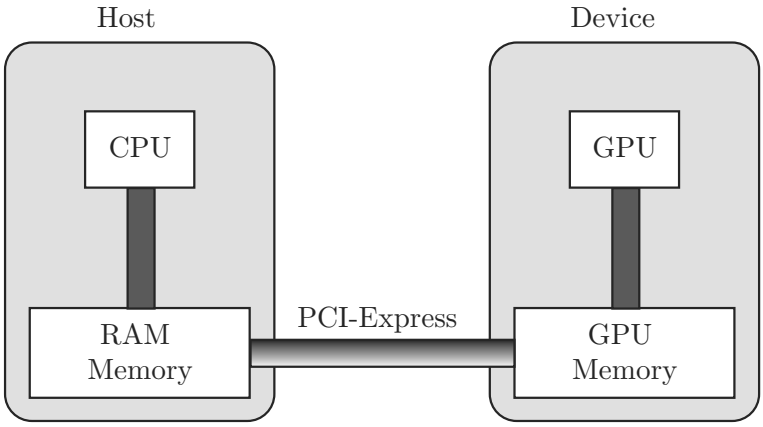


Figure 7.1. Interconnection between host and device in the programming model.

formed in the two different parts of the system shown in Fig. 7.1: the *host*, formed by one or more CPU and their corresponding RAM memory and the *device*, where one or more GPU and GPU memory are located. Host and device communicate through the PCI express bus. The time required for this communication is sometimes the bottleneck of implementations, thus, it is advisable to minimize data transfers between host and device [98].

According to Flynn's taxonomy [99], from a conceptual point of view, a GPU can be considered as a *single instruction, multiple data* (SIMD) machine; that is, a device in which a single set of instructions is executed on different data sets. Fragments of code that are executed by many threads in the device are called *kernels*, which can be only launched by the host. The number of threads that execute the parallel code (or kernel code) depends on the amount of data, the data type and the operation to be executed. Nevertheless, the kernel only contains the code to be executed by one thread and all the threads assigned to it will execute the same code. The logical structure of a CUDA program is shown in Fig. 7.2. First of all, the data to be processed is loaded in CPU RAM memory. Next, data are transferred to GPU memory and the computations are carried out by the GPU kernel. Once the device ends, results are transferred back to the CPU.

Threads are grouped in blocks of usually up to 512 threads (depending

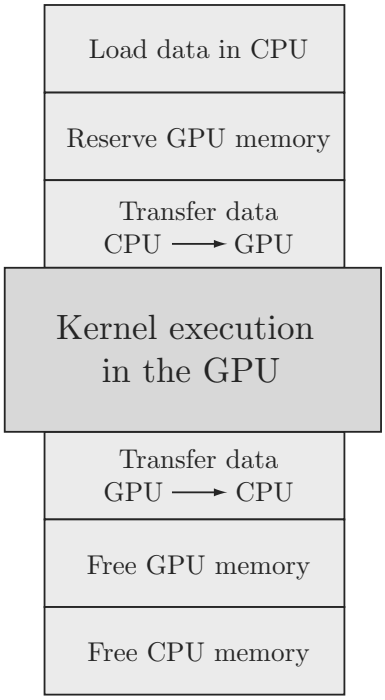


Figure 7.2. CUDA programming model.

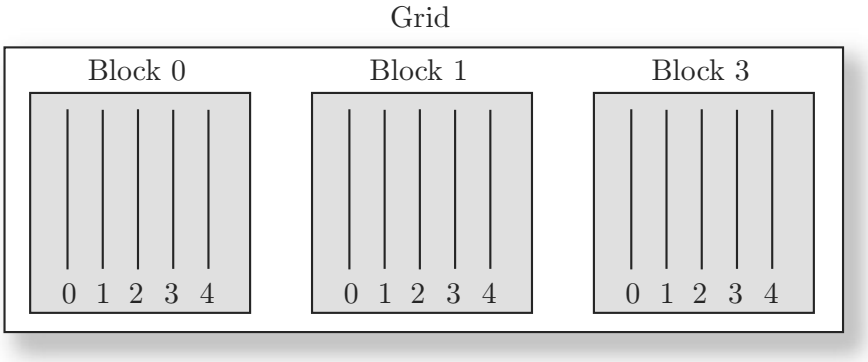


Figure 7.3. Position of threads inside a unidimensional grid with 3 blocks and 5 threads per block.

on the GPU) and threads within a block can communicate through a special memory called *shared memory*. In the same way, thread blocks are grouped to form a *grid*. The grid organization also depends on the data type being used. Each thread has a unique identifier within a block, which, at the same time, has a unique identifier within a grid. Blocks and grids can be unidimensional, bidimensional or tridimensional. For instance, Fig. 7.3 shows a unidimensional grid with three blocks, each of them with 5 threads. All this multilevel thread organization is intended to lead the programmer towards getting maximum GPU performance. Therefore, before launching a CUDA execution, the number of threads per block and their distribution together with the number of blocks per grid and their distribution must be carefully selected.

Each block to be executed is partitioned into warps, which are groups of 32 threads. The block size must be a multiple of 32, otherwise the last warp of the block will have some remaining threads which occupy hardware but do not perform any useful work.

In the CUDA model, although each thread executes the kernel independently, threads within a block can synchronize through a barrier and write simultaneously to shared memory to share data between them. In contrast, thread blocks are completely independent among themselves and can only share data through the global memory once the kernel ends.

Table 7.1. Nvidia Tesla C2070 features.

Number of stream multiprocessors	14
Number of cores	448
Clock rate	1.15 GHz
Global memory	4 GB
Constant memory	64 kB
Shared memory per block	48 kB

7.2.2 GPU Architecture

The first-proposed GPU architecture, also known as Tesla architecture [100], was based on a unit called *Stream Multiprocessor* (SM), which characterizes most of the Tesla and GeForce NVIDIA GPU models. In those GPU versions supporting CUDA computing capabilities between 1.0 and 1.3 [97], each SM consisted of 8 scalar single-precision processors, also called CUDA cores, one double-precision processor, 16 kB of shared memory and 16384 registers.

There is a wide variety of GPU models but the recently appeared Fermi architecture [101] deserves important attention. This architecture includes CUDA capabilities 2.0 or 2.1, where the number of cores per SM was increased to 32 [97]. Furthermore, the maximum parallelism level is guaranteed by overlapping the execution of several kernels, overlapping data copy and kernel execution, transferring data from host to device and from device to host simultaneously, etc.

In this thesis, we employed for the implementations the NVIDIA Tesla C2070 GPU with 2.0 CUDA capability. Its specifications can be seen in Table 7.1. The installed CUDA toolkit and SDK version is 4.0 [97]. Note that asynchronous memory copy is supported to overlap data transfer and kernel execution.

7.3 Algorithms Selected for GPU Implementation

In [102], the impact of GPU and multi-core CPU in signal processing was evaluated quantitatively through testing several implementations of signal processing algorithms (benchmarks) in different platforms. One of the conclusions extracted therein was that the performance of GPU is highly dependent on the problem to be solved. In fact, those algorithms with certain dependency among intermediate results can hardly benefit from the potential of GPU. In this section, the selection of those MIMO detectors that can best fit within the GPU architecture is addressed. In addition, a novel soft-output scheme with fully parallel structure is proposed.

7.3.1 FSD versus K-Best

In [102], one of the selected benchmarks was the K-Best tree-search detector, which was described in Section 2.3.3 and has been employed throughout this dissertation many times. The K-Best approach performs the necessary calculations to expand a number of $N = K \cdot M$ tree-nodes per level. These calculations include the branch weight, the accumulated PED update and the selection of the best K paths to proceed the tree-search in the next level. Note that the node expansions of all the surviving paths at the same level can be carried out in parallel and are the same for all levels. However, the results in one level affect the calculations in the following level, thus, parallelism among levels cannot be exploited.

In the GPU implementation proposed in [102], the PED calculation of each tree-branch was assigned to a different thread. When all the threads finished, the results associated to the N nodes and their distances were sent to the CPU, which obtained the K minimum distances. Unfortunately, the inherent dependency between the tree levels required mandatory barrier synchronization among threads. This dependency also made impossible to employ shared memory (much faster than the global one) to store intermediate calculations and to get the K survivors in the kernel, since the shared memory is only accessible by a certain processor. After all, the GPU parallelization was only worth for the PEDs calculation and not for the sorting and selection of the K-best survivors, which had to be carried out sequen-

tially. In the end, the total computational times of the algorithm were very similar in the CPU and GPU implementations, which showed that the K-Best method was not an appropriate algorithm for GPU implementation.

The above described limitations of the K-Best tree-search detector led us to investigate the parallel processing capabilities of the hard-output FSD in [46]. As described in Section 2.3.5, in the hard-output FSD, all the SIC branches are totally independent among them and this feature makes the algorithm very suitable for GPU implementation, as will be shown later. Hence, the hard-output FSD was one of the algorithms selected for GPU implementation. In addition, we implemented in GPU the soft-output version of the FSD (SFSD) [55], which was presented in Section 6.3. The motivation for this selection was twofold. On the one hand, the first stage of the SFSD method is based on the FSD and, thus, it is suitable for parallel implementation. On the other hand, the method exhibits fixed complexity, which leads to a predictable and constant execution time.

Finally, we designed a GPU-aware soft-output detector to further decrease the computational time needed to carry out this task. The proposed approach is described next.

7.3.2 Proposed Soft-Output Detection Scheme

Although the SFSD method proposes a smart list extension based on the lowest distance paths in the initial list, this extension cannot be performed at the same time as the FSD hard-output stage since the paths to be extended are known for the first time when the FSD stage ends. Therefore, the parallelism degree that can be exploited by the SFSD varies among its stages. For this reason, we propose an alternative soft-output FSD with a fully parallel structure: the fully-parallel FSD (FPFSD).

The proposed soft-output approach is purely based on the hard-output FSD scheme. The list of candidates and distances necessary to obtain soft information is calculated by means of n_T hard-output FSD searches, each with a different channel matrix ordering. The n_T different channel orderings ensure that a different layer (level) of the system is placed at the top of the tree each time. This way, candidate paths containing all the bit labelling

possibilities in every level are guaranteed and, thus, soft information about all the bit positions is always available. Fig. 7.4 shows the search-tree of the FPFSD for the case with $n_T = 4$ and QPSK symbols. Note that the $n_T = 4$ hard-output FSD searches are totally independent and can be carried out in parallel.

In [46], a special ordering was proposed to place the detection layers associated to the less reliable received symbols at the top of the tree. This way, full expansion was performed for those symbols to make the solution independent of the decision in these levels. The rest of symbols were detected from the most reliable one to the less. This ordering strategy, although showing good performance, involves the calculation of a pseudo-inverse matrix with cost $\mathcal{O}(n_T^3)$ at each of the n_T iterations, leading to a complexity of $\mathcal{O}(n_T^4)$. On the other hand, the calculations to be carried out for each iteration are not independent among them, thus, no parallelism can be exploited at such preprocessing stage. Due to the just-described reasons, the preprocessing stage proposed in [46] was discarded for our parallel implementation.

In this work we propose the use of a much simpler ordering strategy which is fully parallel and easy to be implemented in either multi-core or GPU. First of all, the norms of the columns of the channel matrix are obtained (requiring n_T products, $n_T - 1$ sums and one squared root operation each) and sorted in ascending order (n_T^2 flops in the worst-case). Thus, the complexity of this proposed ordering is $\mathcal{O}(n_T^2)$. Note that this can be computed considerably faster if the norms are processed in parallel. Generally, this ordering leads to more reliable decisions than a random ordering, since symbols with the highest signal-to-noise ratio are detected before those with the lowest, thus reducing error propagation. Once the norm-based ordering is available, the n_T orderings needed by the FPFSD are directly built based on this initial norm-based ordering. Taking into account the above explained requirements, the first level of each new ordering is assigned a different detection position in order to guarantee the availability of soft information for all the possible bit values in every system level. Note that, as when using the FSD ordering, the reliability of the symbol placed in the FE stage is irrelevant. Then, the remaining levels are ordered following the

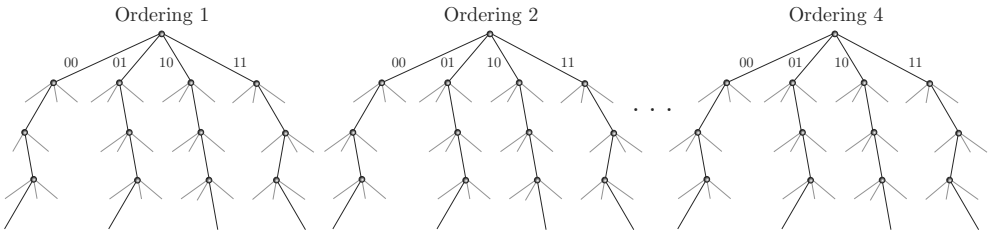


Figure 7.4. Decoding trees of the FPFSD algorithm for a 4×4 MIMO system with QPSK symbols.

Table 7.2. Symbol detection position and corresponding tree-level in the initial system for the involved FPFSD orderings in an example with $n_T = 4$.

Detection position	Norm-based ordering	Ordering #1	Ordering #2	Ordering #3	Ordering #4
1st	2	1	2	3	4
2nd	4	2	4	2	2
3rd	3	4	3	4	3
4th	1	3	1	1	1

initial column-norm-based ordering but skipping the level that was already set on the top. The example in Table 7.2 shows how the ordering is set up for a particular column-norm-based ordering of a 4×4 channel, which in this case is $\{2, 4, 3, 1\}$. As the first row of Table 7.2 shows, the ith proposed ordering starts the data detection at the ith tree-level, being $i \in \{1, 2, 3, 4\}$. Then, the remaining levels are explored following the column-norm-based ordering in column 2.

The GPU implementation of the FPFSD method together with the FSD and SFSD implementations are addressed in the next section.

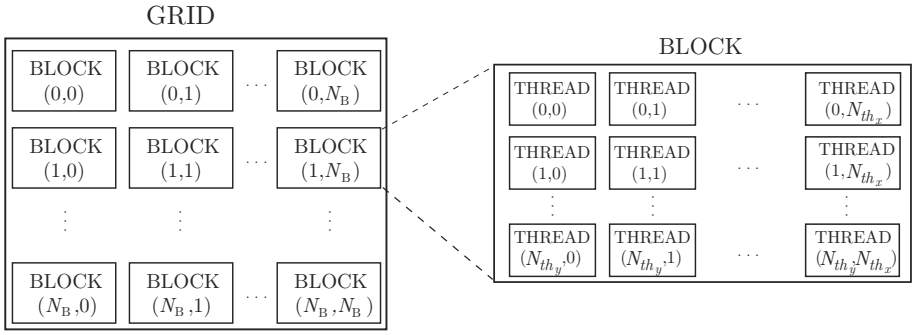


Figure 7.5. Grid and block distributions considered for the preprocessing kernel of the hard- and soft-output FSD implementations.

7.4 Implementation of MIMO Detection Algorithms in CUDA

This section describes the grid configuration, block size and number of kernels that were chosen for the GPU implementations developed in this thesis. In addition, detailed information about the kernels structures is also given.

Fig. 7.5 shows the bidimensional grid considered for the algorithms, where the number of blocks per dimension is denoted by N_B and, similarly, the numbers of threads per block dimension are denoted by N_{th_x} and N_{th_y}, respectively. The block size $N_{th_x} \times N_{th_y}$ will be chosen to be a multiple of 32 in order to avoid incomplete warps. Moreover, the most suitable block size for each configuration, which includes the value of N_c, n_T and M, will be selected to minimize the execution time. Asynchronous memory copy is supported to overlap data transfer and kernel execution, this way, the necessary time to transfer results back to the CPU can be disregarded.

The particular parameters of the grids together with the kernels information are described for each specific algorithm subsequently.

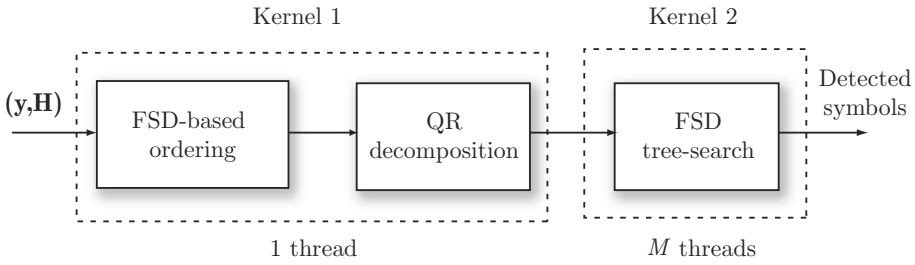


Figure 7.6. Block diagram of the proposed FSD GPU implementation including the number of threads which execute each kernel for each subcarrier.

7.4.1 Hard-Output FSD

The FSD GPU implementation is formed by the two kernels shown in Fig. 7.6. The first kernel is in charge of the preprocessing stage and the second one attains the tree-search.

In kernel 1, each thread calculates the ordering of a different channel matrix and, afterwards, the QR decomposition of the reordered channel matrix is computed by the same thread. Note that the channel matrices to be decomposed are quite small (4×4), thus, multi-thread execution of each QR factorization is not considered. As shown in [103], its multi-core execution is advised for bigger problem sizes (1000×1000 and above). Thus, only one thread per subcarrier is needed and the number of subcarriers that can be processed by a single block is directly

$$N_{cb} = (N_{th_x} \cdot N_{th_y}). \tag{7.1}$$

Once N_{th_x} and N_{th_y} are selected, the value of N_B is chosen as:

$$N_B = \left\lceil \sqrt{\frac{N_c}{(N_{th_x} \cdot N_{th_y})}} \right\rceil. \tag{7.2}$$

As it was described in Section 2.3.5, the FSD ordering requires the computation of a pseudoinverse matrix in every step. This calculation can be performed more efficiently by solving the two linear systems that are next presented.

The pseudoinverse matrix to be computed at step i is

$$\mathbf{G} = \mathbf{H}_i^{+} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H. \tag{7.3}$$

Calling $\widetilde{\mathbf{H}}_i = (\mathbf{H}_i^H \mathbf{H}_i)$ and with $\widetilde{\mathbf{H}}_i = \widetilde{\mathbf{Q}}_i \widetilde{\mathbf{R}}_i$, if (7.3) is multiplied by $\widetilde{\mathbf{H}}_i$, it can be equivalently expressed as:

$$\widetilde{\mathbf{H}}_i \mathbf{G} = \widetilde{\mathbf{Q}}_i \widetilde{\mathbf{R}}_i \mathbf{G} = \mathbf{H}_i^H. \tag{7.4}$$

Next, (7.5) can be multiplied by $\widetilde{\mathbf{Q}}_i^H$ resulting in

$$\widetilde{\mathbf{Q}}_i^H \widetilde{\mathbf{Q}}_i \widetilde{\mathbf{R}}_i \mathbf{G} = \widetilde{\mathbf{R}}_i \mathbf{G} = \widetilde{\mathbf{Q}}_i^H \mathbf{H}_i^H, \tag{7.5}$$

which can be solved following the next described two steps. First, an auxiliary matrix $\mathbf{J}_i$ is computed as:

$$\mathbf{J}_i = \widetilde{\mathbf{Q}}_i^H \mathbf{H}_i^H, \tag{7.6}$$

then, solving the following upper triangular system gives matrix $\mathbf{G}$ as a result:

$$\widetilde{\mathbf{R}}_i \mathbf{G} = \mathbf{J}_i. \tag{7.7}$$

The steps carried out by each thread within a block for the first kernel of the FSD are detailed in Algorithm 6, where the permutation matrix $\mathbf{P}$ is built and the QR factorization matrices of the reordered channel matrix are computed and stored in global memory.

In kernel 2, the M independent branches of the hard-output FSD scheme of all the subcarriers within the same group are simultaneously executed by different threads. Since M threads are needed for the tree-search, the number of subcarriers per block is given by

$$N_{cb} = \frac{(N_{th_x} \cdot N_{th_y})}{M}, \tag{7.8}$$

and the value of N_B is chosen as:

$$N_B = \left\lceil \sqrt{\frac{(M \cdot N_c)}{(N_{th_x} \cdot N_{th_y})}} \right\rceil. \tag{7.9}$$

Algorithm 6 Calculations carried out by the qth thread at the FSD preprocessing stage

Get $\mathbf{H}$ associated to the current subcarrier from global memory
$\Delta = \{1, 2, \cdots, n_T\}$
$dis_{1:n_T} = [1, 1, \cdots, M]$
$\mathbf{P} = zeros(n_T, n_T)$
for $i = n_T, \ldots, 1$ **do**
  $\mathbf{H}^{(i)} = \mathbf{H}_{:,\Delta}$
  Obtain $\mathbf{G}$ solving Eqs. (7.6) and (7.7)
  $d_{max} = 0$
  $d_{min} = 1e6$
  for $j = 1, \ldots, length(\Delta)$ **do**
    $norm_j = \|G_{j,:}\|^2$
    if $dis_i == M$ and $norm_j > d_{max}$ **then**
      $d_{max} = norm_j$
      $k = \Delta_j$
    else if $dis_i = M$ and $norm_j < d_{min}$ **then**
      $d_{min} = norm_j$
      $k = \Delta_j$
    end if
  end for
  $P_{k,i} = 1$
  $\Delta = \Delta - \{\Delta_j\}$
end for
Permute columns of matrix $\mathbf{H}$ with $\mathbf{P}$
$\mathbf{H} = \mathbf{QR}$

Algorithm 7 shows the operations carried out by each thread within a block of kernel 2, where q goes over the threads in the block and, hence, it ranges between 1 and $N_{th_x} \cdot N_{th_y}$. On the other hand, i ranges between 1 and M to go over the set of constellation symbols. Note that the referred equations are not here included because they were already well detailed in Section 2.3.5).

7.4.2 Soft-Output FSD

The SFSD scheme was described in Section 6.3. This method starts from the list of candidates that the hard-output FSD gets and extends this list

Algorithm 7 Calculation of one of the branches of the hard-output FSD by the qth thread.

1: Get one of the values of Ω (denoted by Ω_i) from constant memory,
2: Assign $s_{n_T,q} = \Omega_i$ and store it in shared memory,
3: Get $\mathbf{R}$ and $\mathbf{y}'$ associated to the current subcarrier from shared memory,
4: Compute the PED $d_{n_T}(S_q^{(n_T)})$ with equations (2.26-2.25) and store it in shared memory,
5: **for** $k = n_T - 1, \ldots, 1$ **do**
6: Compute the kth symbol using SIC (2.19),
7: Update path distance $d_k(S_q^{(k)})$,
8: **end for**
9: Store the whole path and all the distances in shared memory,
10: Sync barriers

to provide more information about the counter bits of the ML solution. The two main stages of the SFSD scheme are shown in Fig. 6.3.

The proposed GPU implementation is structured in the two kernels shown in Fig. 7.7. The first kernel coincides with the one set for the FSD, thus, its description is again given by Algorithm 7. The second kernel, however, includes two more stages intended to extend the list and compute the LLRs, respectively.

In the SFSD GPU implementation, after the hard-output part is finished, the first thread of each subcarrier calculates the N_{iter} minimum distances. Then, the first kernel ends and the indices of the N_{iter} best paths are stored in shared memory. In order to obtain the $N_{iter} \cdot \log_2 M \cdot (n_T - 1) \cdot N_{cb}$ new candidates in kernel 2, all the calculations are equally distributed among the threads of the block. Then, each thread calculates every new candidate path as shown in Algorithm 8. After this, a list with $P = M + N_{iter} \cdot \log_2 M \cdot (n_T - 1)$ paths and distances is stored in shared memory. Thus, the last part of kernel 2 finds within this list the minimum distances of paths having the counter bits and computes the $\log_2 M \cdot n_T$ LLRs.

The steps carried out by each thread in the last stage of the second kernel are described in Algorithm 9.

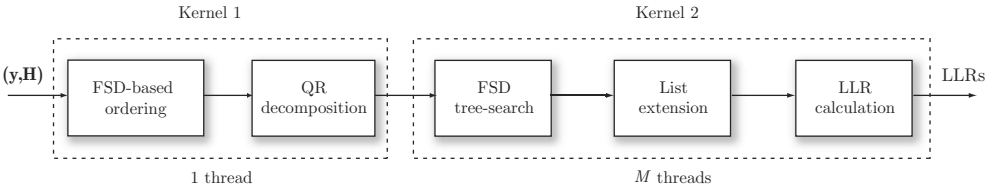


Figure 7.7. Block diagram of the proposed SFSD GPU implementation including the number of threads which execute each kernel for each subcarrier.

Algorithm 8 Calculation of new candidates for the soft-output FSD by the qth thread.

1: Get level (l), bit position (b) and selected path (N_{it}) to be extended,
2: Copy symbols from level l to n_T from shared memory into $s_{l+1:n_T,q}$,
3: Negate bth bit, keep the rest of bits as in the N_{it}-path symbol and assign the associated symbol to $s_{l,q}$,
4: Compute the PED $d_l(S_q^{(l)})$ with equations (2.25-2.26) and store it in shared memory,
5: **if** $l > 1$ **then**
6: **for** $k = l-1, \ldots, 1$ **do**
7: Compute the kth symbol using SIC (2.19),
8: Update path distance $d_k(S_q^{(k)})$,
9: **end for**
10: **end if**
11: Store the new path and its accumulated distances in shared memory,
12: Sync barriers

Algorithm 9 Computation of the LLR for the bth bit of the lth level by the qth thread.

1: $d_{min} = 1e6$,
2: **for** $k = 1, \ldots, P$ **do**
3: **if** ($d_1(S_k^{(1)}) < d_{min}$) and (bth bit of $s_{l,k}$ is equal to $\overline{x_{l,b}^{ML}}$) **then**
4: $d_{min} = d_1^{S_k^{(1)}}$
5: **end if**
6: **end for**
7: $L(x_{l,b}) = (d^{ML} - d_{min})(1 - 2x_{l,b}^{ML})/\sigma$

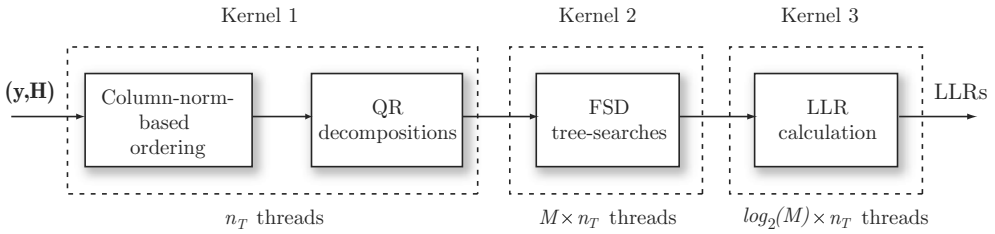


Figure 7.8. Block diagram of the proposed FPFSD GPU implementation including the number of threads which execute each kernel for each subcarrier.

7.4.3 Fully-Parallel SFSD

The proposed GPU implementation of the FPFSD is composed of three consecutive kernels that are executed over the bidimensional grid configuration depicted in Fig. 7.5. Since the parallelism degree that can be exploited within the different stages of the FPFSD is not always the same, the proposed GPU implementation consists of three differentiated kernels, shown in Fig. 7.8. The number of threads associated to each subcarrier within a certain block, which differs among kernels, is also included in the figure. Note that the first kernel is in charge of the preprocessing stage, composed by the proposed channel matrix orderings together with the corresponding QR decompositions. The second kernel performs the n_T FSD tree-searches. And, finally, the third kernel calculates the LLR values.

In the first kernel, each thread calculates one column-norm and only one out of n_T threads sorts the n_T norms. After this, each thread computes one of the necessary n_T QR decompositions. Therefore, n_T kernels per subcarrier are needed and, hence, the number of processed subcarriers per block is given by

$$N_{cb} = \left\lceil \frac{(N_{th_x} \cdot N_{th_y})}{n_T} \right\rceil, \tag{7.10}$$

and the value of N_B for the first kernel is:

$$N_B = \left\lceil \sqrt{\frac{(n_T \cdot N_c)}{(N_{th_x} \cdot N_{th_y})}} \right\rceil. \tag{7.11}$$

Algorithm 10 Calculations carried out by the qth thread at the norm-based preprocessing stage.

1: $norm_q = \|\mathbf{H}_{:,q}\|$,
2: Sync barriers
3: **if** $q == 1$ **then**
4: Sort n_T norms
5: **end if**
6: Sync barriers
7: $\mathbf{H}^{(q)} = \mathbf{Q}^{(q)}\mathbf{R}^{(q)}$

The tasks carried out by each thread of kernel 1 are detailed in Algorithm 10.

After computing the QR factorizations of the reordered channel matrices of all the subcarriers, these are copied into the GPU global memory to be used by the second kernel.

The second kernel must perform n_T independent FSD stages, thus, the $M \cdot n_T$ independent SIC problems for all the subcarriers are simultaneously executed by different threads. Hence, the number of processed subcarriers per block for the second kernel of the FPFSD is

$$N_{cb} = \left\lceil \frac{(N_{th_x} \cdot N_{th_y})}{(M \cdot n_T)} \right\rceil, \tag{7.12}$$

and the following value of N_B is selected:

$$N_B = \left\lceil \sqrt{\frac{(M \cdot n_T \cdot N_c)}{(N_{th_x} \cdot N_{th_y})}} \right\rceil. \tag{7.13}$$

The operations carried out by each thread within a block at the second kernel are those already addressed in Algorithm 7.

Once the n_T FSD stages have been performed, the second kernel ends and a list with $P = M \cdot n_T$ paths and distances is stored in global memory to allow its access from the third kernel. The third kernel has to find the minimum values of $d_{min}^{(0)}$ and $d_{min}^{(1)}$ associated to each LLR calculation. This task is carried out by $\log_2 M \cdot n_T \cdot N_{cb}$ threads working in parallel, thus, the following values of N_{cb} and N_B are selected for this kernel:

Algorithm 11 Computation of the LLR for the bth bit of the lth level by the qth thread.

1: $d_{min}^{(0)} = 1e6$, $d_{min}^{(1)} = 1e6$,
2: **for** $k = 1, \ldots, P$ **do**
3: **if** $(d_1(S_k^{(1)}) < d_{min}^{(0)})$ and (bth bit of $s_{l,k}$ is equal to 0) **then**
4: $d_{min}^{(0)} = d_1(S_k^{(1)})$,
5: **end if**
6: **if** $(d_1(S_k^{(1)}) < d_{min}^{(1)})$ and (bth bit of $s_{l,k}$ is equal to 1) **then**
7: $d_{min}^{(1)} = d_1(S_k^{(1)})$,
8: **end if**
9: **end for**
10: $L(x_{l,b}) = (d_{min}^{(0)} - d_{min}^{(1)})/\sigma$

$$N_{cb} = \left\lceil \frac{(N_{th_x} \times N_{th_y})}{(\log_2 M \cdot n_T))} \right\rceil, \tag{7.14}$$

$$N_B = \left\lceil \sqrt{\frac{(\log_2 M \cdot n_T \cdot N_c)}{(N_{th_x} \cdot N_{th_y})}} \right\rceil. \tag{7.15}$$

The steps of the third kernel of the FPFSD are detailed in Algorithm 11.

7.5 Results

7.5.1 Configuration Parameters and Performance Measures

We considered 2×2 and 4×4 MIMO systems with QPSK, 16-QAM and 64-QAM symbol alphabets. The values of $N_{th_x} \times N_{th_y}$ were selected to minimize the execution time. We considered the LTE standard specifications, where a 0.5 ms time slot is composed of 7 MIMO-OFDM symbols plus their respective cyclic prefixes [57]. Furthermore, the performances with the different N_c values reported in the LTE standard, i.e. $N_c = \{150, 300, 600, 900, 1200\}$, are investigated. In the proposed implementation all the symbols in the same time slot are detected simultaneously.

As described in LTE Release 10 [3], transmission bandwidths up to 100 MHz can be employed by means of the aggregation of up to five component carriers as the ones in Release 8. Thus, the transmission can be done over a maximum of $5 \times 1200 = 6000$ subcarriers. The latter values of N_c were also considered for some of the results.

Apart from the BER performance, which has been evaluated through all this thesis, two different performance measures were considered to evaluate the proposed implementations:

- *Speedup*, which is defined as the ratio between the computational time resulting of executing the algorithms sequentially on a single core of a high-performance CPU and the time to execute the same algorithms on the GPU. This calculation shows how advantageous the use of GPU is to process large amounts of data compared to CPU. The selected CPU is an Intel Xeon X5680 hexacore processor at 3.33 GHz with 96 GB of DDR3 main memory. It has 12 MB of cache memory and, with the hyperthreading technology, each core has 24 virtual processors. Note that, although this is not a fair comparison (since only one core of the CPU is being considered), this parameter gives an idea of the level of parallelism that the GPU is exploiting. Moreover, the GPU can always act as a co-processor and leave CPU resources free for other tasks carried out in the MIMO system. Hence, this comparison is not intended to detract the use of the CPU, but to assess the parallel processing capabilities of the GPU.

- *Throughput*, which is defined as the number of processed information bits per second. Note that this parameter was previously used to assess other implementations on GPU such as the ones in [95] and [94], thus, it allows a fair comparison among implementations. Moreover, the throughput evaluation exposes whether a given implementation guarantees the real-time requirements of a certain wireless standard.

Table 7.3. Optimal block size for the preprocessing part of the Hard-output FSD implementation.

N_c	QPSK	16-QAM	64-QAM
300	4×4	16×16	4×64
1200	16×4	16×16	2×64
6000	32×4	8×16	2×64

7.5.2 Hard-Output FSD

Speedup

The parallel implementation of the hard-output FSD on GPU was compared to the implementation in the above-mentioned high-performance CPU. The values of $N_{th_x} \times N_{th_y}$ that minimize the execution time for the two considered system sizes and three representative values of N_c are collected in Table 7.3.

Fig. 7.9(a) shows the speedup for a 2×2 system and the three considered constellations as a function of the number of subcarriers. It can be observed that, generally, the higher the number of subcarriers and constellation size, the higher the achieved speedup. Next, the speedup of the 4×4 system was evaluated and depicted in Fig. 7.9(b). The comparison between Figs. 7.9(a) and (b) reveals that higher speedup is achieved when the system size increases. Therefore, as the computational saving increases as the problem size does, the use of GPU is very promising for high dimensional constellations and/or LTE configurations dealing with a large amount of subcarriers.

Note that for values of $N_c \geq 3600$, the speedup achieved for the 16-QAM case is higher than the speedup achieved for the 64-QAM case. As said in [97], the number of registers and shared memory used by a kernel can have a significant impact on the number of resident warps. Moreover, the occupancy of the device for a certain configuration gives an idea of how well an algorithm exploits its parallel processing capabilities. However, a higher occupancy does not mean higher performance, since it depends on

other factors such as global memory accesses, divergent branches, etc. Nevertheless, we evaluated the occupancy of the device for each configuration using the CUDA Visual Profiler, resulting in 67% for the QPSK and 16-QAM cases and 58% for 64-QAM. The lower occupancy for the 64-QAM may justify the lower speedup achieved.

Throughput

Next, the throughput and runtime of the hard-output FSD implementation for a 4×4 were evaluated and compared to the one of the trellis-based detector proposed in [95], which considered the Nvidia 9600 GT GPU with 64 cores at 1.9 GHz.

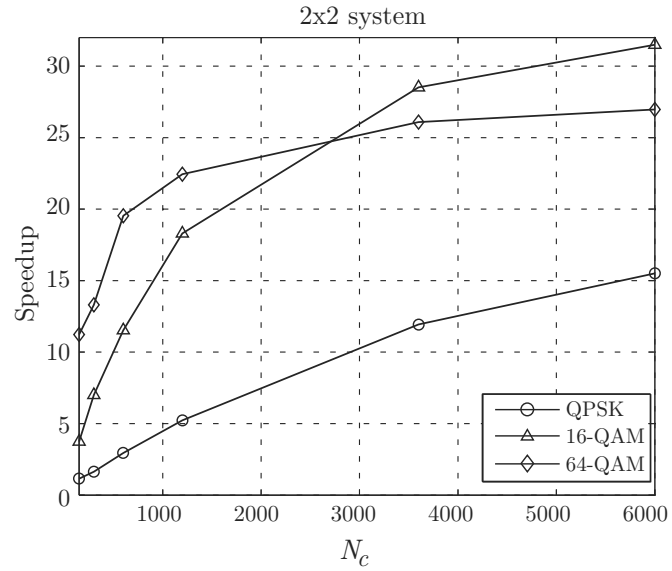
To allow a fair comparison, we defined a factor α which gathers the differences between two GPU (a and b) both in number of cores (N_{core}) and their clock frequencies (f), as follows:

$$\alpha = \frac{N_{core}^{(a)} \cdot f_a}{N_{core}^{(b)} \cdot f_b}. \tag{7.16}$$

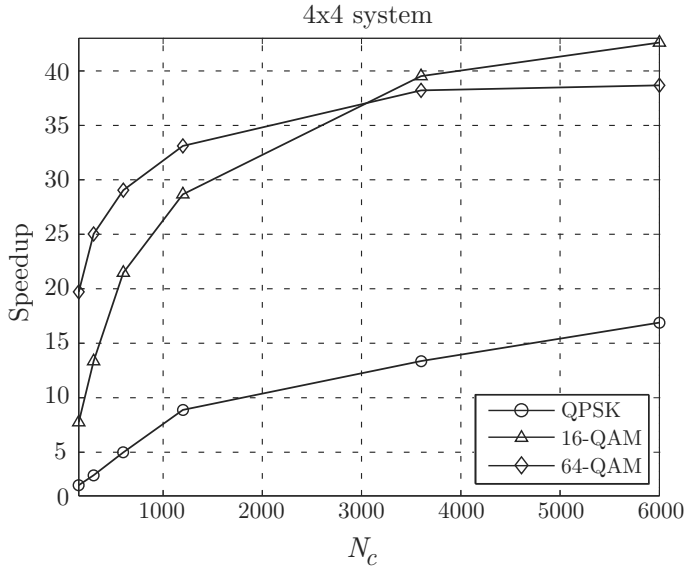
Note that α compares the total processing capability of a certain GPU with another. Since our implementation uses 448 cores against the 64 cores used in [95] (7 times more) but the cores of our device (Nvidia Tesla C2070 GPU) are much slower (1.15 GHz against 1.9 GHz), this gives a value of $\alpha_h \simeq 4.24$. Thus, we included among the results in Table 7.5 some results weighted by α_h to allow a fairer comparison with the trellis-based implementation.

It can be seen that our proposed approach outperforms the scheme in [95] for all the constellation values. Moreover, recall that while the hard-output FSD algorithm achieves the same performance as the optimal ML detector, the trellis-based architecture is not shown to have this behavior and may not reach the ML solution in all cases. Thus, our proposed GPU implementation is more efficient than the trellis-based approach both in terms of throughput and BER performance.

Regarding the LTE real-time requirements, for the QPSK and 16-QAM cases a whole slot can be processed before having received the following one (i.e. in $\leq$0.5 ms) for all the considered subcarrier configurations. For the 64-QAM case, the runtime is $\leq$0.5 ms for the $N_c = 150$ and $N_c = 300$



(a)



(b)

Figure 7.9. Speedup for the hard-output FSD with different constellations and number of subcarriers: (a) 2×2 MIMO system and (b) 4×4 MIMO system.

Table 7.4. Runtime of the FSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [95].

	QPSK	16-QAM	64-QAM
FSD ($N_c = 150$)	0.046 ms	0.054 ms	0.219 ms
FSD ($N_c = 300$)	0.054 ms	0.077 ms	0.409 ms
FSD ($N_c = 600$)	0.078 ms	0.126 ms	0.772 ms
FSD ($N_c = 900$)	0.102 ms	0.174 ms	1.152 ms
FSD ($N_c = 1200$)	0.125 ms	0.220 ms	1.518 ms
FSD ($N_c = 300$)*	0.229 ms	0.326 ms	1.73 ms
Trellis ($N_c = 300$)**	0.350 ms	0.427 ms	3.31 ms

*Results weighted by α_h.

**Results using Nvidia 9600 GT GPU.

cases. Therefore, the proposed GPU implementation can manage all the LTE configurations except the three having the highest values of N_c when using 64-QAM symbols. Thus, further work is needed to decrease the runtime of the latter configurations, either by further optimizing the code or by making use of more powerful GPUs.

We used the CUDA Compute Visual Profiler to assess our implementation and observed that the bottleneck of our implementation is the size of the shared memory.

7.5.3 Soft-Output FSD

Speedup

Before comparing to other soft-output FSD implementations, the parallel implementation of the soft-output FSD on GPU was compared to the implementation in CPU. Contrary to the hard-output FSD implementation, the values of $N_{th_x} \times N_{th_y}$ that minimize the execution time depend on the system size. The selected values are included in Table 7.6.

Table 7.5. Throughput of the FSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [95].

	QPSK	16-QAM	64-QAM
FSD ($N_c = 150$)	182.61 Mbps	311.11 Mbps	115.07 Mbps
FSD ($N_c = 300$)	311.11 Mbps	436.36 Mbps	123.23 Mbps
FSD ($N_c = 600$)	430.77 Mbps	533.33 Mbps	130.57 Mbps
FSD ($N_c = 900$)	494.12 Mbps	579.31 Mbps	131.25 Mbps
FSD ($N_c = 1200$)	537.60 Mbps	610.91 Mbps	132.81 Mbps
FSD ($N_c = 300$)*	73.37 Mbps	102.92 Mbps	29.06 Mbps
Trellis ($N_c = 300$)**	46.16 Mbps	74.30 Mbps	14.50 Mbps

*Results weighted by α_h. **Results using Nvidia 9600 GT GPU

Table 7.6. Optimal block size for the SFSD implementation.

	QPSK		16-QAM		64-QAM	
N_c	$n_T = 2$	$n_T = 4$	$n_T = 2$	$n_T = 4$	$n_T = 2$	$n_T = 4$
300	16×4	4×4	4×16	4×16	8×64	1×64
1200	16×4	8×4	32×16	4×16	4×64	1×64
6000	32×4	16×4	8×16	4×16	2×64	1×64

The resulting speedup is depicted in Fig. 7.10. It can be seen that, for the QPSK and 16-QAM cases, the speedup for the 4×4 system remains nearly the same than for the 2×2 case. On the other hand, the speedup of the 64-QAM scheme increases with respect to the 2×2 case. However, it experiences a saturation effect for high N_c values and it is always below the speedup of the 16-QAM configuration.

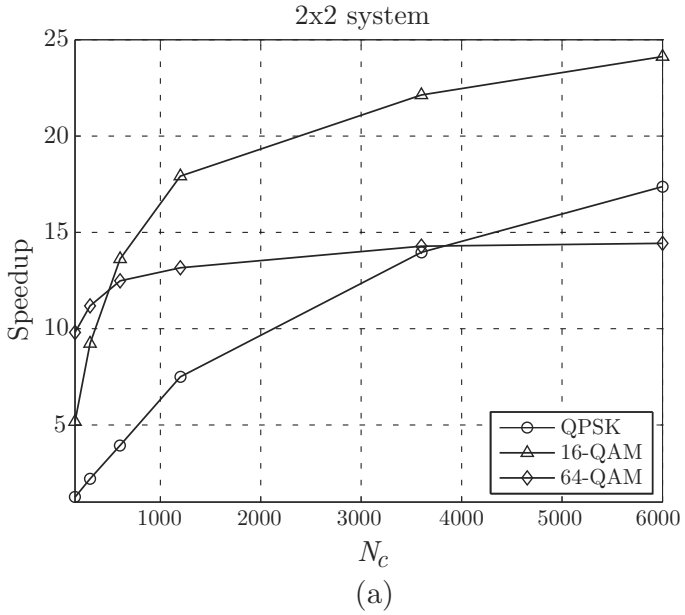
The GPU occupancy was evaluated and scored 50% for QPSK and 16-QAM, whereas only 33% for 64-QAM. This fact together with the higher amount of shared memory needed by the 64-QAM case with respect to the 16-QAM somewhat justifies the saturation effect shown by the 64-QAM speedup curve.

Throughput

Next, the throughput achieved by the SFSD FSD implementation for a 4×4 was evaluated and compared to the one of the trellis-based soft MIMO detector proposed in [20], which considered the detection of 8 streams of 8192 symbols simultaneously in a Nvidia Tesla C1060 GPU with 240 cores at 1.3 GHz.

The comparison between our device and the one used in [20] leads to a value of $\alpha_s \simeq 1.65$. Thus, we included among the results in Tables 7.7 and 7.8 some results weighted by α_s to allow a fairer comparison with the trellis-based implementation. The comparison in Table 7.8 reveals that, considering the weighted results, the proposed SFSD implementation achieves higher throughput than the trellis-based approach for all cases.

Regarding the LTE real-time requirements, for the QPSK case, the proposed SFSD GPU implementation can process a whole slot before having received the following one for all N_c values except for $N_c = 1200$. Nevertheless, real-time is nearly achieved for the latter case. For the 16-QAM case, however, the runtime is $\leq$0.5 ms only for the $N_c = 150$ case. Therefore, the SFSD GPU implementation requires further optimizations than the hard-output FSD to meet real-time. In fact, the occupancy values are lower than their respective ones in the FSD implementation.



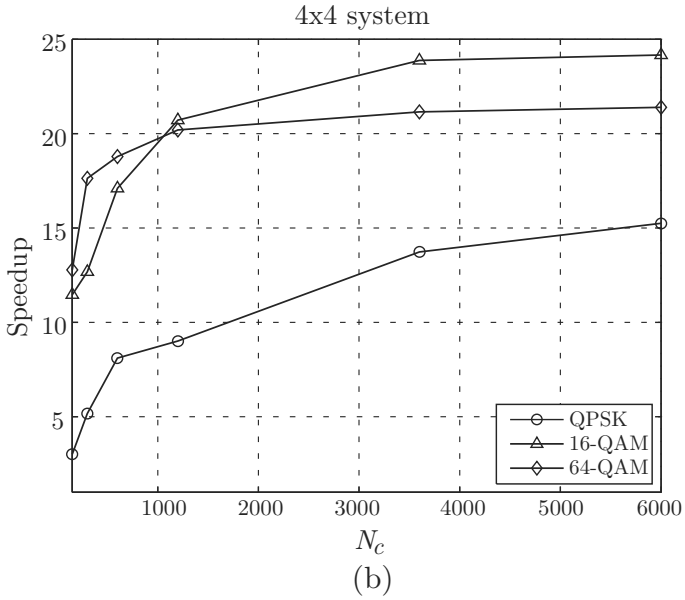


Figure 7.10. Speedup for the soft-output FSD with different constellations and number of subcarriers: (a) 2×2 MIMO system and (b) 4×4 MIMO system.

Table 7.7. Runtime of the SFSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [20].

	QPSK	16-QAM	64-QAM
SFSD ($N_c = 150$)	0.115 ms	0.373 ms	2.57 ms
SFSD ($N_c = 300$)	0.204 ms	0.680 ms	5.17 ms
SFSD ($N_c = 600$)	0.317 ms	1.29 ms	10.40 ms
SFSD ($N_c = 900$)	0.426 ms	1.94 ms	15.77 ms
SFSD ($N_c = 1200$)	0.535 ms	2.58 ms	21 ms
SFSD (8×8192 sym)*	1.56 ms	8.28 ms	67.88 ms
Trellis (8×8192 sym)**	1.76 ms	8.31 ms	124.62 ms

*Results weighted by α_s.

**Results using Nvidia Tesla C1060 GPU.

Table 7.8. Throughput of the SFSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [20]

	QPSK	16-QAM	64-QAM
SFSD ($N_c = 150$)	73.04 Mbps	45.04 Mbps	9.82 Mbps
SFSD ($N_c = 300$)	82.35 Mbps	49.41 Mbps	9.74 Mbps
SFSD ($N_c = 600$)	105.99 Mbps	52.09 Mbps	9.68 Mbps
SFSD ($N_c = 900$)	118.31 Mbps	51.88 Mbps	9.59 Mbps
SFSD ($N_c = 1200$)	125.61 Mbps	52.09 Mbps	9.60 Mbps
SFSD (8×8192 sym)*	87.36 Mbps	32.86 Mbps	6.01 Mbps
Trellis (8×8192 sym)**	74.47 Mbps	27.94 Mbps	3.15 Mbps

*Results weighted by α_s. **Results using Nvidia Tesla C1060 GPU.

Table 7.9. GPU execution times for the proposed norm-based preprocessing and for the FSD-based preprocessing, with $n_T = 4$ and different N_c values.

N_c	150	300	600	1200	3600	6000
Norm	0.618 ms	0.631 ms	0.694 ms	1.36 ms	2.87 ms	4.32 ms
FSD	6.96 ms	10.4 ms	16.8 ms	26 ms	70.5 ms	105 ms

7.5.4 Fully-Parallel SFSD

Preprocessing execution time

As said in Section 2.3.5, the authors in [46] proposed a channel-matrix ordering which was shown to obtain good performance when used before the FSD. This ordering, from now on referred to as FSD ordering, was recently used by the implementation in [104] to perform soft-output detection. The method in [104] consisted of n_T different FSD-like orderings followed by n_T FSD tree-searches.

For the FPFSD scheme we proposed a column-norm-based ordering and argued that it is more suitable for GPU implementation than the FSD-based. Note that the FPFSD BER results obtained with the two channel matrix orderings under study revealed that both preprocessing strategies achieve the same average performance. Thus, the execution time of the column-norm-based ordering implementation can be fairly compared to the FSD-based channel-matrix ordering strategy suggested in [104]. Table 7.9 shows the execution times of the proposed norm-based preprocessing and of the FSD-based preprocessing used in [104], both implemented on GPU for a 4×4 system with different numbers of subcarriers. It can be seen that the execution time of the FSD-based preprocessing is much higher (up to $\simeq 24$ times) than the norm-based preprocessing time, showing that the proposed norm-based preprocessing exploits the GPU resources much better.

Table 7.10. Optimal block size for the FPFSD implementation.

	QPSK		16-QAM		64-QAM	
N_c	$n_T = 2$	$n_T = 4$	$n_T = 2$	$n_T = 4$	$n_T = 2$	$n_T = 4$
300	8×8	8×8	16×16	8×8	16×16	16×16
1200	8×16	8×8	8×16	8×16	16×16	16×64
6000	8×16	8×8	8×16	8×16	16×16	16×16

Speedup

Next, the GPU implementation of the FPFSD was compared to its respective sequential implementation in CPU. Table 7.10 contains the values of $N_{th_x} \times N_{th_y}$ that minimize the execution time for the FPFSD GPU implementation. Note again that, for simplicity, the results for only some representative N_c values were included.

Fig. 7.11 (a) shows the speedup for a 2×2 system and the three considered constellations, as a function of the number of subcarriers. It can be observed that the higher the constellation size, the higher the achieved speedup. Fig. 7.11 (b) depicts the results for a 4×4 MIMO system. Here it can be observed that, for the QPSK and 16-QAM constellations, the speedup increases with respect to the one of the 2×2 case. However, for the 64-QAM case, the speedup increase is hardly noticeable. Therefore, it can be said that, in general, the speedup increases as the number of subcarriers and/or the system size gets higher and, particularly for the 2×2 system, the speedup also increases as the constellation size grows.

In fact, the speedup for the 64-QAM case in a 4×4 system exhibits a totally different behavior, which can be caused by the use of a higher amount of shared memory, which brings the GPU to saturation and makes some threads be executed sequentially. In any case, a substantial gain with respect to sequential execution in one core of a high-performance CPU is observed for all cases under study.

The GPU occupancy for this implementation is 67% for QPSK and

16-QAM and 58% for 64-QAM. These values are almost the same ones as those reached by the hard-output FSD implementation, being lower than the ones of the SFSD. This fact may be related with the high speedup of the FPFSD implementation against the SFSD one.

Throughput

Tables 7.11 and 7.12 collect the runtime and throughput values of the FPFSD implementation for the considered LTE configurations. To allow a fair comparison between our proposed approach and the trellis-based implementation in [20], we also executed the FPFSD CUDA code on the GPU employed in [20] (Nvidia Tesla C1060) and for the configuration used there, i.e. for the simultaneous detection of 8 streams of 8192 symbols. This comparison reveals that the FPFSD implementation achieves the same throughput as the trellis-based detector for QPSK and slightly higher throughput for the 16-QAM case. For 64-QAM, our proposed approach doubles the throughput.

Regarding the LTE real-time requirements, for the QPSK case a whole slot can be processed before having received the following one (i.e. in $\leq$0.5 ms) in all cases. For the 16-QAM case, the runtime is $\leq$0.5 ms for the $N_c = 150$ and $N_c = 300$ cases, being the peak throughput approximately halved in all cases with respect to QPSK. For the 64-QAM case, the LTE requirement is not fulfilled at any subcarrier configuration. Also, the increase in the number of bits per symbol does not compensate the runtime increase, leading to much lower throughput values. Since the GPU occupancy is 50% for the three constellations, the time to access GPU global memory was found to be the limiting factor of the implementation. The reason is that the higher the constellation size, the higher the number of threads accessing this memory.

Moreover, further work is also needed to further optimize the code or to make use of more powerful GPU in order to decrease the runtime of those configurations not fulfilling the LTE standard requirements.

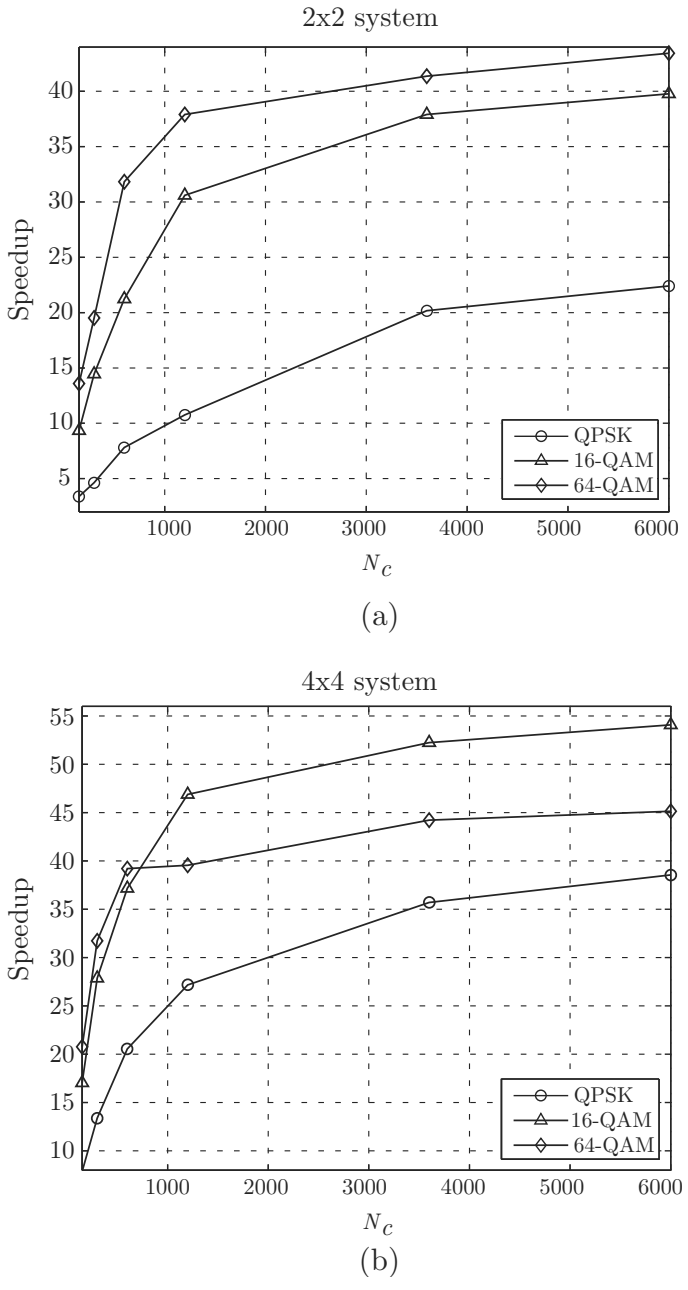


Figure 7.11. Speedup of the FPFSD for different constellations and number of subcarriers in a MIMO system of size: (a) 2×2 and (b) 4×4.

Table 7.11. Runtime of the FPFSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [20].

	QPSK	16-QAM	64-QAM
FPFSD ($N_c = 150$)	0.078 ms	0.232 ms	1.36 ms
FPFSD ($N_c = 300$)	0.112 ms	0.407 ms	2.71 ms
FPFSD ($N_c = 600$)	0.186 ms	0.743 ms	5.65 ms
FPFSD ($N_c = 900$)	0.253 ms	1.103 ms	8.68 ms
FPFSD ($N_c = 1200$)	0.317 ms	1.456 ms	11.72 ms
FPFSD (8×8192 sym)*	1.75 ms	8.07 ms	64.74 ms
Trellis (8×8192 sym)*	1.76 ms	8.31 ms	124.62 ms

*Results using Nvidia Tesla C1060 GPU

Table 7.12. Throughput of the FPFSD proposed implementation in the Nvidia Tesla C2070 GPU for a 4×4 system for different configurations compared to trellis-based detector results in [20].

	QPSK	16-QAM	64-QAM
FPFSD ($N_c = 150$)	107.70 Mbps	72.41 Mbps	18.60 Mbps
FPFSD ($N_c = 300$)	150 Mbps	82.56 Mbps	18.59 Mbps
FPFSD ($N_c = 600$)	180.65 Mbps	90.44 Mbps	17.84 Mbps
FPFSD ($N_c = 900$)	199.21 Mbps	91.39 Mbps	17.41 Mbps
FPFSD ($N_c = 1200$)	211.99 Mbps	92.31 Mbps	17.20 Mbps
FPFSD (8×8192 sym)*	74.88 Mbps	32.48 Mbps	6.07 Mbps
Trellis (8×8192 sym)*	74.47 Mbps	27.94 Mbps	3.15 Mbps

*Results using Nvidia Tesla C1060 GPU

7.6 Conclusion

In this chapter, several MIMO detectors based on the fixed-complexity sphere decoder have been implemented in GPU. The detection stage is highly accelerated through exploiting two parallelism levels: first, independent parts of the considered algorithms are processed in parallel and, second, the detection step is carried out simultaneously for all the subcarriers in the system through forwarding each parallel stream to a different thread.

The block size was shown to be crucial to efficiently exploit the GPU capabilities. A study of the execution time for all the constellations, number of subcarriers and system sizes was carried out to select the best block configuration, and it showed the high dependency of this parameter on the system features.

The selected methods were also implemented in a high-performance CPU and the CPU execution times were compared to the execution times of the proposed GPU implementations. Speedup results showed that the GPU-based hard-output FSD implementation performs up to 40 times faster than its CPU-equivalent for some cases whereas the SFSD scheme reaches a speedup of 24.

In addition, a fully parallel soft-output scheme (FPFSD) with a low-complexity preprocessing stage was proposed and implemented in GPU. The speedup and throughput of the algorithm were assessed and compared to those of the SFSD and trellis-based schemes. Results showed that the GPU-based FPFSD implementation performs nearly up to 55 times faster than its CPU-equivalent for some cases. Hence, this method exploits the GPU capabilities even further than the SFSD. Thus, a high degree of parallelism is indeed exploited for the three considered algorithms. Moreover, the speedup increases with the system size and number of subcarriers, showing the interest of GPU implementation for configurations managing many data streams simultaneously.

The throughput of the three GPU implementations considering a 4×4 system was obtained for the three constellations. Among the hard-output

algorithms, the FSD achieves higher throughput than the trellis-based scheme. On the other hand, comparing the two soft-output implementations (SFSD and FPFSD) and the trellis-based soft-output scheme, the FPFSD implementation achieves the highest throughput.

Future work is needed to decrease the runtime for those configurations not attaining real-time. The use of either more powerful GPU and/or more than one GPU might be promising solutions for this purpose. Another interesting topic for future research is to analyze the amount of energy consumed by the proposed GPU implementation.

The GPU implementations of the FSD and SFSD are reported in [105]. The paper containing the design and GPU implementation of the FPFSD is [106].

8 Conclusions

8 Conclusions

The overall aim of this research was to deepen into MIMO wireless systems and, specially, into the task of MIMO data detection. The motivation of this research came from the necessity of finding low-complexity receivers with as good BER performance as possible.

This chapter will summarize the findings of this research work, revisiting the research objectives given in the introductory chapter. First, Section 8.1 will review the contents of this study, outlining the main conclusions that were extracted from each chapter. Recommendations for future research will be discussed in Section 8.2. Additionally, the final section contains a list of works published during the course of candidature for the degree.

8.1 Summary

The first part of this dissertation presented a performance and a complexity study of the application of preprocessing techniques to the K-Best tree-search MIMO detector. Both, a comparison between two LR algorithms

(the LLL and fcLLL) and a column ordering strategy based on the channel matrix, the VBLAST ZF-DFE, was performed. Also, an efficient QR-based implementation of the latter was proposed, showing a complexity reduction from $\mathcal{O}(n_T^5)$ to $\mathcal{O}(n_T^4)$. The results demonstrated that the VBLAST ZF-DFE preprocessing algorithm requires a higher number of operations than the fcLLL and LLL algorithms for the average case. If the worst-case is considered, however, the complexity of VBLAST ZF-DFE is between the fcLLL and the LLL, being the LLL the most costly algorithm. The comparison conducted between both methods showed that their performance is highly dependent on the SNR and K value. As a result, general guidelines to select the best algorithm are hard to be given and the preprocessing method must be selected according to the application needs and the available resources.

Chapter 4 presented two condition-number-thresholding-based techniques built from the K-Best detector. It was also proposed a condition number estimator that reuses the computation of the QR decomposition of the channel matrix, which is always a previous step before tree-search detection. A meaningful way to determine the threshold condition number was additionally proposed. The performance and complexity results showed that the proposed schemes achieve the same performance as other conventional K-Best detectors but with the advantage of having lower average complexity. Thus, these schemes offer average power saving. Moreover, the proposed condition number estimator exhibited lower complexity than other published approaches without degrading the performance of the proposed detection schemes.

Chapter 5 presented several contribution involving the use of LR techniques. First, the combination of LR with the K-Best algorithm was investigated. Then, new efficient LRA K-Best schemes were developed. An extended LLL algorithm for LR was proposed to assist the preprocessing stage of the above schemes. In the last part of the chapter, the extended LLL algorithm was exploited to decrease the computational cost of several LRA precoding methods for multiuser communication. Finally, these efficient precoding approaches were compared to some well-known precoding methods in terms of both computational cost and complexity.

Chapter 6 considered soft-output detection in MIMO-BICM systems working with LLR quantization. The computational cost of a soft-output fixed-complexity sphere decoder (SFSD) was reduced by a quantization-based pruning. The SFSD scheme was modified to perform quantization-based tree pruning and LLR clipping. The results showed that nearly uniform complexity reduction along the SNR range was achieved without noticeably degrading the results after quantization.

The implementation of several MIMO detection algorithms in GPU was covered in Chapter 7. Both, hard- and soft-output tree-search-based algorithms were implemented. The computational times of the proposed GPU implementations were compared with their execution on a high performance CPU in order to provide speedup results. In addition, the throughput of the algorithms was evaluated and compared to those of other recent implementations. Results showed that high speedup was achieved (up to 55), which increased with the system size and number of subcarriers. Thus, the interest of GPU implementation for configurations managing many data streams simultaneously was demonstrated. Furthermore, the throughput values of the methods were calculated for some LTE configurations and compared to the ones of previously proposed trellis-based detectors. The resulting throughput outperformed those of the trellis-based detectors for all the cases under study and was also shown to support many configurations included in the LTE Advanced standard.

8.2 Further Work

Following the investigations described in this thesis, the main lines of research that remain open are listed below:

- This thesis was focused on the design of MIMO receivers assuming perfect knowledge about the channel matrix (perfect CSI). In practice, however, it is common to only have information about a version of the channel matrix which includes estimation errors. It has been shown that the design of MIMO receivers taking into account the possible estimation errors improves detection performance. Hence,

it would be interesting to evaluate and redesign the detection approaches developed in this thesis taking into account channel estimation errors.

- The MIMO receivers developed in this thesis were mainly designed and tested assuming the transmission through flat fading channels with zero-mean Gaussian entries. Either using more realistic channel models or directly real channel measurements to test the detectors would be interesting topics for future research.

- The performance of MIMO-BICM receivers was tested considering only iterative decoding instead of joint iterative demodulation and decoding (i.e. no information was exchanged iteratively between the demodulator and the channel decoder). It has been shown that the detection performance can be further improved if the information from already decoded bits is reused by the demodulator as *a priori* information. Considering these kind of iterative receivers for the design of efficient demodulators would be definitely a good future research line.

- Almost all the algorithms selected for GPU implementation were based on the fixed-complexity sphere decoder, since this method could exploit well the parallel processing capabilities of these devices. Nevertheless, there is still room for further development of preprocessing and detection schemes suitable for GPU implementation.

- The GPU implementations carried out in this thesis considered the use of a single GPU. Nevertheless, devices including more than one GPU are already a reality. Thus, an open research line is to further exploit the parallelism of the implemented MIMO schemes by mapping them into multiple GPUs.

8.3 List of Publications

A list of published work produced during the course of candidature for the degree is presented in what follows. Note that the publications where the author of this thesis is the primary author are differentiated from the rest of publications.

Publications as the first author

Refereed ISI Journals

- **S. Roger**, A. Gonzalez, V. Almenar, G. Matz, "Efficient Fixed-Complexity Sphere Decoder with Quantized Outputs", *IEEE Communications Letters*, submitted 2012.
- **S. Roger**, C. Ramiro, A. Gonzalez, V. Almenar, A. M. Vidal, "Fully parallel GPU Implementation of a Fixed-Complexity Soft-Output MIMO Detector", *IEEE Transactions on Vehicular Technology*, submitted 2012.
- **S. Roger**, C. Ramiro, A. Gonzalez, V. Almenar, A. M. Vidal, "An Efficient GPU Implementation of Fixed-Complexity Sphere Decoders for MIMO Wireless Systems", *Integrated Computed-Aided Engineering*, submitted 2011.
- **S. Roger**, A. Gonzalez, V. Almenar, A. M. Vidal, "Practical Aspects of Preprocessing Techniques for K-Best Tree Search MIMO Detectors", *Computers and Electrical Engineering (Elsevier)*, vol. 37, no. 4, pp. 451-460, July 2011.
- **S. Roger**, A. Gonzalez, V. Almenar, A. M. Vidal, "Extended LLL Algorithm for Efficient Signal Precoding in Multiuser Communication Systems", *IEEE Communications Letters*, vol. 14, no. 3, pp. 220-222, March 2010.

- **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "MIMO Channel Matrix Condition Number Estimation and Threshold Selection for Combined K-Best Sphere Decoders," *IEICE Transactions on Communications*, vol. E92-B, no. 4, pp. 1380-1383, April 2009.

Peer-reviewed non-ISI Journals

- **S. Roger**, F. Domene, C. Botella, G. Piñero, A. Gonzalez and V. Almenar, "Recent Advances in MIMO Wireless Systems", in *Waves*, vol. 1, pp. 115-123, 2009.

- **S. Roger** and M. Cobos, "Developing Your Electrical Engineering Degree Thesis", in *IEEE Potentials Magazine*, no. 4, vol. 28, pp. 12-16, 2009.

Papers in International Conferences

- **S. Roger**, C. Ramiro, A. Gonzalez, V. Almenar and A. M. Vidal, "Rapid Prototyping of MIMO Detectors Using Graphic Processing Units", *First Women's Workshop on Communications and Signal Processing*, Banff, Canada, July 2012.

- **S. Roger**, F. Domene, A. Gonzalez, V. Almenar, G. Piñero, "An Evaluation of Precoding Techniques for Multiuser Communication Systems", *International Symposium on Wireless Communications Systems* (ISWCS), York, United Kingdom, September 2010.

- **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "Variable-Breadth K-Best Detector for MIMO Systems", *IEEE International Wireless Communications and Mobile Computing Conference* (IWCMC), Caen, France, July 2010.

- **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "Lattice-Reduction-Aided K-Best MIMO Detector based on the Channel Matrix Condition Number", *IEEE International Symposium on Communications, Control and Signal Processing* (ISCCSP), Limassol, Cyprus, March 2010.

- **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "On Decreasing the Complexity of Lattice-Reduction-Aided K-Best MIMO Detectors," *European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, United Kingdom, August 2009.

- **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "Combined K-Best Sphere Decoder based on the Channel Matrix Condition Number," *IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP)*, St. Julians, Malta, March 2008.

Papers in National Conferences

- **S. Roger**, C. Ramiro, A. Gonzalez, V. Almenar, A. M. Vidal, "On the Use of Graphic Processing Units for the Efficient Implementation of MIMO Detectors", *Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Elche, Spain, September 2012.

Patents

- A Maximum-Likelihood Sphere Decoding Method for Multiple-Input Multiple-Output (MIMO) Systems.
  - *Inventors*: V. M. Garcia, A. Gonzalez, **S. Roger** and A. M. Vidal.
  - *Number*: P201230387.
  - *Date*: 14/03/2012.
  - *Holder*: Universitat Politècnica de València.

Other coauthored publications related to this thesis

Peer-reviewed non-ISI Journal Papers

- V. M. Garcia, A. Gonzalez, C. Gonzalez, F. J. Martinez-Zaldivar, C. Ramiro, **S. Roger**, A. M. Vidal, "The Impact of GPU/Multicore in Signal Processing: a Quantitative Approach", in *Waves*, vol. 3, pp. 96-106, 2011.

- A. Gonzalez, J. A. Belloch, G. Piñero, J. Lorente, M. Ferrer, **S. Roger**, C. Roig, F. J. Martinez, M. de Diego, P. Alonso, V. M. Garcia, E. S. Quintana-Ortí, A. Remon and A. M. Vidal, "Application of Multi-core and GPU Architectures on Signal Processing: Case Studies", in *Waves*, vol. 2, pp. 86-96, 2010.

Papers in International Conferences

- C. Ramiro, **S. Roger**, A. Gonzalez, V. Almenar and A. M. Vidal, "Parallel Implementation of a Fixed-Complexity MIMO Detector on a Multi-Core System", in *International Conference on Computational and Mathematical Methods in Science and Engineering* (CMMSE), La Manga, Spain, 2012.

- F. Domene, **S. Roger**, C. Ramiro, G. Piñero, A. Gonzalez, "A Reconfigurable GPU Implementation for Tomlinson-Harashima Precoding", *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), Kyoto, Japan, 2012.

- V. M. Garcia, **S. Roger**, R. A. Trujillo, A. M. Vidal, A. Gonzalez, "A Deterministic Lower Bound for the Radius in Sphere Decoding Search", *International Conference on Advanced Technologies for Communications* (ATC/REV), Ho-Chi-Mihn City, Vietnam, 2010.

- J. Fink, **S. Roger**, A. Gonzalez, V. Almenar and V. M. Garcia, "Complexity Assessment of Sphere Decoding Methods for MIMO Detection," *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Ajman, UAE, December 2009.

- R. A. Trujillo, V. M. Garcia, A. M. Vidal, **S. Roger** and A. Gonzalez, "A Gradient-Based Ordering for MIMO Detection," *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Ajman, UAE, December 2009.

Papers in National Conferences

- F. Domene, **S. Roger**, C. Ramiro, G. Piñero, A. Gonzalez, "Efficient GPU Implementation of Precoding Algorithms for MIMO-OFDM Systems", *Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Elche, Spain, September 2012.

- J. Fink, **S. Roger**, A. Gonzalez and V. Almenar, "Complexity evaluation of Sphere Decoding techniques for MIMO detection," *Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Santander, Spain, September 2009.

- V. Motos, **S. Roger**, A. Gonzalez and V. Almenar, "Análisis de las prestaciones de detectores MIMO en presencia de errores de estimación del canal," *Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Santander, Spain, September 2009.

Bibliography

[1] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bölcskei, "An overview of MIMO communications - a key to Gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–218, 2004.

[2] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge University Press, 2007.

[3] 3GPP TS 36.201, V10.0.0, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer - General Description," December 2010.

[4] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, no. 3, p. 311335, March 1998.

[5] I. E. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, November 1999.

[6] V. Tarokh, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communication: performance criterion

and code construction," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 744–765, March 1998.

[7] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, October 1998.

[8] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1456–1467, July 1999.

[9] L. Zheng and D. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[10] M. Jiang and L. Hanzo, "Multiuser MIMO-OFDM for next-generation wireless systems," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1430 –1469, 2007.

[11] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, August 2002.

[12] B. Hassibi and H. Vikalo, "On Sphere Decoding algorithm. Part I, the expected complexity," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 2806–2818, August 2005.

[13] E. G. Larsson, "MIMO detection methods: How they work," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 91–95, May 2009.

[14] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-Best Sphere Decoding for MIMO Detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491–503, March 2006.

[15] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, Island of Kos, Greece, May 2006.

[16] H. Yao and G. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *IEEE Global Communications Conference*, Taipei, Taiwan, November 2002.

[17] C. Windpassinger and R. F. H. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *IEEE Information Theory Workshop*, Paris, France, 2003.

[18] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "Near-Maximum-Likelihood Detection of MIMO Systems using MMSE-Based Lattice Reduction," in *IEEE International Conference on Communications*, Paris, France, June 2004.

[19] G. Falcao, V. Silva, and L. Sousa, "How GPUs can outperform ASICs for fast LDPC decoding," in *International Conference on Supercomputing*, Yorktown Heights, New York (USA), 2009.

[20] M. Wu, Y. Sun, S. Gupta, and J. Cavallaro, "Implementation of a high throughput soft MIMO detector on GPU," *Journal of Signal Processing Systems*, vol. 64, no. 2, pp. 123–136, July 2011.

[21] A. Lenstra, H. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[22] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs. Technical Journal*, vol. 1, no. 2, pp. 41–59, Autumn 1996.

[23] D. Seethaler, G. Matz, and F. Hlawatsch, "Low-Complexity MIMO Data Detection using Seysen's Lattice Reduction Algorithm," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, Honolulu, Hawaii, USA, April 2007.

[24] A. Paulraj, R. Nabar, and D. Gore, *"Introduction to Space-Time Wireless Communications*. United Kingdom: Cambridge University Press, 2003.

[25] J. Janhunen, O. Silvén, and M. Juntti, "Programmable processor implementations of K-best list sphere detector for MIMO receiver," *Signal Processing*, vol. 90, no. 1, pp. 313–323, January 2010.

[26] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 623–656, July 1948.

[27] I. Telatar, "Capacity of multi-antenna gaussian channels," ATT Bell Laboratories, Tech. Rep., 1995.

[28] A. Goldsmith, S. A. Jafar, N. Jindal, and S. Vishwanath, "Capacity limits of MIMO channels," *IEEE Journal of Selected Areas in Communications*, vol. 21, no. 5, pp. 684–702, June 2003.

[29] D. Wübben, D. Seethaler, J. Jalden, and G. Matz, "Lattice reduction: A survey with applications in wireless communications," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 70–91, May 2011.

[30] T. Kailath, H. Vikalo, and B. Hassibi, *"MIMO Receive Algorithms," in Space-Time Wireless Systems: From Array Processing to MIMO Communications, (editors H. Bolcskei, D. Gesbert, C. Papadias, and A. J. van der Veen).* United Kingdom: Cambridge University Press, 2005.

[31] J. Barry, E. Lee, and D. Messerschmitt, *Digital Communications.* United States: Ed. Springer, 2003 (3rd Edition).

[32] M. O. Damen, H. E. Gamal, and G. Caire, "On Maximum-Likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, October 2003.

[33] I. Berenguer and X. Wang, "Space-Time coding and signal processing for MIMO communications," *Journal of Computer Science and Technology*, vol. 18, no. 6, pp. 689–702, November 2003.

[34] K. Su and I. J. Wassell, "A new ordering for efficient Sphere Decoding," in *IEEE International Conference on Communications*, Seoul, Korea, May 2005.

[35] R. A. Trujillo, V. M. Garcia, A. M. Vidal, S. Roger, and A. Gonzalez, "A gradient-based ordering for MIMO detection," in *IEEE ISSPIT*, Ajman, UAE, 2009.

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

[37] K. Su, "Efficient Maximum Likelihood detection for communication over MIMO channels," University of Cambridge, Technical Report, February 2005.

[38] 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.

[39] C. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, no. 2, pp. 181–191, September 1994.

[40] M. Stojnic, H. Vikalo, and B. Hassibi, "Speeding up the sphere decoder with h-infinity and SDP inspired lower bounds," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 712–726, February 2008.

[41] J. Maurer, J. Jalden, D. Seethaler, and G. Matz, "Achieving a continuos diversity-complexity tradeoff in wireless MIMO systems via preequalized sphere-decoding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 6, pp. 986–999, December 2009.

[42] V. M. Garcia, S. Roger, R. A. Trujillo, A. M. Vidal, and A. Gonzalez, "A deterministic lower bound for the radius in sphere decoding search," in *International Conference on Advanced Technologies for Communications*, Ho Chi Mihn City, Vietnam, October 2010.

[43] Y. H. Wu, Y. T. Liu, H.-C. Chang, Y.-C. Liao, and H.-C. Chang, "Early-pruned K-best sphere decoding algorithm based on radius constraints," in *IEEE International Conference on Communications*, Beijing, China, May 2008.

[44] C.-A. Shen and A. Eltawil, "A radius adaptive K-best decoder with early termination: Algorithm and VLSI architecture," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 9, pp. 2476 –2486, September 2010.

[45] Q. Li and Z. Wang, "Improved K-Best Sphere Decoding algorithms for MIMO systems," in *International Symposium on Circuits and Systems (ISCAS 2006)*, Island of Kos, Greece, May 2006.

[46] L. G. Barbero and J. S. Thompson, "Fixing the complexity of the sphere decoder for MIMO detection," *IEEE Transactions on Wireless Communications*, vol. 7, no. 6, pp. 2131–2142, June 2008.

[47] J. Jalden, L. G. Barbero, B. Ottersten, and J. S. Thompson, "The error probability of the fixed-complexity sphere decoder," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2711–2720, July 2009.

[48] J. Fink, S. Roger, A. Gonzalez, V. Almenar, and V. M. Garcia, "Complexity assessment of sphere decoding methods for MIMO detection," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Ajman, UAE, December 2009.

[49] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[50] J. Boutros, F. Boixadera, and C. Lamy, "Bit-interleaved coded modulations for multiple-input multiple-output channels," in *IEEE Sixth International Symposium on Spread Spectrum Techniques and Applications*, New Jersey, USA, September 2000.

[51] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.

[52] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, March 2003.

[53] R. Wang and G. B. Giannakis, "Approaching MIMO channel capacity with reduced-complexity soft sphere decoding," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, March 2003.

[54] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 2, pp. 290–300, February 2008.

[55] L. G. Barbero, T. Ratnarajah, and C. Cowan, "A low-complexity soft-MIMO detector based on the fixed-complexity sphere decoder," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, Nevada (USA), March 2008.

[56] L. G. Barbero and J. S. Thompson, "Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2804–2814, September 2008.

[57] 3GPP TS 36.300, V8.9.0, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer - General Description," December 2009.

[58] B. Hochwald, C. Peel, and A. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication, part II: Perturbation," *IEEE Transactions on Communications*, vol. 53, no. 3, pp. 537–544, January 2005.

[59] C. Windpassinger, R. Fischer, T. Vencel, and J. Huber, "Precoding in multiantenna and multiuser communications," *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1305–1316, July 2004.

[60] X.-F. Qi and K. Holt, "A Lattice-Reduction-Aided Soft Demapper for High-Rate Coded MIMO-OFDM Systems," *IEEE Signal Processing Letters*, vol. 14, no. 5, pp. 305 – 308, May 2007.

[61] G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 11, pp. 1841–1852, November 1999.

[62] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Transactions on Signal Processing*, vol. 51, no. 7, pp. 1722–1730, July 2003.

[63] T. Koike-Akino, "Low-complexity systolic V-BLAST architecture," *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2172–2176, May 2009.

[64] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore: The Johns Hopkins University Press, 1996.

[65] M. Seysen, "Simultaneous reduction of a lattice basis and its reciprocal basis," *Combinatorica*, vol. 13, pp. 363–367, 1993.

[66] H. Vetter, V. Ponnampalam, M. Sandell, and P. A. Hoeher, "Fixed complexity LLL algorithm," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1634–1637, April 2009.

[67] S. Roger, A. Gonzalez, V. Almenar, and A. M. Vidal, "Practical aspects of preprocessing techniques for K-best tree search MIMO detectors," *Computers and Electrical Engineering*, vol. 37, no. 4, pp. 451–460, 2011.

[68] H. Artes, D. Seethaler, and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection," *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2808 – 2820, November 2003.

[69] J. E. Gentle, *Numerical Linear Algebra for Applications in Statistics*. United States: Ed. Springer, 1998, ISBN: 978-0-387-98542-8.

[70] J. Maurer, G. Matz, and D. Seethaler, "Low-complexity and full-diversity MIMO detection based on condition number thresholding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, Honolulu, Hawaii, USA, April 2007.

[71] S. Xu and J. Zhang, "A new data mining approach to predicting matrix condition numbers," *Communications in information and systems, International Press*, vol. 4, no. 4, pp. 325–340, 2004.

[72] A. Edelman, "Eigenvalues and Condition Numbers of Random Matrices," Massachusetts Institute of Technology, Cambridge (MA), Ph.D. thesis, 1989.

[73] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, "An estimate for the condition number of a matrix," *SIAM Journal on Numerical Analysis*, vol. 16, no. 2, pp. 368–375, April 1979.

[74] S. Roger, A. Gonzalez, V. Almenar, and A. M. Vidal, "Combined K-Best Sphere Decoder based on the channel matrix condition number," in *IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP 2008)*, St. Julians, Malta, March 2008.

[75] ——, "Variable-breadth K-best detector for MIMO systems," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, Caen, France, June 2010.

[76] ——, "MIMO channel matrix condition number estimation and threshold selection for combined K-best sphere decoders," *IEICE Transactions on Communications*, vol. E92, no. 4, pp. 1380–1383, 2009.

[77] ——, "Lattice-reduction-aided K-best MIMO detector based on the channel matrix condition number," in *IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP)*, Limassol, Cyprus, March 2010.

[78] I. Berenguer, J. Adeane, I. Wassell, and X. Wang, "Lattice-reduction-aided receivers for MIMO-OFDM in spatial multiplexing systems," in *Proc. Int. Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, vol. 2, September 2004, pp. 1517–1521.

[79] M. Shabany, K. Su, and P. G. Gulak, "A pipelined scalable high-throughput implementation of a Near-ML K-Best complex lattice decoder," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Las Vegas, Nevada, USA, April 2008.

[80] F. J. Richards, "A flexible growth function for empirical use," *J. Exp. Bot.*, vol. 10, pp. 290–300, 1959.

[81] C. Windpassinger, R. F. H. Fischer, and J. B. Huber, "Lattice-reduction-aided broadcast precoding," *IEEE Transactions on Communications*, vol. 52, no. 12, pp. 2057–2060, December 2004.

[82] D. Xu, Y. Huang, and L. Yang, "Improved nonlinear multiuser precoding using lattice reduction," *Signal, Image and Video Processing*, vol. 3, no. 1, pp. 47–52, February 2009.

[83] S. Roger, A. Gonzalez, V. Almenar, and A. M. Vidal, "Extended LLL algorithm for efficient signal precoding in multiuser communication systems," *IEEE Communications Letters*, vol. 14, no. 3, pp. 220–222, March 2010.

[84] ——, "On decreasing the complexiy of lattice-reduction-aided K-best MIMO detectors," in *European Signal Processing Conference (EUSIPCO 2009)*, Glasgow, Scotland, August 2009.

[85] S. Roger, F. Domene, A. Gonzalez, V. Almenar, and G. Piñero, "An evaluation of precoding techniques for multiuser communication systems," in *International Symposium on Wireless Communication Systems*, York, United Kingdom, September 2010.

[86] C. Novak, P. Fertl, and G. Matz, "Quantization for soft-output demodulators in bit-interleaved-coded-modulation systems," in *IEEE ISIT*, Seoul, Korea, 2009.

[87] C. Novak, C. Studer, A. Burg, and G. Matz, "The effect of unreliable LLR storage on the performance of MIMO-BICM," in *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA (USA), November 2010.

[88] W. Rave, "Quantization of log-likelihood ratios to maximize mutual information," *IEEE Signal Processing Letters*, vol. 16, pp. 283–286, April 2009.

[89] S. Roger, A. Gonzalez, V. Almenar, and G. Matz, "Efficient fixed-complexity sphere decoder with quantized outputs," *IEEE Communications Letters*, Submitted 2012.

[90] A. Gonzalez, J. A. Belloch, F. J. Martinez, P. Alonso, V. M. Garcia, E. S. Quintana-Orti, A. Remon, and A. M. Vidal, "The impact of the multi-core revolution on signal processing," *Waves*, vol. 2, pp. 74–85, 2010.

[91] A. Gonzalez, J. A. Belloch, G. Pinero, J. Lorente, M. Ferrer, S. Roger, C. Roig, F. J. Martinez, M. de Diego, P. Alonso, V. M. Garcia, E. S. Quintana-Orti, A. Remon, and A. M. Vidal, "Applications of multi-core and GPU architectures on signal processing: Case studies," *Waves*, vol. 2, pp. 86–96, 2010.

[92] M. Palkovic, P. Raghavan, M. Li, A. Dejonghe, L. Van der Perre, and F. Catthoor, "Future software-defined radio platforms and mapping flows," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 22–33, March 2010.

[93] J. Kim, S. Hyeon, and S. Choi, "Implementation of an SDR system using graphics processing unit," *IEEE Communications Magazine*, vol. 48, no. 3, pp. 156–162, March 2010.

[94] T. Nylanden, J. Janhunen, O. Silven, and M. Juntti, "A GPU implementation for two MIMO-OFDM detectors," in *International Conference on Embedded Computer Systems*, Samos, Greece, July 2010.

[95] M. Wu, Y. Sun, S. Gupta, and J. Cavallaro, "A GPU implementation of a real-time MIMO detector," in *IEEE Workshop on Signal Processing Systems*, Tampere, Finland, October 2009.

[96] F. J. Martínez-Zaldívar, A. M. Vidal, A. Gonzalez, and V. Almenar, "Tridimensional block multiword LDPC decoding on GPUs," *Journal of Supercomputing*, vol. 58, no. 3, pp. 314–322, December 2011.

[97] "NVIDIA CUDA C programming guide," November 2011. [Online]. Available:

http://developer.download.nvidia.com/compute/DevZone/docs
/html/C/doc/CUDA_C_Programming_Guide.pdf

[98] "NVIDIA CUDA C best practices guide," January 2012. [Online]. Available: http://developer.download.nvidia.com/compute/DevZone/docs
/html/C/doc/CUDA_C_Best_Practices_Guide.pdf

[99] M. Flynn, "Some computer organizations and their effectiveness," *IEEE Transactions on Computers*, vol. 21, pp. 948–960, September 1972.

[100] D. B. Kirk and W.-M. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. Massachusetts, CA (USA): Morgan Kaufmann Publishers, February 2010.

[101] "NVIDIA's next generation: FERMI," November 2011. [Online]. Available: http://www.nvidia.com/content/PDF/fermi_white_papers
/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

[102] V. M. Garcia, A. Gonzalez, C. Gonzalez, F. J. Martinez-Zaldivar, C. Ramiro, S. Roger, and A. M. Vidal, "The impact of GPU/multicore in signal processing: a quantitative approach," *Waves*, vol. 3, November 2011.

[103] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra, "Parallel tiled QR factorization for multicore architectures," LAPACK Working Note 190, Tech. Rep., 2007, http://www.netlib.org/lapack/lawnspdf/lawn190.pdf.

[104] Q. Qi and C. Chakrabarti, "Parallel high throughput soft-output sphere decoding algorithm," *Journal of Signal Processing Systems*, vol. 68, no. 2, pp. 217–231, August 2012.

[105] S. Roger, C. Ramiro, A. Gonzalez, V. Almenar, and A. M. Vidal, "An efficient GPU implementation of fixed-complexity sphere decoders for MIMO wireless systems," *Integrated Computed-Aided Engineering*, Submitted 2011.

[106] ——, "Fully parallel GPU implementation of a fixed-complexity soft-output MIMO detector," *IEEE Transactions on Vehicular Technology*, Submitted 2011.