HAL open science

Complexité et Performance des Récepteurs MIMO

Luis Miguel Bazdresch Sierra

To cite this version:

Luis Miguel Bazdresch Sierra. Complexité et Performance des Récepteurs MIMO. domain_other. Télécom ParisTech, 2004. English. NNT : . pastel-00001176

HAL Id: pastel-00001176

https://pastel.hal.science/pastel-00001176v1

Submitted on 5 Apr 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Complexité et performance des récepteurs MIMO

Luis Miguel Bazdresch Sierra

École Nationale Supérieure des Télécommunications, E.N.S.T.

Contents

1 Introduction **1**

2 State of the art in Space-Time Codes **5**
2.1 Problem statement 5
2.2 Definition of a MIMO system 6
2.2.1 Probability distribution of $\mathbf{n}$ 8
2.2.2 Probability distribution of h_{ij} 9
2.2.3 Equivalent Real Model 9
2.3 Performance measurements in MIMO systems 9
2.4 Rayleigh Fading in MIMO systems 12
2.5 Limits on performance and MIMO capacity 13
2.5.1 Capacity of some particular MIMO systems 14
2.5.2 Important remarks on the expected capacity and outage probabilities 17
2.6 A Note on Diversity 24
2.7 Space-Time Codes 25
2.7.1 Space-Time Trellis Codes 27
2.7.2 Space-Time Block Codes 27
2.7.3 Layered Space-Time Codes 27

3 Operating Principles of Vertical BLAST **31**
3.1 Introduction 31
3.2 V-BLAST Coding 32
3.3 V-BLAST detection: ordering, canceling, nulling 32
3.3.1 V-BLAST spectral efficiency 35
3.4 The V-BLAST reception algorithm 35
3.5 Capacity, outage probability and diversity of V-BLAST 37

4 Practical Considerations in V-BLAST Implementations **41**
4.1 Introduction 41
4.1.1 The problem with profiling 42
4.1.2 The problem with the O(·) notation 42
4.1.3 Note on simulation details 43

4.2 Computation of the Moore-Penrose pseudo-inverse 43
4.2.1 Numerical stability results 44
4.2.2 Complexity measurements 44
4.3 Complexity of thin QR V-BLAST 47
4.4 The role of L in V-BLAST complexity 50
4.4.1 Simulation results 51
4.5 V-BLAST error rate simulation results 52
4.6 Experimental results 55
4.7 V-BLAST memory requirements 57
4.8 Loss of BER performance caused by sub-optimal ordering 57

5 V-BLAST as a least-squares problem 61
5.1 Introduction 61
5.2 QR Decomposition in V-BLAST 62
5.3 The Sorted QR algorithm 64
5.4 Complexity analysis and simulation results 65
5.4.1 BLER performance 66
5.4.2 Complexity as a function of L 66
5.4.3 Complexity as a function of n_R 68
5.5 Experimental Results 71
5.6 Final words 71

6 Lattice decoding applied to vertical space-time codes 75
6.1 Introduction 75
6.2 The maximum-likelihood criterion 76
6.2.1 The ML criterion in MIMO systems 77
6.3 Applying lattice decoding to V-BLAST 77
6.3.1 Finding a lattice representation of S_M 78
6.3.2 Mapping $\hat{\mathbf{u}}$ to S 79
6.4 A lattice decoding algorithm for V-BLAST 80

7 Lattice decoding in MIMO systems: practical considerations 83
7.1 Introduction 83
7.2 Error rate comparison 84
7.3 Complexity comparisons 87
7.3.1 LLL reduction complexity 93
7.4 V-CP compared to maximum-likelihood 98
7.5 Conclusions 98

8 Conclusions 103

9 Résumé en français de la thèse **107**
9.1 Introduction . . . 107
9.2 Codes espace-temps : état de l'art . . . 108
9.2.1 Description du problème . . . 108
9.2.2 Définition d'un système MIMO . . . 108
9.2.3 Mesures de performance de systèmes MIMO . . . 109
9.2.4 Limites sur la performance et capacité des systèmes MIMO . . . 109
9.2.5 Une note sur la diversité . . . 111
9.2.6 Codes espace-temps . . . 111
9.3 Les principes de BLAST Vertical(VBLAST) . . . 113
9.3.1 Introduction . . . 113
9.3.2 Codage V-BLAST . . . 113
9.3.3 Détection V-BLAST : ordre, suppression et annulation . . . 113
9.4 Considérations pratiques dans une implantation de V-BLAST . . . 116
9.4.1 Introduction . . . 116
9.4.2 Calcul de la pseudo-inverse de Moore-Penrose . . . 116
9.4.3 Mesures de complexité . . . 117
9.4.4 Complexité de V-BLAST avec QR mince . . . 118
9.4.5 Le rôle de L dans la complexité de V-BLAST . . . 118
9.4.6 Résultats de simulation . . . 119
9.4.7 Résultats expérimentaux . . . 120
9.5 V-BLAST comme un problème de moindres carrés . . . 120
9.5.1 Introduction . . . 120
9.5.2 Décomposition QR dans V-BLAST . . . 121
9.5.3 L'algorithme QR ordonné . . . 121
9.5.4 Analyse de la complexité et résultats de simulation . . . 122
9.5.5 Taux d'erreurs par bloc . . . 122
9.5.6 Résultats expérimentaux . . . 123
9.6 Décodage de réseau de points dans le contexte des codes espace-temps verticaux . . . 123
9.6.1 Introduction . . . 123
9.6.2 L'utilisation des décodeurs de réseau dans le contexte V-BLAST 124
9.6.3 Un algorithme de décodage de réseau de points adapté à V-BLAST . . . 126
9.7 Considérations pratiques sur l'utilisation d'un décodeur de réseau de points dans un système MIMO . . . 126
9.7.1 Introduction . . . 126
9.7.2 Comparaison des taux d'erreur . . . 127
9.7.3 Comparaisons de complexité . . . 127
9.7.4 V-CP et maximum de vraisemblance comparés . . . 129
9.7.5 Commentaires . . . 129
9.8 Conclusions . . . 130

A Example of a space-time block code **131**

B A simulator platform for MIMO systems **133**

C Papers published and submitted for publication **135**

List of Figures

2.1 MIMO channel model. Boxes represent the multiplicative channel gains between antennas. Each antenna receives a linear combination of all signals transmitted plus noise. 7

2.2 Baseband block diagram of a MIMO system. 8

2.3 Expected capacity as a function of γ for $n_T = 2$ and $n_R = 2, 3, 4$. . . . 15

2.4 Expected capacity as a function of γ for $n_T = 4$ and $n_R = 4, 6, 8$. . . . 16

2.5 Expected capacity as a function of γ for $n_T = 6$ and $n_R = 6, 9, 12$. . . 16

2.6 Expected capacity as a function of γ for $n_T = 8$ and $n_R = 8, 12, 16$. . 17

2.7 Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 2$ and $n_R = 2$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 18

2.8 Complementary Cumulative Distribution Function of the capacity for $n_T = 2$ and $n_R = 3$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 18

2.9 Complementary Cumulative Distribution Function of the capacity for $n_T = 2$ and $n_R = 4$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 19

2.10 Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 4$ and $n_R = 4$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 19

2.11 Complementary Cumulative Distribution Function of the capacity for $n_T = 4$ and $n_R = 6$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 20

2.12 Complementary Cumulative Distribution Function of the capacity for $n_T = 4$ and $n_R = 8$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 20

2.13 Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 6$ and $n_R = 6$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 21

2.14 Complementary Cumulative Distribution Function of the capacity for $n_T = 6$ and $n_R = 9$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 21
2.15 Complementary Cumulative Distribution Function of the capacity for $n_T = 6$ and $n_R = 12$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 22
2.16 Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 8$ and $n_R = 8$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 22
2.17 Complementary Cumulative Distribution Function of the capacity for $n_T = 8$ and $n_R = 12$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR. 23
2.18 Complementary Cumulative Distribution Function of the capacity for $n_T = 8$ and $n_R = 16$. The SNR varies between 0 and 21 dB in steps of 3 dB. Each line represents a different SNR. 23
2.19 Block diagram of a layered space-time coder with five transmit antennas. Seven time intervals are shown. The information stream is demultiplexed into five data streams; here, a_i, $1 \leq i \leq n_T$, represent five constellation points. A layered space-time encoder layers each point in space and time; the encoding of point a_1 is highlighted. Symbols z_i represent data from the previous layer, while b_i represent data belonging to the next layer. 28

3.1 A V-BLAST transmitter in the case $n_T = 5$. 32
3.2 Capacity of V-BLAST as a function of $\alpha = n_T/n_R$, in units of bps/Hz/dimension, for $\gamma = 10$dB. 38

4.1 Total number of operations for a MIMO system with $n_T = 2$ and $n_R = 2, 4, 6$; $L = 10$. 45
4.2 Total number of operations for a MIMO system with $n_T = 4$ and $n_R = 4, 6, 8$; $L = 10$. 46
4.3 Total number of operations for a MIMO system with $n_T = 6$ and $n_R = 6, 9, 12$; $L = 10$. 46
4.4 Total number of operations for a MIMO system with $n_T = 8$ and $n_R = 8, 12, 16$; $L = 10$. 47
4.5 Behavior of O_b in V-BLAST as a function of n_R and L. The arrows indicate the direction of growth of n_R and L. 51
4.6 General shape of *BLER* as a function of O_b in a V-BLAST receiver. . 52
4.7 O_b as a function of L for a (4,4) V-BLAST receiver. 53
4.8 *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 2$, $L = 10$. 53

4.9 *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 4$, $L = 10$. 54
4.10 *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 6$, $L = 10$. 54
4.11 *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 8$, $L = 10$. 55
4.12 Instruction cycle count per received bit of a Texas Instruments 6711 DSP running V-BLAST with $n_T = 2$ and $n_R = 3$, as a function of L. . 56
4.13 Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 6$, 16-QAM. 59
4.14 Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 9$, 16-QAM. 60
4.15 Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 12$, 16-QAM. 60

5.1 *BLER* as a function of average SNR (dB) for $n_T = 2$; $n_R = 2, 3, 4$; $L = 10$; 16-QAM. 66
5.2 *BLER* as a function of average SNR (dB) for $n_T = 4$; $n_R = 4, 6, 8$; $L = 10$; 16-QAM. 67
5.3 *BLER* as a function of average SNR (dB) for $n_T = 6$; $n_R = 6, 9, 12$; $L = 10$; 16-QAM. 67
5.4 *BLER* as a function of average SNR (dB) for $n_T = 8$; $n_R = 8, 12, 16$; $L = 10$; 16-QAM. 68
5.5 O_b as a function of L for $n_T = 4$; $n_R = 4$; 69
5.6 O_b as a function of L for $n_T = 4$; $n_R = 16$; 69
5.7 O_b as a function of L for $n_T = 4$; $n_R = 100$; 70
5.8 O_b as a function of n_R; $n_T = 4$; $L = 15$ 70
5.9 Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 3$; 16-QAM . . 72
5.10 Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 6$; 16-QAM . . 72
5.11 Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 23$; 16-QAM . 73

6.1 A lattice in two dimensions. The solid points represent the lattice points. The straight lines emphasize the periodic arrangement of the points. Also shown is a point $\mathbf{r} \in \mathbb{R}^2$, and the closest lattice point. . . 76
6.2 (a) A 16-QAM constellation. (b) The same constellation translated by $(2n-1)e_1$. 79

7.1 *BLER* comparison between V-LS and V-CP; $n_T = 2$, $n_R = 2, 3,$ and 4, $L = 10$, 16-QAM. 85
7.2 *BLER* comparison between V-LS and V-CP; $n_T = 4$, $n_R = 4, 6,$ and 8, $L = 10$, 16-QAM. 85

7.3 *BLER* comparison between V-LS and V-CP; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM. 86
7.4 *BLER* comparison between V-LS and V-CP; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM. 86
7.5 *BER* comparison between V-LS and V-CP; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM. 87
7.6 *BER* comparison between V-LS and V-CP; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM. 88
7.7 *BER* comparison between V-LS and V-CP; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM. 88
7.8 *BER* comparison between V-LS and V-CP; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM. 89
7.9 O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM 89
7.10 O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM 90
7.11 O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM 90
7.12 O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM 91
7.13 O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 8$, $n_R = 8$, 12, $L = 10$, 16-QAM 93
7.14 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM 94
7.15 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM 94
7.16 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM 95
7.17 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM 95
7.18 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 100$, 16-QAM 96
7.19 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 100$, 16-QAM 96
7.20 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 100$, 16-QAM 97
7.21 Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 100$, 16-QAM 97
7.22 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 2$, 16-QAM 99
7.23 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 3$, 16-QAM 99

7.24 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 4$, 16-QAM 100
7.25 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 4$, 16-QAM 100
7.26 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 6$, 16-QAM 101
7.27 *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 8$, 16-QAM 101

A.1 Block diagram of a space-time block encoder. $n_T b$ information bits are modulated in groups of b bits into vector $\mathbf{a}$. This vector is used to construct codeword matrix $\mathbf{A}$, whose rows are then transmitted one at a time. 132

List of Tables

3.1 Maximum value of V-BLAST capacity for several values of average SNR. 37
3.2 Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = n_R = 4$. 38
3.3 Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = n_R = 8$. 39
3.4 Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = 4$, $n_R = 12$. 39

4.1 Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 2$ and $L = 10$. 48
4.2 Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 4$ and $L = 10$. 49
4.3 Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 6$ and $L = 10$. 49
4.4 Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 8$ and $L = 10$. 49
4.5 Clock cycles and data rates as a function of L. Results for instruction cycle $I_c = 6.7$ns were obtained by running V-BLAST with $n_T = 2$ and $n_R = 3$ on a Texas Instruments 6711 DSP. Results for $I_c = 3.3$ns are an extrapolation to a current-generation DSP. 56
4.6 Matrices required by V-BLAST. A description of their function, their dimensions, and whether they are real or complex. 58

5.1 Summary of conditions under which each algorithm is a better option than the other. 71

List of Algorithms

1 Original V-BLAST 36
2 V-BLAST 36
3 V-LS 63
4 V-SQR 65
5 ClosestPoint 81
6 V-CP 82

Résumé

Le codage espace-temps est une technique qui permet d'exploiter de façon très efficace la diversité spatiale et temporelle présente dans certains systèmes de communication, dont le canal sans fil. Le principal avantage de cette technique est une très grande efficacité spectrale. Dans nos jours, où le canal radio-mobile est de plus en plus utilisé pour transmettre tout type d'information, les méthodes permettant une utilisation plus efficace du spectre électromagnétique ont une importance fondamentale.

Les algorithmes de réception connus aujourd'hui sont très complexes, même en ce qui concerne les codes espace-temps les plus simples. Cette complexité reste l'un des obstacles principaux à l'exploitation commerciale de ces codes.

Cette thèse présente une étude très détaillée de la complexité, la performance, et les aspects les plus intéressants du comportement des algorithmes de réception pour des codes espace-temps, étude qui présente un moyen rapide pour une éventuelle conception des architectures adaptées à ce problème.

Parmi les sujets présentés dans cette thèse, une étude approfondie de la performance de ces algorithmes a été réalisée, ayant pour objectif d'avoir une connaisance suffisante pour pouvoir choisir, parmi le grand nombre d'algorithmes connus, le mieux adapté à chaque système particulier. Des améliorations aux algorithmes connus ont aussi été proposées et analysées.

Acknowledgements

This thesis bears my name on the cover, but it is the result not only of my work, but also that of many others who have helped or supported me along the way. Without them, the present work would not have been possible.

The financial and logistical support received from the Mexican government (through both CONACYT and their representatives in France, SFERE) was instrumental. Without it, I would not have been able to dedicate these last four years to this work.

I wish to thank the École Nationale Supérieure des Télécommunications, E.N.S.T. Paris, for opening their doors to me, and for making all their resources available so that my work could be developed under the best conditions.

A very special, heartfelt "Thank You" to my thesis advisor, Georges Rodríguez-Guisantes. I have more reasons to be thankful to him than space in these pages; I will mention only two. He personally undertook to guide me in this work in spite of the uncertainty and risk involved; and through his good humor, outlook on life, and optimism, he taught me more than just how to be a scientist. ¡Gracias Jorge!

Je voudrais indiquer ma reconaissance aux rapporteurs et examinateurs de ma thèse: M. Arturo Veloz, M. Jean-François Diouris, M. Bernard Huyart, et M. Patrick Loumeau. Leurs commentaires et questions ont été très importants pour m'aider à mieux comprendre l'ensemble de problèmes autour d'une implantation pratique des codes espace-temps.

Agradezco también a mis papás y hermanos, y a toda mi familia; su compañía desde lejos, su apoyo constante y su interés ininterrumpido me fueron muy importantes para superar esta prueba.

A mi esposa amada no solamente le agradezco desde el fondo de mi corazón su amor, su ayuda, su apoyo y su compañía; también le dedico esta tesis y estos cuatro largos años de trabajo. ¡Gracias Rebe!

Acronyms

All acronyms used in this thesis are listed below.

AWGN: Additive, white Gaussian noise.
BER: Bit-error rate.
BLAST: Bell Labs architecture for space-time codes.
BLER: Block-error rate.
CSI: Channel-side information.
DSP: Digital signal processor.
KZ: Korkine-Zolotareff basis reduction.
LLL: Lenstra-Lenstra-Lovász basis reduction.
LSTC: Layered space-time codes.
MEA: Multiple-element array.
MIMO: Multiple-input, multiple-output.
MFLOPS: Millions of floating-point operations per second.
ML: Maximum likelihood.
MMSE: Minimum mean-squared error.
MPPI: Moore-Penrose pseudo-inverse.
QAM: Quadrature amplitude modulation.
SNR: Signal-to-noise ratio.
STBC: Space-time block code.
STTC: Space-time trellis codes.
SVD: Singular-value decomposition.
V-BLAST: Vertical Bell Labs architecture for space-time codes.
V-CP: Vertical Closest Point. A receiver algorithm for vertical codes based on the ClosestPoint lattice decoder algorithm.
V-LS: Vertical Least Squares. A receiver algorithm for vertical codes, based on V-BLAST, that employs least-squares techniques.
V-SQR: Vertical Sorted QR. A modification of V-BLAST that uses the sorted QR decomposition.

Chapter 1

Introduction

We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.

Charles Anthony Richard Hoare

EXCEPT FOR the written word, human communication has always been transmitted over the atmosphere. Voice, sounds, smoke signals, etc. use mechanical or visual cues to transmit messages.

With the advent of telephony and telegraphy, long-distance communications became possible, inexpensive and reliable. These techniques used electrical wire as the communication medium with great success: thousands of homes and cities all over the world deployed hundreds of thousands of kilometers of cable to allow people to communicate.

However, it was soon determined that the exclusive use of wires severely limited the use and flexibility of the new communication systems. From an economic perspective, providing each home with a wired telephone cable is very expensive. Replacing the wire infrastructure when it fails is even more so. From a practical point of view, the end point of the wire is, by necessity, fixed; this means that, in order to profit from it, a person has to be present in a specific place. Also, wired communications are essentially one-to-one; sharing the same connection between several people is, at best, cumbersome. Another limiting factor is the reduced amount of data that can be transmitted over long distances.

Over the last century, enormous advances have been made in telecommunications. Ever greater amounts of information have been made available to the public from a variety of sources: radio, television, internet, multimedia mobile phones. The old wire seems to be ill-suited to the new possibilities offered by technology. In fact, more and more information is transmitted wirelessly, returning to the ancient use of the atmosphere as communication medium.

The reasons are simple; in short, wireless communications overcome the basic limitations described above. There is no need to lay out huge amounts of material

to homes, businesses and offices. The transmitted signal can cover a large area (even the whole planet), allowing the use of the medium from any location, and making broadcasting trivially easy.

One of the most important properties of the atmosphere as a communication medium is its huge capacity to transfer information. Hundreds of television and radio channels and thousands of telephone conversations can be transmitted with relative ease. As the amount of information that is generated increases, along with the desire (and the ability) to obtain it and share it, even this medium can become saturated. In developed countries, spectrum allocation (the licensing of specific frequencies for specific uses) is considered just as important, or more, than any other national infrastructure problem.

As the spectrum becomes scarce, the economy starts to play a role as well. Frequency bands become coveted goods, much like any other scarce resource. There is a risk that, as with the telephone cable, the wireless channel will become inflexible and expensive, and ultimately unable to meet the growing requirements and needs of society.

Fortunately, there is evidence that the perceived spectrum scarcity is artificial, because it is being used inefficiently. As digital communications supplant the old analog signals, new, powerful techniques can be used to increase the spectral efficiency, defined as the number of information bits that are transmitted per hertz of bandwidth, per second. One of the most promising techniques is a new type of channel codes called *space-time codes*.

One essential feature of the wireless channel is called *fading*, which occurs when the signal follows multiple paths between the transmit and receive antennas. Under certain, not uncommon conditions, the arriving signals will add up destructively, reducing the received power to zero (or very near zero). No communication is possible under such circumstances.

Fading is countered with *diversity*, in which the information is transmitted not once but several times, in the hope that at least one of the replicas will not be faded (or that what little information is conveyed by each replica can be used to estimate the original signal).

Diversity makes use of another essential property of the wireless channel: it can be modeled as a large number of independent, separate channels. These channels can be separate in the frequency they use, in the particular time intervals in which they are used, in the path they follow from the transmitter to the receiver, or in any other of a variety of methods.

Space-time codes exploit diversity not only to combat fading but to increase spectral efficiency while reducing error rates and remaining power efficient. To do so, they combine space diversity (the use of multiple antennas to create multiple signal paths) with time diversity (the "spreading" of each information bit to multiple time instants). The result is unprecedented spectral efficiencies.

In the last few years the research community has made a tremendous effort to understand space-time codes, their performance and their limits. It has realized

that neither the design of such codes nor the design of efficient receiver algorithms is an easy task.

Finding receiver algorithms that can be implemented efficiently and reliably, and that make economic sense, is crucial if the promise of space-time codes is to be brought to fruition. An algorithm or a code design that is theoretically very powerful, but is so complex that it is unlikely ever to leave the paper it's written on, is not very useful in solving the spectrum availability problems described above (or any other practical communications problem).

This thesis confronts the problem of receiver design. One particular type of space-time code, called *vertical layered codes*, has been selected for study; its performance has been evaluated and several possible receiver algorithms have been compared and scrutinized.

This thesis consists of seven chapters and two appendices. Chapter 2 presents the theoretical foundation on which space-time codes are built, and gives some details of their implementation.

Chapter 3 concentrates on the space-time codes known as vertical layered space-time codes (or vertical codes for short). This type of code is the main subject of this thesis; its principles of operation are presented in this chapter.

Chapter 4 presents the practical problems of implementing and using vertical codes. A receiver algorithm proposed by Bell Labs, called V-BLAST, is studied in detail, and results regarding its error-rate performance and computational complexity are obtained, along with some observations on its limitations and its behavior under varying circumstances.

A modification of V-BLAST is presented in chapter 5. This modification, called V-LS, borrows ideas from least-squares theory; it is shown to have the same error rate as V-BLAST, but much lower complexity. V-LS is compared to another algorithm, proposed recently in the literature on the subject, that also uses least-squares concepts, but in a different way.

Chapter 6 studies a very different kind of receptor than V-BLAST. Lattice decoding, a method that solves the closest point problem in a lattice, is adapted for use in the reception of vertical codes. The resulting algorithm, called V-LD, is close to optimal.

Finally, chapter 7 presents results on the error-rate performance and computational complexity of V-LD, and a comparison is made with V-LS. V-LD is found to exhibit peculiar behavior under some circumstances, which are explored and explained.

Appendix 1 presents an example of a space-time block code, as a supplement to chapter 2. Appendix 2 introduces the simulator platform used to generate all the simulation results presented in this thesis. In addition, a list of all acronyms used is provided.

Chapter 2

State of the art in Space-Time Codes

2.1 Problem statement

WIRELESS data communication systems can be broadly divided in two types, according to the signal frequencies and types of antennas they use. In the first type, the antennas' positions are fixed and the surrounding environment is more or less stable. In this type of systems, the transmitted energy can be concentrated in a single ray aimed at the receiving antenna.

In the second type of system, at least one antenna (of the transmitter-receiver pair) is mobile and its position unknown. Furthermore, the environment surrounding the mobile antenna is subject to continuous change. It is this type of system that we are interested in.

The mobility of the antennas (or, equivalently, the continuous movement of objects around and between the antennas) and the consequent uncertainty about their position means that the transmitted energy must be radiated to a large section of space. The main consequence of this, along with a reduction in received power, is the appearance of a phenomenon known as *multipath*: the transmitted signal travels through multiple paths before arriving at the receiving antenna. Each of these multiple reflections arrives at different times and is subject to different attenuations than the others; when the reflections add destructively, a phenomenon known as *fading* appears.

Fading can drastically reduce the received power. If reliable communication over wireless channels is to be achieved, measures must be put in place to counteract its effects. The most common technique against fading is to transmit many replicas of the original signal, in the hope that at least one of them will not fade; this technique is known as *diversity*. Diversity has the drawback of causing inefficiencies in the system, since at least part of the available resources must be used to send the signal replicas. This introduces redundancy.

In recent years, researchers have realized that multipath, as well as giving rise

to fading, can help to combat it. Multiple reflections are a natural phenomenon in wireless channels, and they can be harnessed to provide diversity; no other system resource such as bandwidth needs to be employed. Furthermore, it has been determined that the capacity available in wireless channels, at least a fraction of which can be attained with clever codes and receivers, is unprecedented.

These new techniques use the properties of the wireless channel not only to provide diversity, but also promise to deliver high data rates at low signal power, enabling new communication applications. The design, performance, and complexity of some of these techniques are the subject of this thesis.

2.2 Definition of a MIMO system

The generic term used to denote communications systems that employ multiple antennas at both the transmitter and receiver ends is *multiple-input, multiple-output* (MIMO) systems (also known as *multi-element arrays*, or *MEA*).

In a MIMO system n_T antennas are used to transmit and n_R antennas are used to receive. The data to be transmitted is divided into n_T streams, and each stream is fed to a different antenna. The n_T transmit antennas are symbol-synchronized, use the same frequency band, and the same signal constellation $S = \{s_1, s_2, \cdots, s_{2^b}\}$; b is defined as the number of information bits carried by each signal in S. The constellation is assumed to be bi-dimensional. The average symbol energy is denoted by E_s. All symbols are equally likely.

In general, a MIMO system can have any number of transmit and receive antennas. Except for a few sections in this chapter where it is indicated otherwise, in most of this thesis it is assumed that $n_R \geq n_T$. This assumption is required by the receivers studied in the following chapters; although it is conceivable that they could be modified to work with $n_R < n_T$, such modifications are beyond the scope of this thesis.

A MIMO system with n_T transmit antennas and n_R receive antennas will frequently be referred to simply as a (n_T, n_R) MIMO system.

The channel is assumed to be frequency-flat with slow Rayleigh fading, and is represented by a matrix $\mathbf{H}$, which has n_R rows and n_T columns. Element h_{ij} is the transfer function from transmitter j to receiver i (see figure 2.1). The elements of $\mathbf{H}$ are assumed to be complex Gaussian independent and identically distributed (i.i.d.) random variables of zero mean and variance 0.5 per dimension. The channel is assumed to be constant during the transmission of a *block* (also sometimes referred to as a *frame*) of size $L \times n_T$ symbols (for L an integer greater than zero), and to change from one block to another. Also, it is assumed that the channel is memoryless between blocks; that is, matrices associated with different blocks are statistically independent. Such a channel is known as a frequency-flat, slow fading channel, or simply as a *block-fading* channel [1]. These characteristics are typical in fixed wireless applications, where some slow channel variations are expected; an example

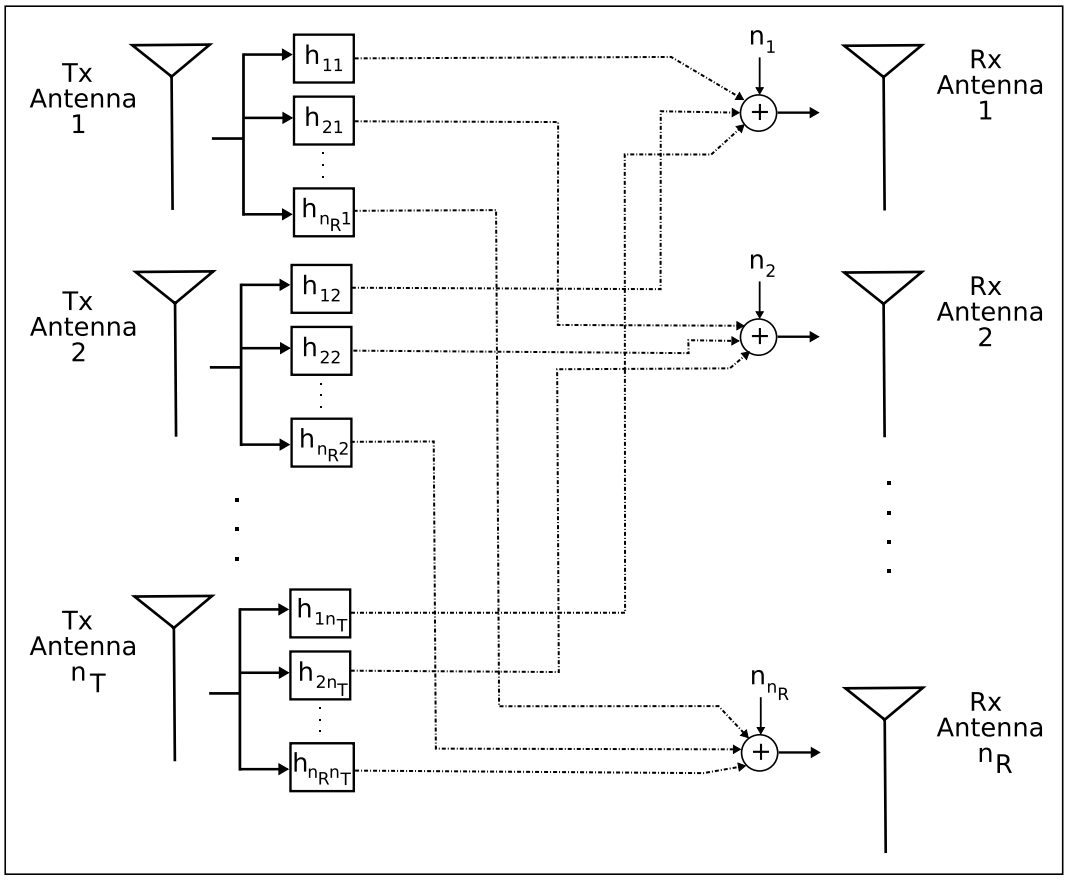

Figure 2.1: MIMO channel model. Boxes represent the multiplicative channel gains between antennas. Each antenna receives a linear combination of all signals transmitted plus noise.

would be an office environment where people constantly move around at walking speed.

Matrix $\mathbf{H}$ is assumed to be full-rank, that is, its rank is equal to n_T. This is justified because the probability of a randomly generated matrix presenting non-independent rows and columns is very close to zero. In practice, this means that the receiver antennas must be adequately spaced. This requirement is not considered unreasonable in modern wireless applications where the carrier frequency is in the range of a few gigahertz and thus the required separation would be a few centimeters.

Each receiver is assumed to have estimated $\mathbf{H}$ perfectly through the use of some appropriate method, such as a training sequence transmitted with each block. This situation is frequently described in the literature as the receiver having perfect *channel-state information* (*CSI*).

Let $\mathbf{a} = (a_1, a_2, ..., a_{n_T})^T$ denote the vector of transmitted symbols; all elements of $\mathbf{a}$ are elements of the same signal constellation S, and all have the same average power $E_s = \mathcal{E}[|a_i|^2]$, $1 \leq i \leq n_T$. The received vector $\mathbf{r}$ can be expressed as:

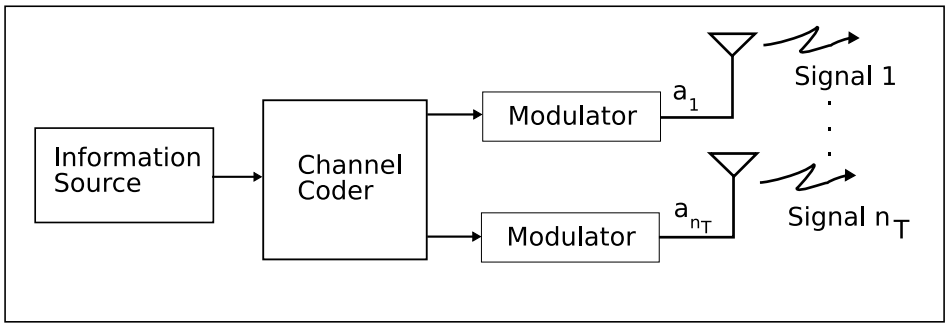

Figure 2.2: Baseband block diagram of a MIMO system.

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n}, \tag{2.1}$$

where $\mathbf{n}$ is a noise vector whose elements are Gaussian circularly-symmetric i.i.d. complex random variables of zero mean and variance N_0 per dimension.

Vector $\mathbf{a}$ is generated by a channel coder (see figure 2.2). Codes designed specifically for MIMO systems are discussed below.

2.2.1 Probability distribution of n

A complex random vector $\mathbf{n}$ is said to be Gaussian if the real random vector

$$\tilde{\mathbf{n}} = \begin{bmatrix} \Re(\mathbf{n}) \\ \Im(\mathbf{n}) \end{bmatrix}$$

is Gaussian, where $\Re(\mathbf{n})$ and $\Im(\mathbf{n})$ are the real and imaginary parts of $\mathbf{n}$, respectively.

To determine the distribution of vector $\tilde{\mathbf{n}}$, its expectation and covariance matrix must be specified. Let $\mathbf{A}^\dagger$ denote the adjoint matrix of $\mathbf{A}$, and $\mathcal{E}[\cdot]$ denote the expected value. If the covariance matrix of $\tilde{\mathbf{n}}$ has the form

$$\mathcal{E}[(\tilde{\mathbf{n}} - \mathcal{E}[\tilde{\mathbf{n}}])(\tilde{\mathbf{n}} - \mathcal{E}[\tilde{\mathbf{n}}])^\dagger] = \frac{1}{2}\begin{bmatrix} \Re(\mathbf{Q}) & -\Im(\mathbf{Q}) \\ \Im(\mathbf{Q}) & \Re(\mathbf{Q}) \end{bmatrix},$$

where $\mathbf{Q} \in \mathbb{C}^{n_R \times n_R}$ is a Hermitian non-negative definite matrix, then $\mathbf{n}$ is said to be *circularly symmetric*. In this case, the covariance matrix of $\mathbf{n}$ is given by $\mathbf{Q}$.

Since each element of $\tilde{\mathbf{n}}$ is independent of the others, then its covariance matrix has the form:

$$\mathbf{Q}_{\tilde{\mathbf{n}}} = I_{2n_R} \cdot N_0,$$

where $I_{2n_R} \in \mathbb{R}^{2n_R \times 2n_R}$ is the identity matrix; in consequence, the noise in a MIMO system defined above is circularly symmetric. Its mean value is the same as that of $\tilde{\mathbf{n}}$ (zero), and its covariance matrix is given by

$$\mathbf{Q}_{\mathbf{n}} = I_{n_R} \cdot 2N_0.$$

2.2.2 Probability distribution of h_{ij}

The probability density function of a random complex variable can be specified as the joint density function of its real and imaginary parts. In the case of the elements of $\mathbf{H}$, h_{ij}, $1 \leq i \leq n_R$, $1 \leq j \leq n_T$, both its real and imaginary parts are independent Gaussian random variables of zero mean and variance 0.5 per dimension. Let $h_R = \Re(h_{ij})$ and $h_I = \Im(h_{ij})$. The probability density function of h_{ij} is then given by

$$\begin{aligned} p(h_{ij}) &= p(h_R) \cdot p(h_I) \\ &= \frac{\exp(-h_R^2)}{\sqrt{2\pi}\sqrt{1/2}} \cdot \frac{\exp(-h_I^2)}{\sqrt{2\pi}\sqrt{1/2}} \\ &= \frac{\exp(-|h_{ij}|^2)}{\pi}. \end{aligned}$$

Each element $r_i, i = 1, 2, \ldots, n_R$ of $\mathbf{r}$ is a different linear combination of the transmitted vector $\mathbf{a}$ plus noise. The coefficients of the linear combinations are determined by the rows of $\mathbf{H}$:

$$r_i = h_{i,1}a_1 + h_{i,2}a_2 + \cdots + h_{i,n_T}a_{n_T} + n_i.$$

2.2.3 Equivalent Real Model

The system model described by equation (2.1) can be written using only real numbers as follows:

$$\begin{bmatrix} \Re(\mathbf{r}) \\ \Im(\mathbf{r}) \end{bmatrix} = \begin{bmatrix} \Re(\mathbf{H}) & -\Im(\mathbf{H}) \\ \Im(\mathbf{H}) & \Im(\mathbf{H}) \end{bmatrix} \begin{bmatrix} \Re(\mathbf{a}) \\ \Im(\mathbf{a}) \end{bmatrix} + \begin{bmatrix} \Re(\mathbf{n}) \\ \Im(\mathbf{n}) \end{bmatrix}.$$

Let vector $\tilde{\mathbf{n}}$ be defined as above, and matrix $\tilde{\mathbf{H}}$ be defined as

$$\tilde{\mathbf{H}} = \begin{bmatrix} \Re(\mathbf{H}) & -\Im(\mathbf{H}) \\ \Im(\mathbf{H}) & \Im(\mathbf{H}) \end{bmatrix}.$$

Then the equivalent real system can be written in more compact form as:

$$\tilde{\mathbf{r}} = \tilde{\mathbf{H}}\tilde{\mathbf{a}} + \tilde{\mathbf{n}}. \tag{2.2}$$

In this model, $\mathcal{E}[\tilde{h}_{ij}^2] = 1/2$, $\tilde{E}_s = \mathcal{E}[\tilde{a}_i{}^2]/2$ and $\mathcal{E}[\tilde{\mathbf{n}}^2] = N_0$. Functionality-wise, the real and complex models are identical.

2.3 Performance measurements in MIMO systems

The error-rate performance of MIMO systems can be measured in terms of *bit-error rate* (*BER*), as in conventional single-antenna communications systems. Since

the properties of the channel naturally divide the data stream in blocks, it is also interesting to study the block error rate (*BLER*). A block error is defined as the occurrence of at least one bit error within a block. Most results in the literature are presented in terms of *BLER*.

From the point of view of the network applications in which MIMO systems will eventually be used, availability of *BLER* measurements is potentially more interesting than just the bit-error rates. The reason is that the network could be configured to tailor its packet or datagram size to fit a block; in this case, the block error rate provides an indication of how often packets would have to be retransmitted.

The probability of error is usually calculated as a function of *average SNR* (denoted by γ), defined as the ratio of the received power at each antenna and the variance of the noise affecting each component of the received vector:

$$\gamma = \frac{\mathcal{E}[|\mathbf{Ha}|^2]}{\mathcal{E}[|\mathbf{n}|^2]}, \tag{2.3}$$

where $|\cdot|$ is the ℓ^2-norm. Let $\mathbf{s} = \mathbf{Ha}$; then,

$$\begin{aligned}
\mathcal{E}[|\mathbf{s}|^2] &= \mathcal{E}\left[\sum_{k=1}^{n_R} |s_k|^2\right] \\
&= \mathcal{E}\left[\sum_{k=1}^{n_R}\sum_{j=1}^{n_T} |h_{kj}a_j|^2\right] \\
&= \sum_{k=1}^{n_R}\sum_{j=1}^{n_T} \mathcal{E}[|h_{kj}|^2]\cdot\mathcal{E}[|a_j|^2] \\
&= E_s \cdot n_T \cdot n_R. \qquad (2.4)
\end{aligned}$$

Recall that the expected value of each element of $\mathbf{a}$ is E_s, and that of the elements of $\mathbf{H}$ is 1. Given that $\mathcal{E}[|\mathbf{n}|^2] = 2 \cdot n_R \cdot N_0$, substituting (2.4) into (2.3), the average SNR can be written as

$$\gamma = \frac{E_s \cdot n_T}{2N_0}, \tag{2.5}$$

which, as defined above, is the average power in each receiver antenna divided by the average noise power in that antenna. The factor 2 in the average power of the noise appears because the real and imaginary parts of $\mathbf{n}$ each have variance N_0; consequently, the variance of each component of $\mathbf{n}$ is $2N_0$ (see the paragraph above where the distribution of $\mathbf{n}$ was determined).

There are two values of E_s that are frequently used in the literature. One value is $E_s = 1/n_T$, which means that the transmitted power is independent of the number of transmitter antennas. This corresponds to a situation where the system is provided with an amplifier of fixed power, which is shared equally by all antennas. In this case, the average SNR is given by $\gamma = 1/(2N_0)$.

The other common alternative is to make $E_s = 1$. This model is arguably less realistic; however, in this case the signal constellation S is constant and independent of n_T, which makes the design of a simulator much easier. For this reason, it is the model that has been chosen for this thesis. When a value of $E_s = 1$ is used, the average SNR is given by $\gamma = n_T/(2N_0)$.

It should be noted that there is a common alternative view of the system where $E_s = 1$, $\mathcal{E}[|\mathbf{n}|^2] = 1$, and the channel matrix is normalized as follows:

$$\mathbf{B} = \sqrt{\frac{\beta}{n_T}}\mathbf{H}.$$

In this model, the received vector $\mathbf{r} = \mathbf{Ba} + \mathbf{n}$; the average SNR can be found as follows:

$$\begin{aligned}
\gamma &= \mathcal{E}[|\mathbf{Ba}|^2] \\
&= \mathcal{E}[\frac{\beta}{n_T}|\mathbf{s}|^2] \\
&= \frac{\beta \cdot n_T \cdot n_R}{n_T} \\
&= \beta \cdot n_R.
\end{aligned}$$

Different refinements or variations of the definitions can be used to meet different modeling requirements. It should be emphasized that for a given average SNR they are all equivalent, since they are just different forms of equation (2.3). Regardless of the particular model of transmit power, noise, and channel matrix chosen, the average SNR has the same meaning for all of them.

It is common in single-antenna systems to use the ratio of energy per information bit and the noise power spectral density, E_b/N_0, to measure performance. The main advantage of using the average SNR instead of E_b/N_0 in MIMO systems is that the average SNR is essentially independent of the number of antennas; it measures only the signal strength and the noise strength at each receive antenna. Varying the number of antennas while keeping the average SNR constant makes evident the gain or loss of performance due to the number of antennas alone; this is harder to measure when keeping E_b constant. Among other things, a model based on the energy per bit would have to make explicit the way this energy is distributed by the channel among the receive antennas.

Also, the use of average SNR is more common in the literature, making easier the comparison between results presented in this thesis and those published elsewhere. For these reasons, average SNR is used throughout this thesis.

Finally, regarding the real model proposed in equation (2.2), since $\mathcal{E}[\hat{h}_{ij}^2] = 1/2$, $\hat{E}_s = \mathcal{E}[\hat{a}_i{}^2] = E_s/2$ and $\mathcal{E}[|\hat{\mathbf{n}}|^2] = 2 \cdot n_R \cdot N_0$, then $\mathcal{E}[(\hat{\mathbf{H}}\hat{\mathbf{a}})^2] = E_s \cdot n_R \cdot n_T$ and $\gamma = (E_s \cdot n_T)/(2 \cdot N_0)$ as expected.

2.4 Rayleigh Fading in MIMO systems

In the definition of the channel matrix $\mathbf{H}$ above, it has been assumed that each element h_{ij} of $\mathbf{H}$ is complex, with its real and imaginary parts being Gaussian random variables with zero mean and variance 0.5. This assumption is now justified.

Consider a communications system with a single transmitter and receiver, and a fading channel where the signal received is the sum of a large number of delayed and attenuated versions of the original signal. It is assumed that each signal path has constant attenuation and delay during a time period T; this kind of channel is referred to as a *slow, frequency-nonselective* channel: it does not change the frequency spectrum of the transmitted signal, and subjects all symbols transmitted during time period T to the same attenuations.

The attenuation and delay introduced by each path can be expressed as a complex number α_n, where n is an index that runs from 1 to some large number. During the time period $0 \leq t \leq T$, if the transmitted signal is $s(t)$, the received signal $r(t)$ is given by:

$$\begin{aligned} r(t) &= s(t) \cdot \sum_n \alpha_n \\ &= s(t) \cdot \left[\sum_n \Re(\alpha_n) + j \sum_n \Im(\alpha_n) \right]. \end{aligned}$$

Now let $\beta = \sum_n \Re(\alpha_n) + j \sum_n \Im(\alpha_n)$:

$$r(t) = s(t) \cdot \beta. \tag{2.6}$$

The real and imaginary components of each α_n are random variables with unknown distribution. However, the central limit theorem implies that both $\Re(\beta)$ and $\Im(\beta)$ are closely approximated as Gaussian random variables.

In the absence of a direct line-of-sight link between the antennas (which is to say, of a non-fading component in $r(t)$), the expected value of β is zero.

A complex number can be written in polar form as:

$$\beta = R \cdot e^{i\theta}.$$

If the real and imaginary parts of β are Gaussian-distributed with zero mean, then R has a Rayleigh distribution and θ is uniformly distributed in the real interval $[0, 2\pi)$. R is the envelope of the received signal $r(t)$, which is why channels that are described by equation (2.6) are called Rayleigh-fading channels.

The assumptions made on the elements of channel matrix $\mathbf{H}$ are justified by repeating the reasoning above for each pair of transmit and receive antennas. Recent results regarding the validity of these assumptions are presented in [2].

2.5 Limits on performance and MIMO capacity

The problem of estimating the capacity of a MIMO system has been solved in by several authors in [3, 4, 5, 1, 6]. Capacity depends on the signal-to-noise ratio as given in equation (2.5). The average SNR will be denoted by γ[1]. Capacities are given in bits/second/Hz (or bps/Hz).

For comparison purposes, the capacity of a system with a single transmit and a single receive antenna, with additive white Gaussian noise and Rayleigh fading, is studied first. The capacity of such a system is given by

$$C = \log(1 + \gamma|h|^2). \tag{2.7}$$

Here h is a complex scalar known at the receiver. For high SNR, an increase of 3dB in γ gives a gain in spectral efficiency of one bps/Hz.

Equation (2.7) represents a system where all available energy is transmitted through a single channel that exists between transmitter and receiver. Let us consider a system where, without increasing the transmitter power, n signals are transmitted through n independent and uncoupled channels. This means $\mathbf{H} = I_n$. In such a case, the capacity is given by

$$C_n = n \cdot \log(1 + (\gamma/n)). \tag{2.8}$$

The importance of equation (2.8) cannot be stressed enough, because it provides the justification for all subsequent work on MIMO systems. Just by using many channels instead of only one, capacity has increased with respect to that provided by equation (2.7). In addition, as the number of channels grows, the capacity tends to $\gamma/\ln(2)$, which grows linearly with γ.

Naturally, in the general case $\mathbf{H} \neq I_n$ and the channel presents fading and interference between the signals. These problems can be dealt with through the use of coding or other schemes designed to counteract them. The promise of a very large capacity is there if such schemes can be found, and that is in large part the subject of this thesis.

The general capacity expression for a MIMO system is

$$C = \mathcal{E}[\log\det(I_{n_R} + (\gamma/n_T) \cdot \mathbf{H}\mathbf{H}^\dagger)]. \tag{2.9}$$

It should be noted that, defined in such a way, the capacity is a random variable. In equation (2.9) it has been assumed that the channel is memoryless and $\mathbf{H}$ changes every time the channel is used. The capacity can be evaluated to be [3]:

$$C = \int_0^\infty \log(1 + \gamma\lambda/n_T) \sum_{k=0}^{n_T-1} \frac{k!}{(k + n_R - n_T)!} [L_k^{n_R-n_T}(\lambda)]^2 \lambda^{n_R-n_T} e^{-\lambda} d\lambda,$$

[1] All logarithms in the following equations are base 2.

where

$$L_k^l(x) = \frac{1}{k!} e^x x^l \frac{d^k}{dx^k}(e^{-x} x^{l+k})$$

are the associated Laguerre polynomials of order k.

For a fixed n_R, as n_T grows $(1/n_T) \cdot \mathbf{HH}^\dagger \to I_{n_R}$. Then, as n_T grows the capacity will tend to

$$\begin{aligned} \lim_{n_T \to \infty} C &= n_R \log(1+\gamma) \\ &= n_R \log\left(1 + \frac{1}{n_R} \cdot (\gamma \cdot n_R)\right). \end{aligned} \tag{2.10}$$

Comparing equation (2.10) with (2.8), it is clear that increasing the number of transmit antennas has a similar effect to decoupling the receive antennas. Note that this is achieved with no increase in the average SNR. Equation (2.10) is that of n_R independent, non-fading paths, each with average SNR equal to $\gamma \cdot n_R$.

On the other hand, when n_T is fixed and n_R grows, the capacity is given by

$$\begin{aligned} \lim_{n_R \to \infty} C &= n_T \log(1 + \gamma \frac{n_R}{n_T}) \\ &= n_T \log\left(1 + \frac{1}{n_T} \cdot (\gamma \cdot n_R)\right). \end{aligned} \tag{2.11}$$

Again, comparing equations (2.11) and (2.8), it becomes apparent that letting n_R grow ultimately achieves the same capacity as that of a channel with n_T independent, non-fading paths, each with average SNR equal to γn_R.

In the general case where the channel remains constant during the transmission of L symbol periods and then changes, the capacity has been determined to be given by [4]:

$$C_b = L \cdot \mathcal{E}[\log\det(I_{n_R} + (\gamma/n_T) \cdot \mathbf{HH}^\dagger)], \tag{2.12}$$

where C_b is the capacity associated with a block.

2.5.1 Capacity of some particular MIMO systems

In the following chapters, several MIMO systems with specific antenna configurations will be studied. These range from small ($n_T = n_R = 2$) to relatively large ($n_T = 8$, $n_R = 16$). Specifically, four possible transmit antennas have been selected: $n_T = 2$, 4, 6 and 8. For each transmit antenna, three different numbers of receive antennas are used: $n_R = n_T$, $n_R = 1.5 \cdot n_T$, and $n_R = 2 \cdot n_T$. The purpose in selecting these specific sets of antennas is twofold: first, to study small to medium-sized systems, which are those that are more likely to find practical applications in the short

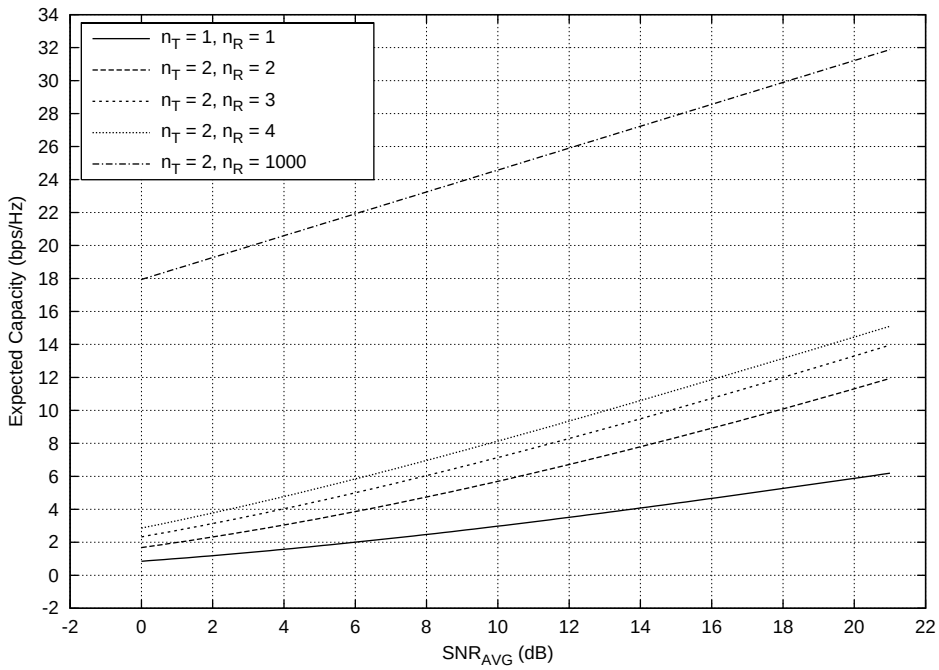

Figure 2.3: Expected capacity as a function of γ for $n_T = 2$ and $n_R = 2, 3, 4$.

to medium term; second, to explore how the receiver performance changes when n_R grows.

Following the analysis made by Foschini [5], the expected capacities of these systems have been calculated numerically using equation (2.9), and are shown in figures 2.3 through 2.6. The method employed to estimate the expected capacities was Monte-Carlo simulation: 10,000 realizations of $\log\det(I_{n_R} + (\gamma/n_T) \cdot \mathbf{HH}^\dagger)$ were calculated for each combination of n_T, n_R, and γ, and then the mean value was found.

For purposes of comparison, each figure also includes the expected capacities of a single-antenna system, calculated using equation (2.7), and that obtained by setting $n_R = 1000$, calculated using (2.11).

The expected capacity is not the only, nor the most useful, measurement when assessing the potential of a fading channel. It provides an indication of how large the capacity of MIMO systems is, but it gives no information about the dynamics of the channel. Therefore, achieving the expected capacity may prove to be challenging from a practical point of view, since the transmitter would somehow have to follow these dynamics.

Of more practical interest is the *probability of outage* $P_{out}(C_{th})$. The probability of outage of a fading channel is the probability that the capacity of the channel at any time is less than C_{th}, for given n_T, n_R, and γ.

$$P_{out}(C_{th}) = P(C < C_{th}).$$

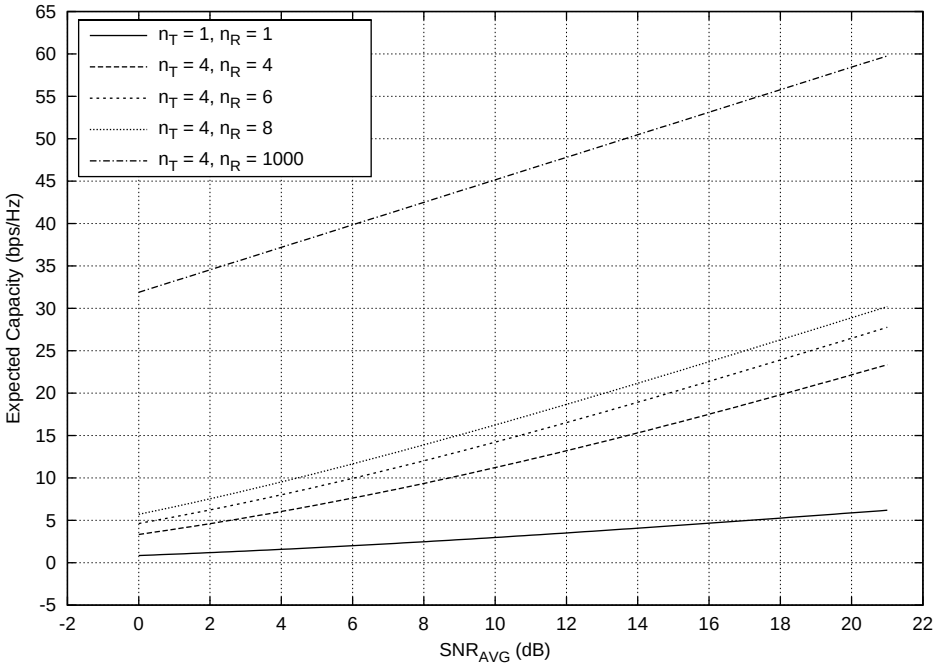

Figure 2.4: Expected capacity as a function of γ for $n_T = 4$ and $n_R = 4, 6, 8$.

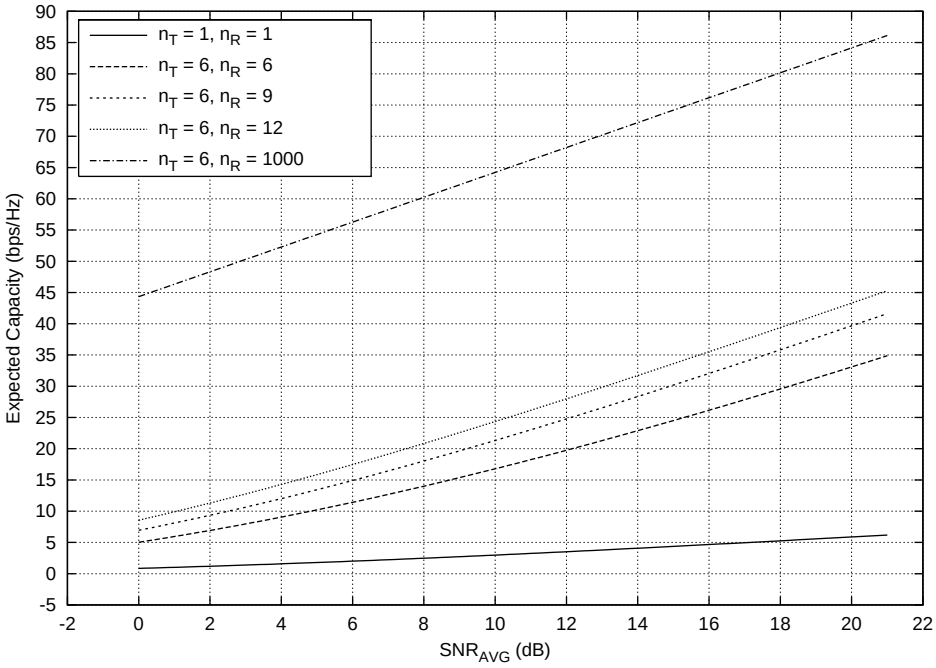

Figure 2.5: Expected capacity as a function of γ for $n_T = 6$ and $n_R = 6, 9, 12$.

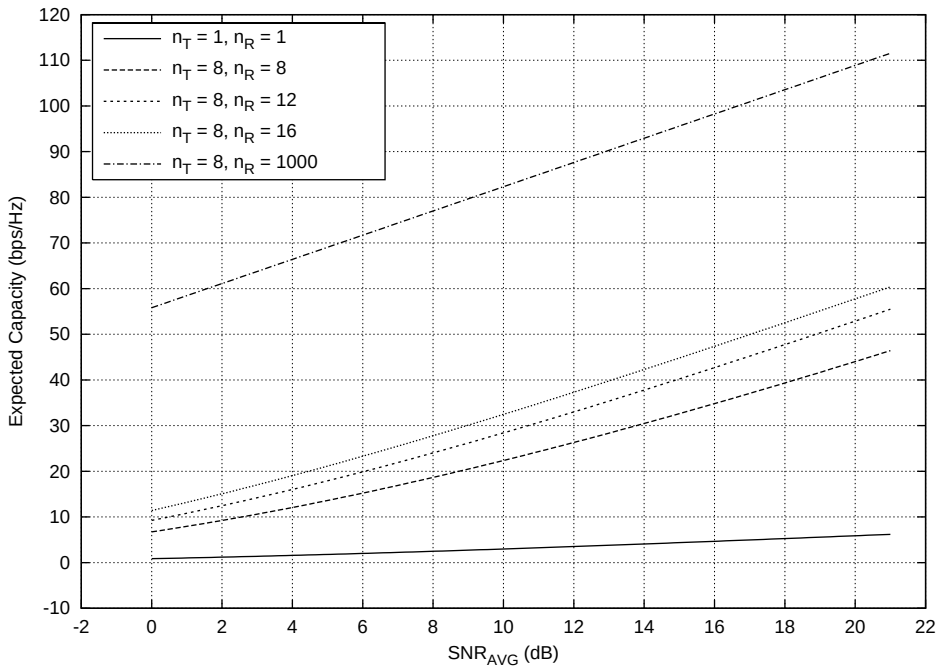

Figure 2.6: Expected capacity as a function of γ for $n_T = 8$ and $n_R = 8, 12, 16$.

C_{th} is a threshold value; the channel will be above this threshold with probability $P_{out}(C_{th})$.

Again following the analysis made by Foschini, the probability of outage is determined by means of the *complementary cumulative distribution function*, or *ccdf* for short. The ccdf is the probability that the capacity at any time will be higher than the specified threshold. Figures 2.7 through 2.18 present the ccdf of the antenna combinations described above.

2.5.2 Important remarks on the expected capacity and outage probabilities

One interesting aspect of the expected capacity curves is the slope variation as n_R increases. Single-antenna systems present the expected 1 bps/Hz for each 3dB increase in power. In contrast, in MIMO systems, the expected capacity increases approximately n_T bps/Hz for each 3dB increase in power; this is equivalent to having n_T independent, separate channels.

For very large values of n_R the expected capacity grows linearly with average SNR. For small values of n_R and small average SNR, however, the growth is not quite linear; it only becomes so for large SNR.

Even for very low outage probabilities (around 1%), and for modest array sizes, MIMO systems offer enormous increases in capacity over single-antenna systems[2].

[2]Note that physical constraints (cost, space, or deficiencies in the channel model) may prevent

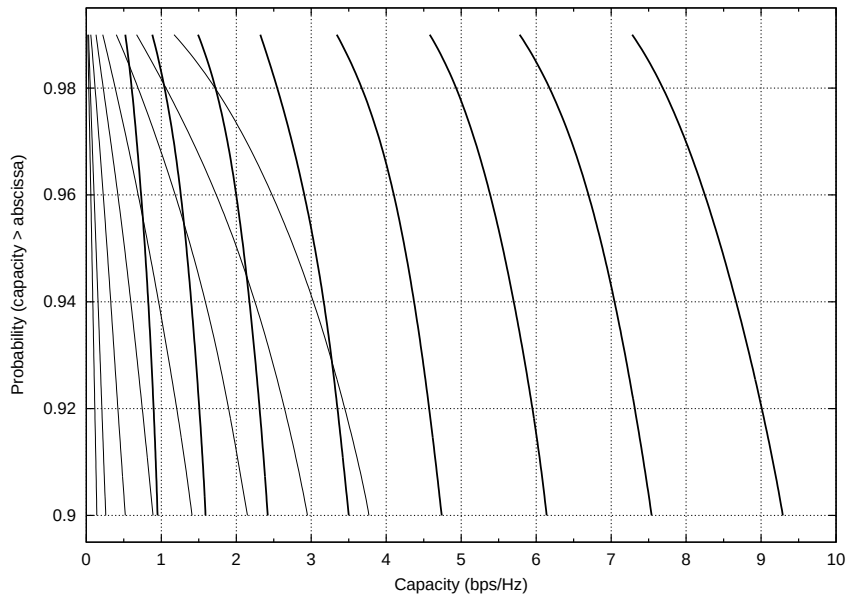

Figure 2.7: Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 2$ and $n_R = 2$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

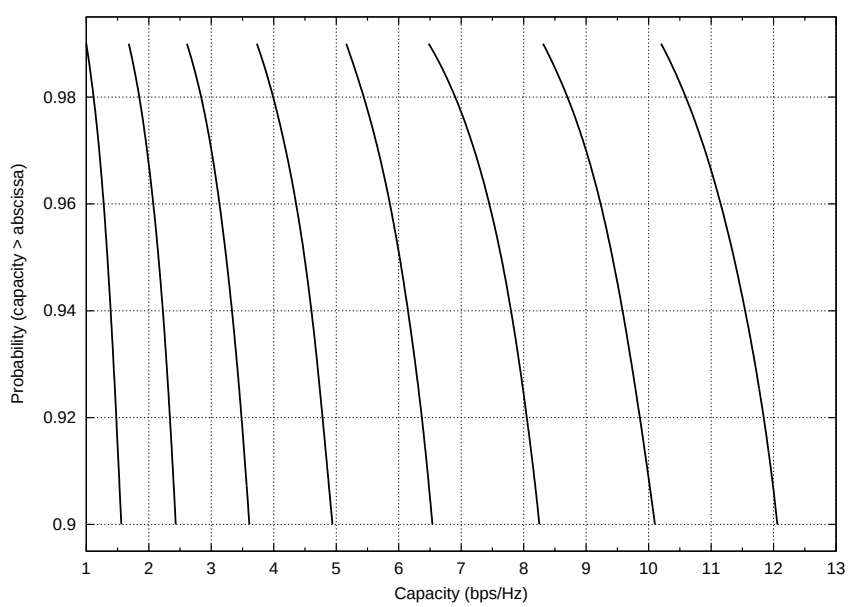

Figure 2.8: Complementary Cumulative Distribution Function of the capacity for $n_T = 2$ and $n_R = 3$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

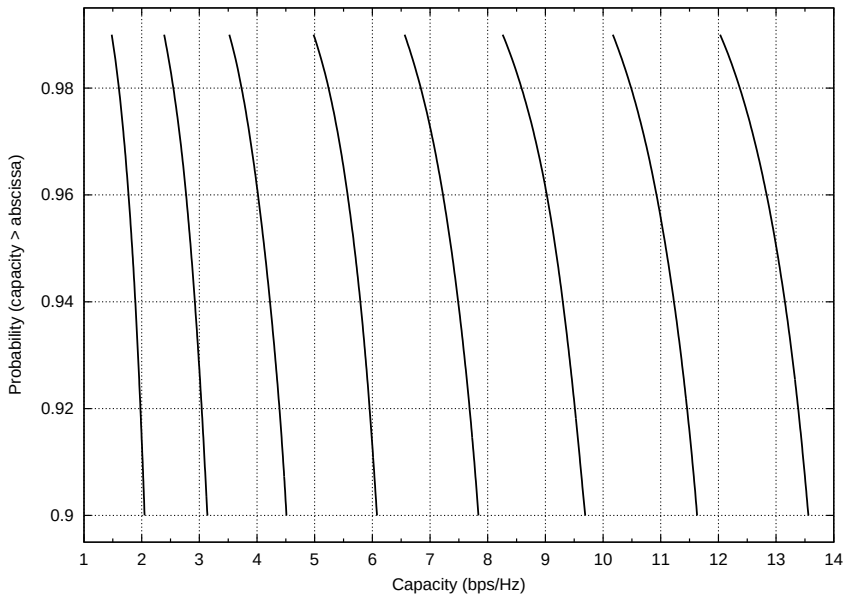

Figure 2.9: Complementary Cumulative Distribution Function of the capacity for $n_T = 2$ and $n_R = 4$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

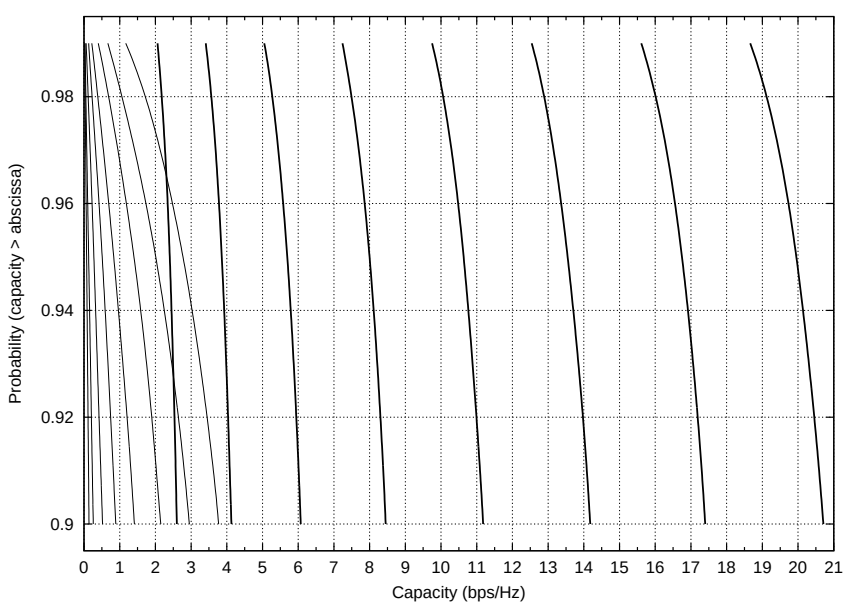

Figure 2.10: Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 4$ and $n_R = 4$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

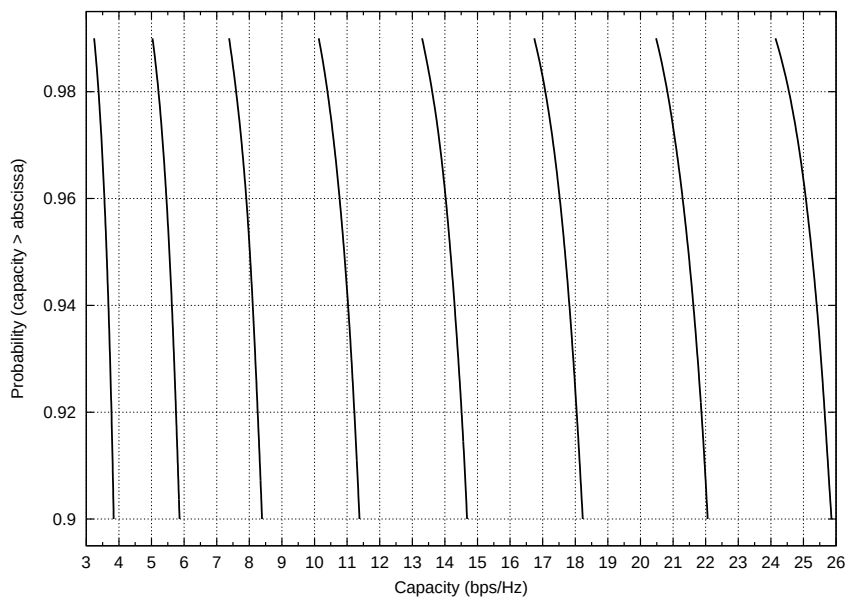

Figure 2.11: Complementary Cumulative Distribution Function of the capacity for $n_T = 4$ and $n_R = 6$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

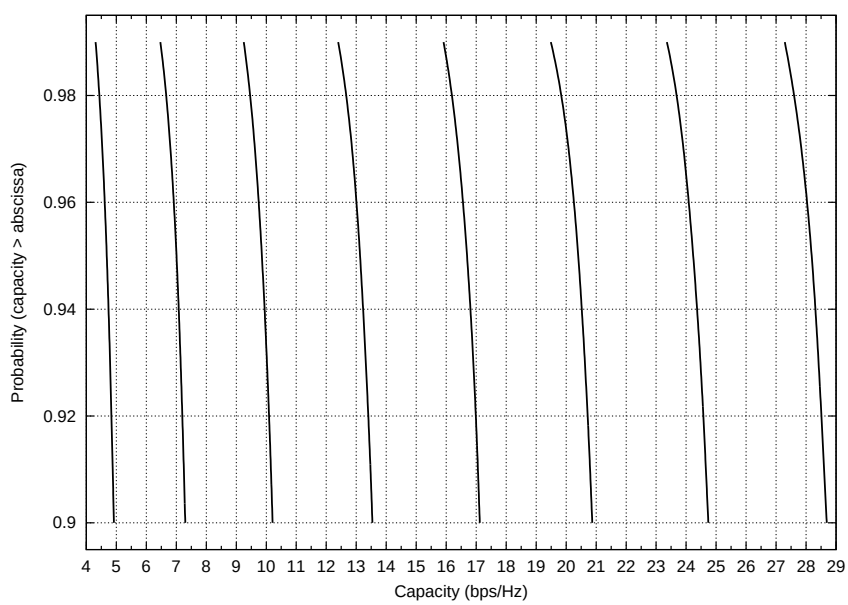

Figure 2.12: Complementary Cumulative Distribution Function of the capacity for $n_T = 4$ and $n_R = 8$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

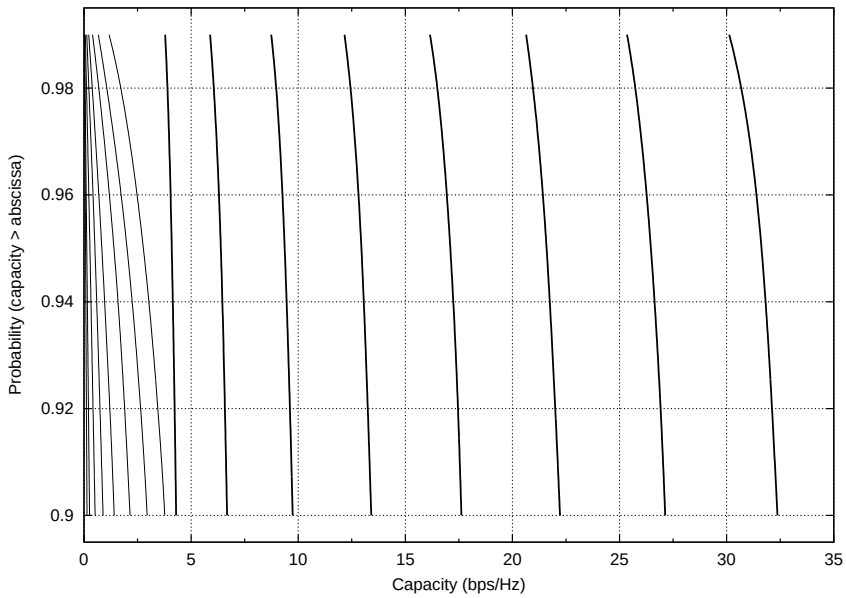

Figure 2.13: Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 6$ and $n_R = 6$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

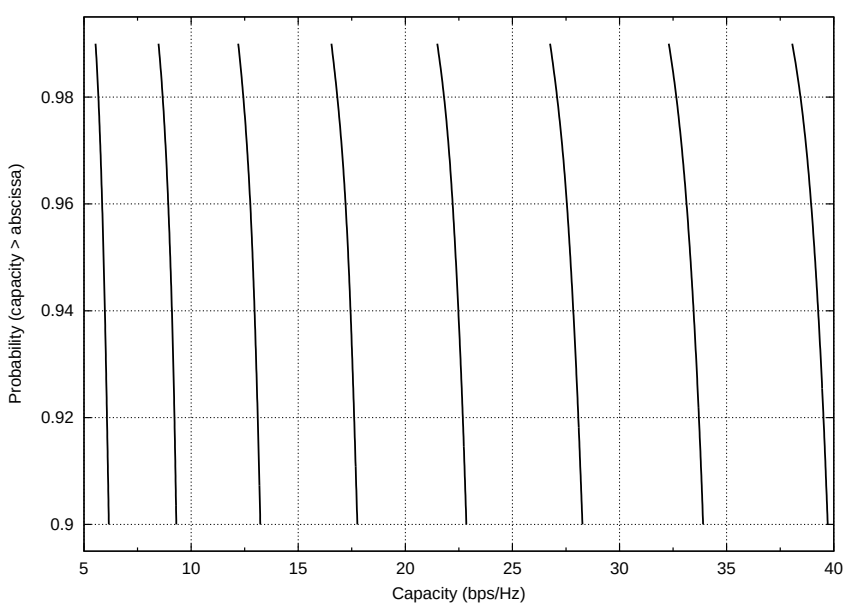

Figure 2.14: Complementary Cumulative Distribution Function of the capacity for $n_T = 6$ and $n_R = 9$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

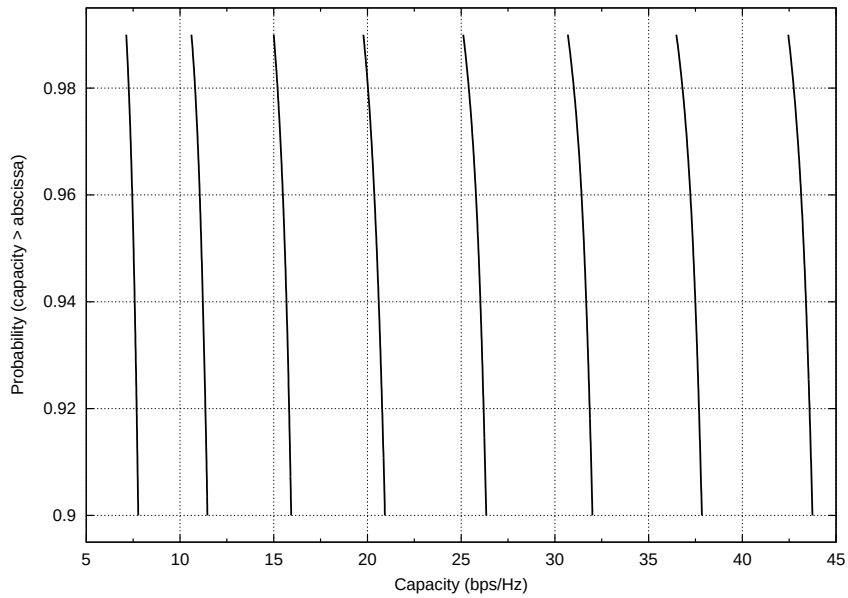

Figure 2.15: Complementary Cumulative Distribution Function of the capacity for $n_T = 6$ and $n_R = 12$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

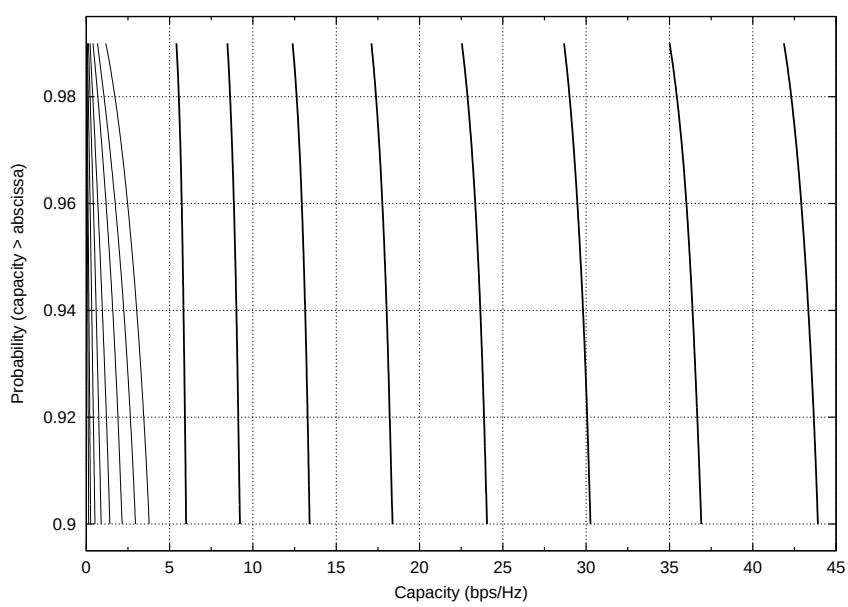

Figure 2.16: Complementary Cumulative Distribution Function of the capacity for $n_T = 1$ and $n_R = 1$ (thin lines) and $n_T = 8$ and $n_R = 8$ (bold lines). The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

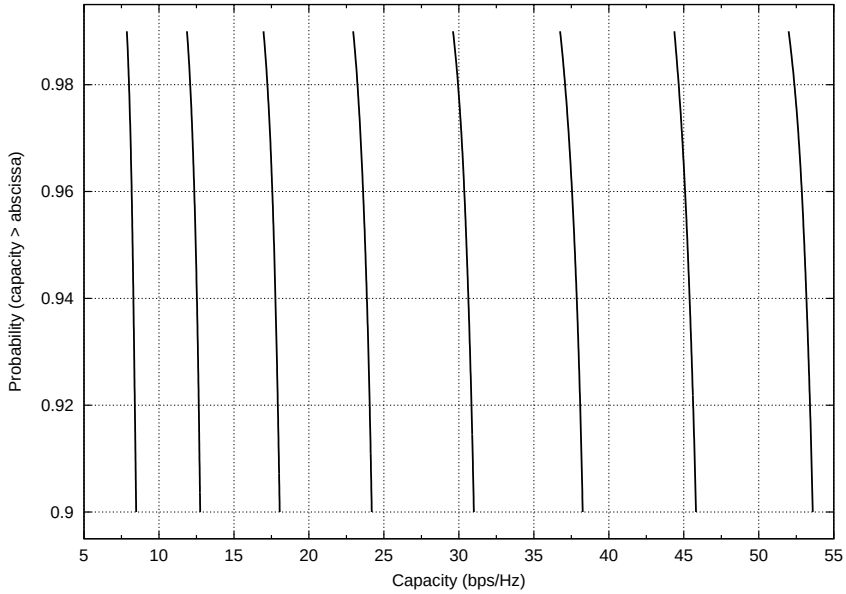

Figure 2.17: Complementary Cumulative Distribution Function of the capacity for $n_T = 8$ and $n_R = 12$. The SNR varies between 0 and 21 dB in increments of 3 dB. Each line represents a different SNR.

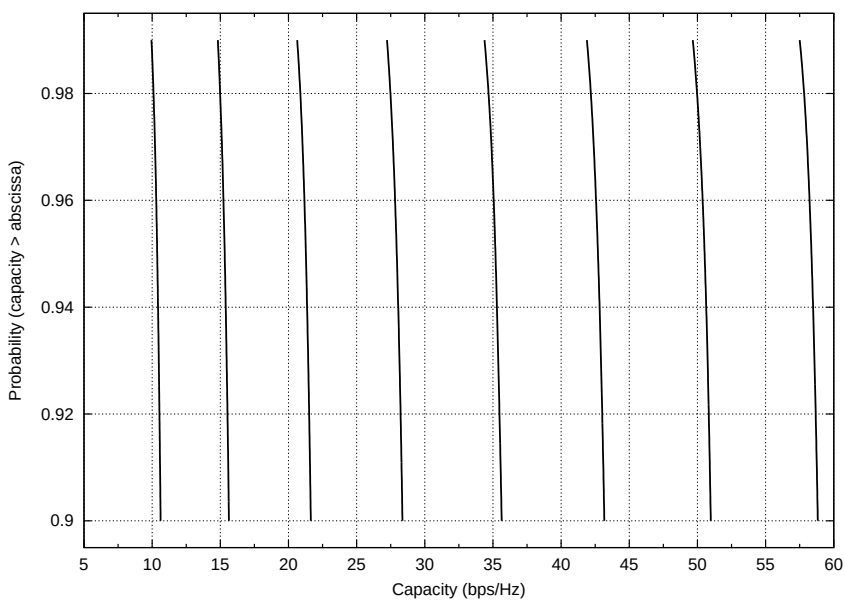

Figure 2.18: Complementary Cumulative Distribution Function of the capacity for $n_T = 8$ and $n_R = 16$. The SNR varies between 0 and 21 dB in steps of 3 dB. Each line represents a different SNR.

For example, at $P_{out} = 1\%$ and average SNR equal to 6dB, a single antenna system has a capacity of .06 bps/Hz, whereas for a (2, 2) system the capacity increases to 1.49 bps/Hz. For larger arrays the capacity becomes bigger and bigger, reaching almost 21 bps/Hz for a (8, 16) system operating at the same average SNR of 6dB and at $P_{out} = 1\%$.

It is also interesting to observe the slope of the ccdfs as n_R increases; the capacity tends to be the same for all values of P_{out}. For example, for average SNR equal to 9dB, and if $n_T = n_R = 2$, the difference between C_{th} for $P_{out} = 0.90$ and $P_{out} = 0.99$ is $3.5 - 2.3 = 2.2$ bps/Hz; the same difference for $n_T = 8$ and $n_R = 16$ is $28.4 - 27.2 = 1.2$ bps/Hz.

2.6 A Note on Diversity

Diversity is one of the most common and effective means available to exploit fading; it means to use multiple, independently fading channels to make replicas of the information signal available to the receiver. Each independently fading channel is called a *branch*. Thus, the probability that all replicas are faded simultaneously is reduced and can be made very small if many independent channels are used: if the probability of a signal fade over any given channel is p, then if D channels are available, the probability of all of them fading simultaneously is p^D.

There are several ways to create diversity. Each is appropriate for different circumstances and has distinct drawbacks. The most common methods are:

- In *frequency diversity* the information is repeated on D different carriers. The frequency separation between the channels needs to be large enough to guarantee independent fading (that is, larger than the *coherence bandwidth* of the channel). Its main disadvantage is that a large bandwidth is needed.

- In *time diversity* the information is repeated at D different times. The time period between repetitions needs to be larger than the channel's *coherence time* to warrant independent fading. It has the disadvantage of lowering the information rate.

- Another method is to use a wideband signal (defined as a signal with bandwidth much greater than the channel's coherence bandwidth). In this case, it is possible for the receiver to resolve each multipath component and thus benefit from *path diversity*. It has the disadvantage of requiring a very large bandwidth.

- The method of *space diversity* consists of employing multiple antennas to provide independent branches. The antennas need to have sufficient separation to warrant independence; this normally means at least some small multiple of

this theoretical capacity to be reached. In this thesis, these aspects are in general not considered.

the wavelength of the signal. Contrary to the other methods described, space diversity entails no loss of information rate and needs no extra bandwidth. Its disadvantage is mainly the extra cost and extra space required.

The result of providing diversity is an increase in the effective SNR at the receiver. When diversity is exploited optimally, using *maximal ratio combining* [7, 8], the effective SNR is the sum of the SNR of the individual branches:

$$\gamma_D = D \cdot \gamma. \tag{2.13}$$

A diversity technique that increases the effective SNR at the receiver to $D \cdot \gamma$ is said to provide a *diversity order* equal to D. Note that the number of physical branches available and the actual diversity order D are not always equal. Especially adverse circumstances can lower D to a fraction of the number of branches; on the other hand, efficient coding techniques can increase D above the number of branches.

In this thesis, the focus is on systems that use space diversity, but where multiple transmit antennas share the available receive antennas. An interesting question is what order of diversity such a system can provide.

An answer, that depends on specific coding techniques, is given in a further section; however, it is possible to obtain a general answer from the capacity equations. Recall from equation (2.8) that, in a (n, n) system with n independent branches, the capacity is given by $C_n = n \cdot \log(1 + (\gamma/n))$. When n_R is very large, the capacity has been given in (2.11) as

$$\lim_{n_R \to \infty} C = n_T \log\left(1 + \frac{1}{n_T} \cdot (\gamma \cdot n_R)\right).$$

This suggests that n_T branches have been obtained; in addition, the average SNR has been multiplied by n_R. The implication is that the order of diversity potentially available in an (n_T, n_R) MIMO system is $D = n_T \cdot n_R$.

2.7 Space-Time Codes

Space-time codes aim to take advantage of the enormous potential of MIMO systems by combining space and time diversity. There are three types of space-time codes: trellis, block, and layered. One specific type of the layered variation, vertical space-time codes, is the focus of this thesis; for this reason, the other codes are not described in great depth here.

Historically, layered codes were the first to be studied in the context of MIMO systems. Today, they remain the most promising technique to increase the capacity of wireless systems. They offer a very high spectral efficiency, ease of code design, and comparatively simple receivers, while still having error rates that are perfectly adequate for many applications.

A space time code maps a group of b information bits to a code vector sequence $\mathcal{A} = \mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_L$. $\mathcal{A}$ is a space-time codeword, and it is also represented as the sequence of numbers

$$\mathcal{A} = a_1^1 a_1^2 \cdots a_1^{n_T} a_2^1 a_2^2 \cdot a_2^{n_T} \cdot a_L^1 a_L^2 \cdot a_L^{n_T},$$

where a_j^i is the signal transmitted from antenna i at time j. Codeword $\mathcal{A}$ spans a whole block, and thus it is subject to constant fading. Under these conditions, it has been determined in [9] that the pairwise probability of a maximum-likelihood receiver deciding erroneously for a valid code word $\mathcal{E}$ when in fact codeword $\mathcal{C}$ was transmitted is bounded by

$$P(\mathcal{C} \to \mathcal{E}) \leq \left(\prod_{i=1}^{r} \lambda_i \right)^{-n_R} (1/2N_0)^{-rn_R}, \tag{2.14}$$

where r is the rank of a matrix $\mathbf{A}$ that depends on the codewords $\mathcal{C}$ and $\mathcal{E}$, given by $a_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$ for $\mathbf{x}_k = (c_1^k - e_1^k, c_2^k - e_2^k, \cdots, c_L^k - e_L^k)$; and λ_i are the non-zero eigenvalues of $\mathbf{A}$.

The exponent of the SNR in equation (2.14) is called the *diversity advantage* of the code; its maximum value is $n_T \cdot n_R$. On the other hand, $(\lambda_1 \lambda_2 \cdots \lambda_r)^{1/r}$ is called the *coding advantage*, which gives an approximation of the gain obtained by a coded system compared with an uncoded system operating at the same diversity advantage.

The code design problem then consists of choosing codewords that maximize the diversity and coding advantages. The *rank criterion* establishes that in order to achieve the maximum possible diversity, matrix $\mathbf{A}$ needs to be full rank for any two codewords $\mathcal{C}$ and $\mathcal{E}$.

In order to maximize the code advantage, the *determinant criterion* establishes that, if r is the rank of $\mathbf{A}$, then the coding advantage is given by the minimum of the r-th roots of the sum of the determinants of all the $r \times r$ cofactors of matrix $\mathbf{A}$ over all pairs of distinct codewords $\mathcal{C}$ and $\mathcal{E}$.

It should be noted that, even in the worst-case scenario where $r = 1$, there is a diversity advantage equal to n_R and there is a potential coding advantage to be gained.

Space-time codes present a tradeoff between constellation size, diversity, and transmission rate. In the case of maximum diversity advantage (equal to $n_T \cdot n_R$), if the constellation S used consists of 2^M signals, then the maximum transmission rate attainable is M bits per second per Hertz [9].

The design of good space-time codes must take this tradeoff into consideration, as well as the receiver complexity. Today, this area is in a prospective phase. Several very efficient codes exist but in general they remain very difficult to decode.

2.7.1 Space-Time Trellis Codes

In a *space-time trellis code* the codeword $\mathcal{A}$ is generated by a trellis coder, which is similar to its traditional counterparts, except that it must be reset at the beginning of each fading block. The labels of the transition branches are sequences of numbers $q^1 q^2 \cdots q^{n_T}$, which means that antenna i transmits the symbol corresponding to q^i, and all antennas transmit simultaneously.

Decoding of these codes can be performed optimally by a Viterbi decoder.

A technique for the construction of optimal trellis codes is given in [9]. Finding good trellis codes for moderate numbers of antennas has proven difficult; also, the large complexity imposed on the receiver by these codes is their main disadvantage.

2.7.2 Space-Time Block Codes

A second category of space-time codes is *space-time block codes* (*STBC*), introduced in [10], and which can be considered a generalization on ideas first proposed in [11]. Although less powerful than trellis codes [9], these codes have the advantage of being very simple to decode.

Assuming a signal constellation of size 2^b signals, an STBC maps $n_T b$ bits to code vector $\mathbf{a} = (a_1, a_2, \cdots, a_{n_T})$. The encoder then creates a $p \times n_T$ codeword matrix $\mathbf{A}$ whose elements are linear combinations of the variables $a_1, a_2, \cdots, a_{n_T}$ and their conjugates. The signal constellation set S in this case is defined as the set of signals that can appear as elements of $\mathbf{A}$. The rate of the code is defined as $R = n_T/p$ [12].

Matrix $\mathbf{A}$ is transmitted one row at a time; element a_i^j is transmitted by antenna j at time i. This matrix is constructed in such a way that it is orthogonal[3], which greatly simplifies the decoding process. A depiction of an STBC transmitter is shown in figure A.1. An example of an STB code is presented in appendix A.

2.7.3 Layered Space-Time Codes

A third variant of space time codes, called *layered space-time codes* (*LSTC*), was first introduced in [13] and [14]. The basic idea is to try to convert an (n_T, n_R) system into n_T (1, n_R) systems where one transmit antenna is processed at a time while the rest are simply regarded as interference. The objective is to apply the huge body of knowledge that has been developed for (1, n_R) systems, while still achieving a significant fraction of the available capacity. This method is also commonly known as *Bell Labs layered space-time*, or *BLAST*.

The encoding process is illustrated in figure 2.19 and works as follows: the information source is demultiplexed into n_T data streams. From each stream, b bits at a time are taken and modulated, producing vector $\mathbf{a} = (a_1, a_2, \cdots, a_{n_T})^T$. The layered space-time coder then layers a_1 in space and time, by transmitting it

[3]Here a $n \times m$ matrix $\mathbf{A}$ is defined as *orthogonal* if $\mathbf{A}\mathbf{A}^T = c\mathbf{I}$ when $\mathbf{A}$ is real and if $\mathbf{A}\mathbf{A}^\dagger = c\mathbf{I}$ when $\mathbf{A}$ is complex; c is a constant. See [10, Section IV.A].

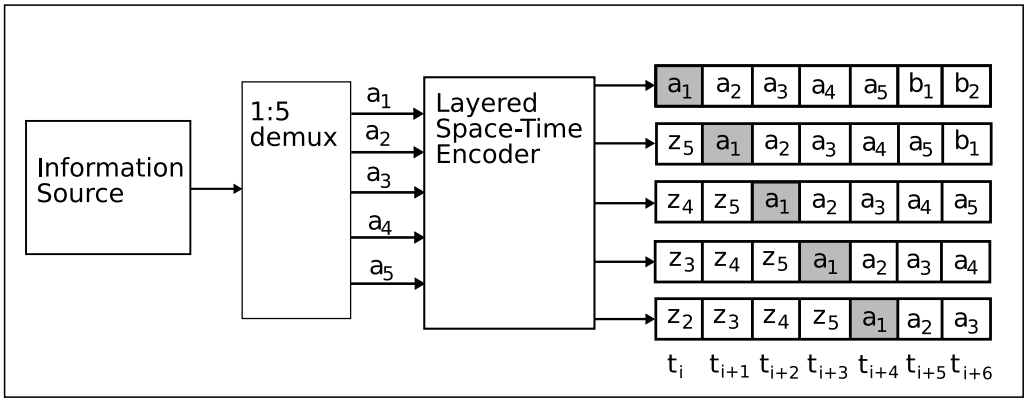

Figure 2.19: Block diagram of a layered space-time coder with five transmit antennas. Seven time intervals are shown. The information stream is demultiplexed into five data streams; here, a_i, $1 \leq i \leq n_T$, represent five constellation points. A layered space-time encoder layers each point in space and time; the encoding of point a_1 is highlighted. Symbols z_i represent data from the previous layer, while b_i represent data belonging to the next layer.

from antenna 1 at time t_1, from antenna 2 at time t_2, and so on, until it is finally transmitted from antenna n_T at time t_{n_T}. As for a_2, it is transmitted from antenna 1 at time t_2, from antenna 2 at time t_3, *etc.* This process is repeated for each a_i. Symbols from previous bit groupings are used to fill all available time slots. In all, each symbol is transmitted n_T times, once per antenna, and once per time interval. It is assumed that all symbol transmissions are made within a single block; as a consequence this scheme loses some efficiency at the beginning and end of each block. This process has been illustrated for a system with n_T equal to five in figure 2.19.

The decoding process for symbol a_1 proceeds largely as follows. Assume the receiver is dedicated to estimating a_1, and will use the vectors received at times t_1 through t_{n_T} for this task; let these vectors be denoted by $\mathbf{r}_k$, $1 \leq k \leq n_R$. It is also assumed that the symbols that are layered below a_1 (denoted by z_j in figure 2.19) have already been detected without error, and that those symbols layered above a_1 have not been detected yet.

Upon reception of each vector $\mathbf{r}_k$, the receiver cancels (or subtracts) the interference caused by those symbols already known, and nulls that caused by the symbols that are not yet known. To do this, it uses its knowledge of the channel matrix $\mathbf{H}$. For ease of exposition and without loss of generality, let us consider the system depicted in figure 2.19 at time t_3. Vector $\mathbf{r}_3$ can be written as follows:

$$\mathbf{r}_3 = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n_R} \end{bmatrix} = \mathbf{H} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ z_5 \\ z_4 \end{bmatrix} + \mathbf{n}.$$

The subindices of vector elements r indicate the receiver antenna. The value a_1 is in position 3 of $\mathbf{a}$. Since all symbols z_j are already known, their contribution to $\mathbf{r}_3$ can be eliminated (they can be canceled). Recall that the principle of BLAST is to regard everything except the particular symbol being estimated as interference. Let us consider the signal received by the first antenna:

$$r_1 = h_{11}a_3 + h_{12}a_2 + h_{13}a_1 + h_{14}z_5 + h_{15}z_4.$$

The quantity $h_{14}z_5 + h_{15}z_4$ is known and can be subtracted from r_1. Let $u_1 = h_{11}a_3 + h_{12}a_2 + h_{13}a_1$ and $\mathbf{H}_3 = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3]$ where $\mathbf{h}_i$ is column i of $\mathbf{H}$. Then, repeating this process for all r_k, vector $\mathbf{u}$ can be defined as

$$\mathbf{u} = \mathbf{H}_3 \begin{bmatrix} a_3 \\ a_2 \\ a_1 \end{bmatrix}.$$

The quantities a_3 and a_2 are not known but they can be nulled. Let $\mathbf{w}$ be a vector such that

$$\mathbf{w}\mathbf{H}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

and $\mathbf{w}\mathbf{u} = \hat{a_1}$ ($\mathbf{w}$ exists and is the third row of the Moore-Penrose pseudo-inverse of $\mathbf{H}_3$ [15, p. 257-258]). $\hat{a_1}$ is an estimate of a_1 produced according to the *zero-forcing* criterion [7]; other criteria (that is, other definitions of $\mathbf{w}$) exist, such as the *minimum mean-squared error* [8]. Repeating this process for each r_k produces n_R estimates of a_1; a process such as maximal ratio combining can then be used to produce a final estimate of a_1 with, in the ideal case, $D = n_R$ branches of diversity. In the same manner, this process can be applied to each layer to recover the whole data stream.

Trellis and block space-time codes have better error rates than layered codes. BLAST has the advantage of simplicity; it also allows for far greater spectral efficiencies. BLAST receivers are less complex than their counterparts and are easier to design. Their error-rate performance is adequate for many applications.

It is important to remark that receiver complexity remains one of the most important obstacles to the adoption of space-time techniques. Given its advantages in this area, together with its other qualities, further study of BLAST is justified.

It is a variant of BLAST that is the subject of this thesis. This variant, called *vertical BLAST* (or *V-BLAST* for short), repeats the symbols a_i only once. It will be studied in greater detail in the next chapter.

Chapter 3

Operating Principles of Vertical BLAST

3.1 Introduction

The *Vertical Bell-Labs Layered Space-Time* (*V-BLAST*) architecture is a space-time coder and receiver algorithm for MIMO systems. It was first introduced by [14, 16, 17]. In this chapter its operating principles are studied.

V-BLAST is interesting mainly because of its simplicity of operation, both in the transmitter and the receiver. This simplicity translates as a very high spectral efficiency. Its main disadvantage is a higher bit-error rate (*BER*) than the other types of codes introduced in Chapter 2.

Even though the formulation of V-BLAST and the resulting architecture are comparatively very simple, finding closed-form expressions for performance measurements such as capacity, outage probability, and probability of a bit error has proved to be extremely difficult. Regarding the latter, for instance, it is only very recently that expressions for some narrow cases have been reported in [18]. A glance at the references list in [18] suggests that there is scant prior work on the subject. Although some important results will be presented here, the emphasis of this thesis is placed on other aspects of the problem, namely: studying the complexity and performance of several different V-BLAST receivers, and finding new, improved algorithms.

Better algorithms are needed because, although simple to formulate, the V-BLAST receivers known to date have an enormous computational complexity, defined as the number of operations (algebraic, logical, and storage-related) that the receiver must perform per information bit received. In order to make V-BLAST feasible from a practical and economical standpoint, more efficient algorithms must be found.

Several implementations of V-BLAST have been presented in the literature since the original idea was published; some examples are those given in [19, 20, 21]. These usually trade *BER* for complexity, or vice-versa. Since V-BLAST's *BER* is subop-

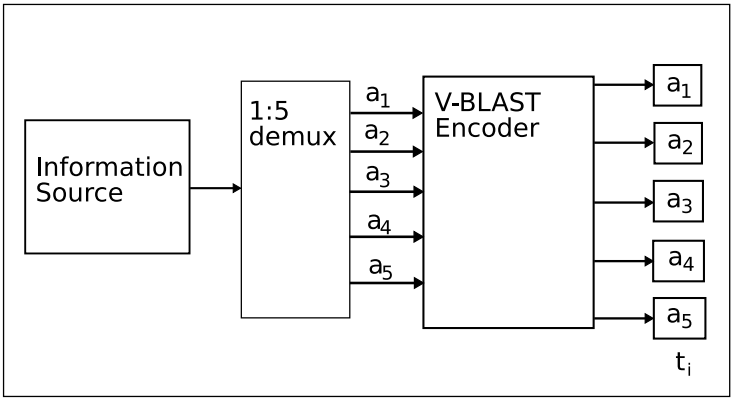

Figure 3.1: A V-BLAST transmitter in the case $n_T = 5$.

timal, and it is too complex to allow a feasible hardware implementation for a relatively small number of antennas, it seems useful to concentrate on finding algorithms that have nearly the same or better *BER* than V-BLAST, while decreasing its complexity.

3.2 V-BLAST Coding

The coding scheme employed in V-BLAST is a particular case of layered space-time coding, with the difference that each signal symbol is transmitted only once. Thus, the temporal aspect of the code is lost and only spatial diversity remains. A diagram of a V-BLAST transmitter with five transmit antennas is shown in figure 3.1.

The information signal is demultiplexed into n_T streams. It is assumed that these streams are uncoded. Each stream is modulated and transmitted; all assumptions presented in chapter 2 regarding the channel model are unchanged.

3.3 V-BLAST detection: ordering, canceling, nulling

The general idea of V-BLAST detection is the same as that of all LSTCs: process the received vector $\mathbf{r}$ to estimate the transmitted vector $\mathbf{a}$ by estimating each component a_i, one at a time, canceling the effect of those symbols already detected, and nulling those yet unknown.

As defined in chapter 2, the received vector $\mathbf{r}$ is defined as

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n},$$

where the elements of $\mathbf{H} \in \mathbb{C}^{n_R \times n_T}$ are complex Gaussian independent random variables with zero mean and variance 0.5 per dimension; $\mathbf{a} \in \mathbb{C}^{n_T \times 1}$ is the transmitted vector; and $\mathbf{n} \in \mathbb{C}^{n_R \times 1}$ is a noise vector of circularly-symmetric complex Gaussian

independent random variables with zero mean and variance N_0 per dimension. The receiver's task is to estimate $\mathbf{a}$ from its knowledge of $\mathbf{r}$ and $\mathbf{H}$.

Probably the simplest way to find an estimate for $\mathbf{a}$ is through the equation:

$$\hat{\mathbf{a}} = \mathbf{H}^{+}\mathbf{r},$$

where $\mathbf{H}^{+}$ is the *Moore-Penrose pseudo-inverse* (*MPPI*) of matrix $\mathbf{H}$. This method does not work very well. For instance, $\hat{\mathbf{a}}$ might not belong to the set of possible transmitted vectors. Removing the effect of the noise on $\hat{\mathbf{a}}$ is difficult, since it is colored by the multiplication with $\mathbf{H}$.

An improvement over this idea is to estimate each transmitted symbol in sequence. In each step of the sequence, the symbols already known are subtracted from $\mathbf{r}$ (or *canceled*), and those symbols that are unknown are considered interference and *nulled*. This idea is similar to decision-feedback equalization; this similarity is the subject of [22].

The question of the order of detection of the symbols remains. One of V-BLAST's key aspects is that different orderings will produce different error rates. Let $\mathcal{K}_i$ be a detection ordering, defined as an ordered set of the integers k, $1 \leq k \leq n_T$. Clearly, there are $n_T!$ possible orderings. One of them is optimal in the sense that it produces the minimum probability of error. Let the optimal ordering be $\mathcal{K}_o = \{k_1, k_2, \ldots, k_{n_T}\}$; $\mathcal{K}_o$ establishes the order in which the transmitted signals a_i will be detected. The method used to determine $\mathcal{K}_o$ will be explained below; for the moment, assuming that an ordering has been found, the detection of a_{k_i} proceeds as follows.

Symbol cancellation. If $i = 1$, no symbols are yet known so no cancellation is possible; therefore, $i > 1$ is assumed. Let $\mathbf{h}_1, \mathbf{h}_2, \ldots \mathbf{h}_{n_T}$ be the columns of channel matrix $\mathbf{H}$. It is assumed that the symbols $a_{k_1}, a_{k_2}, \ldots, a_{k_{i-1}}$ have already been estimated. Their contribution to $\mathbf{r}$ can then be canceled:

$$\mathbf{r}_{k_i} = \mathbf{r} - \hat{a}_{k_1}\mathbf{h}_{k_1} - \hat{a}_{k_2}\mathbf{h}_{k_2} - \cdots - \hat{a}_{k_{i-1}}\mathbf{h}_{k_{i-1}}. \tag{3.1}$$

Alternatively, let $\mathbf{H}_{k_i}$ be equal to $\mathbf{H}$ except that all elements of columns

$$k_1, k_2, \ldots, k_{i-1},$$

are made equal to zero. Then,

$$\mathbf{r}_{k_i} = \mathbf{H}_{k_i}\mathbf{a} + \mathbf{n}. \tag{3.2}$$

Interference nulling. The symbols

$$a_{k_{i+1}}, a_{k_{i+2}}, \ldots, a_{k_{n_T}},$$

have not been detected yet. However, from knowledge of $\mathbf{H}$, they can be nulled. Nulling means linearly weighting the received vector in a way that satisfies some criterion. The most common criteria are:

- *zero-forcing*, when the interference is simply forced to zero and the noise vector $\mathbf{n}$ is disregarded; and
- *minimum mean-squared error* (or *MMSE*), which aims at reducing the mean-squared error $\|\mathbf{a} - \hat{\mathbf{a}}\|^2$ taking into consideration the presence of noise.

The MMSE criterion has better bit-error performance than zero-forcing for low SNR, but it has one major drawback: it requires the receiver to know N_0. Also, for large SNR, MMSE and zero-forcing are equivalent. In this thesis, the focus will be on V-BLAST using the zero-forcing criterion.

Zero-forcing consists in finding $\mathbf{w}_{k_i} \in \mathbb{C}^{1\times n_R}$ such that $\mathbf{w}_{k_i}\mathbf{H} = [0, \ldots, 0, 1, 0, \ldots, 0]$, where the element equal to one is in the position k_i. Then, $\hat{a}_{k_i}$ is given by

$$\begin{aligned} \hat{a}_{k_i} &= \mathbf{w}_{k_i}\mathbf{r}_{k_i} + \mathbf{w}_{k_i}\mathbf{n} \\ &= \mathbf{w}_{k_i}\mathbf{H}\mathbf{a} + \mathbf{w}_{k_i}\mathbf{n} \end{aligned} \tag{3.3}$$

This process can be interpreted from a geometrical perspective as projecting $\mathbf{r}_{k_i}$ onto a vector that is orthogonal to the $n_T - i$ dimensional vector space spanned by the columns of $\mathbf{H}$ that correspond to those symbols that have not yet been estimated: $\mathbf{h}_{k_{i+1}}, \mathbf{h}_{k_{i+2}}, \ldots, \mathbf{h}_{k_{i+n_T}}$

Optimal ordering. The optimal ordering has been found to be determined by the SNR of each a_{k_i}: the symbol with the strongest SNR should be detected first, followed by the strongest symbol among the remaining ones, and so on until all symbols have been detected. In other words, for any i, a_{k_i} must have a larger SNR than a_{k_i}, a_{k_i+1}, ..., $a_{k_{n_T}}$.

The post-detection SNR of a_{k_i} can be determined directly from equation (3.3):

$$\gamma_{k_i} = \frac{|a_{k_i}|^2}{2N_0\|\mathbf{w}_{k_i}\|^2}, \tag{3.4}$$

where $\mathbf{w}_{k_i}$ is the vector used to null the unknown symbols $a_{k_{i+1}}, a_{k_{i+2}}, \ldots, a_{k_{n_T}}$. Choosing k_i requires finding the SNR of all remaining symbols, and choosing the largest one. In other words,

$$k_i = \underset{j\notin\{k_1,k_2,\ldots,k_{i-1}\}}{\operatorname{argmin}} \frac{|a_j|^2}{\|\mathbf{w}_j\|^2}. \tag{3.5}$$

In practice, the receiver does not know the value of a_{k_i} in the numerator of (3.5). Assuming all signals have equal energy, the expression for k_i can be rewritten as

$$k_i = \underset{j\notin\{k_1,k_2,\ldots,k_{i-1}\}}{\operatorname{argmin}} \frac{1}{\|\mathbf{w}_j\|^2}. \tag{3.6}$$

Clearly, if not all symbols in the constellation S have equal energy, equation (3.6) provides a sub-optimal ordering. This introduces a source of performance loss in V-BLAST that will be explored in more detail in the next chapter.

Note that, on average, the norm of $\mathbf{w}_{k_i}$ will tend to be smaller as the number of interferers decreases (that is, as i grows). This gives an intuitive basis for the optimal ordering. The first symbol to be estimated, a_{k_1}, has $n_T - 1$ interferers, which is why it benefits from having a strong SNR. The symbol estimated last, on the other hand, has a comparatively feeble SNR, but it does not suffer from any interference. A rigorous proof is given in [14].

3.3.1 V-BLAST spectral efficiency

If each transmit antenna emits one symbol per second per available hertz of bandwidth, then the spectral efficiency of V-BLAST is given by

$$\Phi = b \cdot n_T,$$

where b has been defined in chapter 2 as the number of information bits per symbol[1]. Then, the raw data rate attainable is $b \cdot n_T \cdot B$, where B is the available bandwidth.

For example, a MIMO system with $n_T = 8$, employing 16-QAM with an available bandwidth of 30kHz, has a spectral efficiency of 32 bps/Hz and a raw data rate of 960kbps. Equation (2.9) can be used to determine if such a rate is feasible, and how many receiver antennas should be used. For instance, if $n_R = 12$, then—as figure 2.17 shows—a good probability of outage should be attainable with this system for an SNR value of at least 12dB. It is clear that a system like V-BLAST will not reach capacity; however, it should be noted that such data rates are at least possible according to information theory.

A portion of the bandwidth must be dedicated to the training sequence needed by the receiver to estimate $\mathbf{H}$, lowering the effective data rate.

3.4 The V-BLAST reception algorithm

The algorithm V-BLAST has been defined in [17] as follows:
where $(\mathbf{G})_j$ is row j of $\mathbf{G}$, and $f_q(\cdot)$ is a function that quantizes the soft estimate y_i to the closest point in S.

This formulation of the algorithm is adequate for the expository aims of [17], but it does not facilitate its complexity analysis. In the first place, it processes each vector $\mathbf{r}$ one at a time, giving the impression that n_T pseudo-inverses are needed per vector. Second, it substitutes the columns of $\mathbf{H}$ for zeroes instead of simply removing them, which is more efficient.

To address these issues, and emphasize the organization of V-BLAST from the standpoint of its complexity, the algorithm is re-written as follows.
where $\hat{a}_{j,O_i}$ is element O_i of $\hat{\mathbf{a}}_j$. At each step one whole column of matrix $\mathbf{A}$ is removed; thus, the pseudo-inverses are calculated on matrices of diminishing size.

[1]The units of Φ are $\frac{\text{bits}}{\text{symbol}} \cdot \frac{\text{symbol}}{\text{s Hz}}$, or $\frac{\text{bit}}{\text{s Hz}}$ (represented by bps/Hz throughout this thesis).

Algorithm 1 Original V-BLAST

Input: an $n_R \times n_T$ matrix $\mathbf{H}$, an $n_R \times 1$ vector $\mathbf{r}$, and a signal constellation S.

Output: an $n_T \times 1$ vector $\hat{\mathbf{a}}$ whose elements are in S and $\mathbf{H}\hat{\mathbf{a}} = \mathbf{r} + \mathbf{v}$, where $\mathbf{v}$ is an error vector.

Let $i = 1$
$\mathbf{G}_1 = \mathbf{H}^+$
$k_1 = \text{argmin}_j \|(\mathbf{G}_1)_j\|^2$
for $i = 1$ to n_T **do**
    $\mathbf{w}_{k_i} = (\mathbf{G}_i)_{k_i}$
    $y_{k_i} = \mathbf{w}_{k_i}\mathbf{r}_i$
    $\hat{a}_{k_i} = f_q(y_{k_i})$
    $\mathbf{r}_{i+1} = \mathbf{r}_i - \hat{a}_{k_i}\mathbf{h}_{k_i}$
    $\mathbf{G}_{i+1} = \mathbf{H}_{k_i}^+$
    $k_{i+1} = \text{argmin}_{j \notin \{k_1, k_2, \ldots, k_i\}} \|(\mathbf{G}_{i+1})_j\|^2$
end for

Algorithm 2 V-BLAST

Input: an $n_R \times n_T$ matrix $\mathbf{H}$, a set of L $n_R \times 1$ vectors $\mathbf{r}_i$, $i = 1, \ldots, L$, and a signal constellation S.

Output: a set of L $n_T \times 1$ vectors $\hat{\mathbf{a}}_i$. Each vector $\hat{\mathbf{a}}_i$ is such that its elements are in S and $\mathbf{H}\hat{\mathbf{a}}_i = \mathbf{r}_i + \mathbf{v}_i$, where $\mathbf{v}_i$ is an error vector.

$\mathbf{A} = \mathbf{H}$
for $i = 1$ to n_T **do**
    Let $\mathbf{G}_i = \mathbf{A}^+$
    Let $k_i = \text{argmin}_j \|(\mathbf{G}_i)_j\|^2$
    Let O_i equal to corresponding column in $\mathbf{H}$
    Remove column k_i of $\mathbf{A}$
end for
for $j = 1$ to L **do**
    for $i = 1$ to n_T **do**
        Let $\mathbf{w}$ equal to row k_i of $\mathbf{G}_i$
        Let $y = \mathbf{w}\mathbf{r}_j$
        Let $\hat{a}_{j,O_i} = f_q(y)$
        Let $\mathbf{z}$ equal to column O_i of $\mathbf{H}$
        Let $\mathbf{r}_j = \mathbf{r}_j - \hat{a}_{j,o_i}\mathbf{z}$
    end for
end for

γ (dB)	-5	0	5	10	15	20
Max. $\mathbf{C_v}$ (bps/Hz/dim)	0.24	0.53	1.03	1.76	2.71	3.83

Table 3.1: Maximum value of V-BLAST capacity for several values of average SNR.

Step 5 is needed to preserve the symbol ordering relative to the original channel matrix $\mathbf{H}$ and not to matrix $\mathbf{A}$.

When V-BLAST is presented as above, it becomes apparent that its complexity can be divided in two parts. One part, which is called the *block setup phase*, is computationally expensive, but is performed only once per block. Its complexity depends on n_T and n_R. The other part, called the *symbol estimation phase*, is computationally simpler but is performed many times. Its complexity depends not only on n_T and n_R but also on L. It also becomes evident that only n_T pseudo-inverses are needed per block.

3.5 Capacity, outage probability and diversity of V-BLAST

General expressions for the capacity, probability of outage, and probability of error of V-BLAST have proved remarkably hard to find. However, there are certain results that provide justification for the use and adoption of this algorithm. Some of the most relevant ones follow.

Assume that n_T is very large and $\alpha = n_T/n_R$. Then, the capacity of V-BLAST measured in bps/Hz/dimension[2] is approximately given by [14]

$$C_v \approx \alpha \cdot \log(1 + \gamma \cdot (\alpha^{-1} - 1). \tag{3.7}$$

There is one value of α that maximizes the capacity for each γ; an (n_T, n_R) system that maximizes capacity is said to be *optimized*. This capacity is lower than the general MIMO capacity but is still attractive. Figure 3.2 illustrates the shape of the capacity curve as a function of α, for average SNR $\gamma = 10$dB. Table 3.1 shows the maximum capacity for several values of γ.

The problem of outage probability P_o is considered in [23] (see also [24, 25]). P_o is given in terms of the rate R (in bps) and average SNR γ at which the system is operating. Recall that V-BLAST aims to separate the MIMO channel into independent subchannels. Each subchannel is subject not only to the noise present in the MIMO channel but to interference from the subchannels that haven't been estimated yet. Outage is defined here as the event that the data rate is greater than the capacity of any subchannel, taking said interference into account, for a given channel realization. Note that this is a very narrow definition since data transmis-

[2]Here, *dimension* refers to the dimension of the transmitted signal, which is equal to n_T.

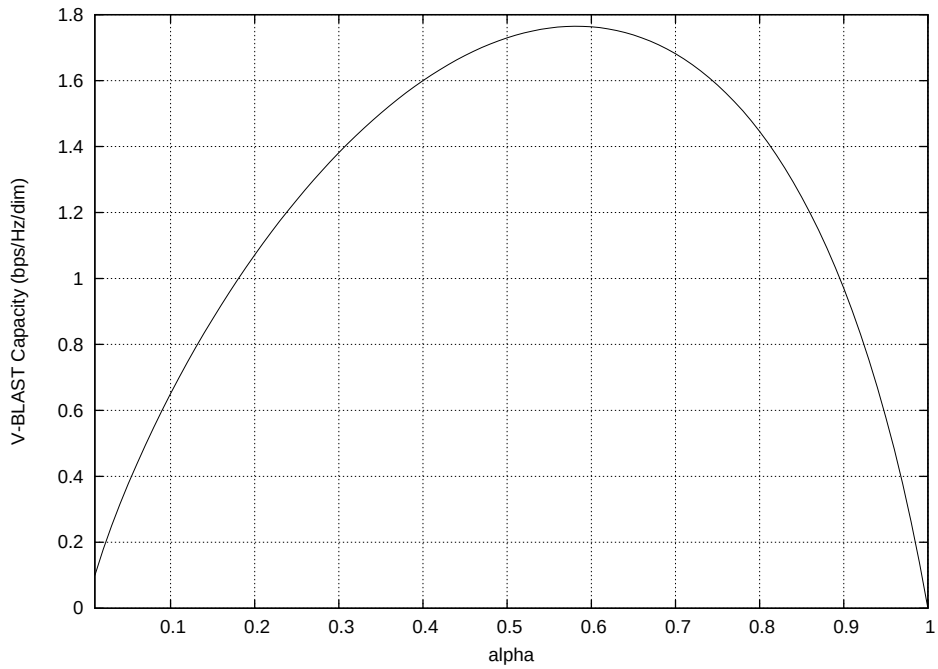

Figure 3.2: Capacity of V-BLAST as a function of $\alpha = n_T/n_R$, in units of bps/Hz/dimension, for $\gamma = 10$dB.

γ	0	5	10	15	20	25
Rate	0.6	1.6	4	8	14	20

Table 3.2: Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = n_R = 4$.

sion is still possible over the subchannels that do not surpass their capacity (P_o can be associated with *BLER*, the probability of a block error).

Let $r = R/n_T$ denote the rate of each subchannel. The probability of subchannel i having a rate larger than its capacity is given by

$$G(i, \gamma, r) = e^{-(2^r-1)/\gamma} \sum_{k=0}^{n_R-n_T+i-1} \frac{1}{k!} \left(\frac{2^r - 1}{\gamma} \right)^k,$$

and the probability of outage is then

$$P_o = 1 - \prod_{i=1}^{n_T} G(i, \gamma, r). \tag{3.8}$$

Note that (3.8) does not take into account the benefits obtained from ordering the detection of the subchannels; in this sense, it can be considered a worst-case outage probability.

Tables 3.2 and 3.3 present some values of R as a function of γ for (8,8) and (4,4) systems, respectively, with $P_o = 0.1$. Observe that the rate per subchannel r is the

γ	0	5	10	15	20	25
Rate	1.2	3.2	8	16	28	40

Table 3.3: Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = n_R = 8$.

γ	0	5	10	15	20	25
Rate	10	16	22.5	29	36	42.5

Table 3.4: Maximum rate of V-BLAST algorithm for $P_o = 0.1$ and $n_T = 4$, $n_R = 12$.

same for both systems, and, in fact, it is the same for all (n, n) systems. Table 3.4 presents the maximum rates for a $(4, 12)$ system. The increase in rate compared with the symmetrical systems for low SNR is remarkable.

In [18], the V-BLAST algorithm is analyzed from a geometrical point of view. Among other results, it is proved that the diversity order for the i-th subchannel is $(n_R - n_T + i)$. This result is true whether optimal ordering is used or not; the benefit of optimal ordering is to increase the effective average SNR of each subchannel (in other words, ordering has no effect on the asymptotic slope of the outage probability curve).

These results, taken together, indicate the potential of the V-BLAST algorithm and justify further analysis and study. Furthermore, Monte-Carlo simulations presented in the following chapter are encouraging. It should also be noted that a (16, 16) V-BLAST experimental prototype is reported to be operational at Bell Labs [14], with performance close to that predicted by the theory.

Chapter 4

Practical Considerations in V-BLAST Implementations

4.1 Introduction

IMPLEMENTING V-BLAST in an efficient manner, whether in hardware or software, is not an easy task. Some of the questions that must be answered are:

- What is the best method to find the pseudo-inverse of the channel matrix, regarding numerical stability as well as speed;
- What are the computational requirements demanded by V-BLAST;
- What data rates are achievable with modern processors;
- Where V-BLAST's bottlenecks are located;
- What are the memory requirements of V-BLAST;
- What is the *BLER* performance that can be expected from a V-BLAST receiver for different antenna configurations;
- What role, if any, does the selection of L play on V-BLAST's speed and error rates;
- What is the effect of non-optimal detection ordering discussed in section 2.3.

The specific answers to some of these questions might vary according to factors that are independent from the algorithm itself, such as the computer architecture used, or the ability of designers and automated tools to optimize its execution. The aim here is not to focus on such particular implementation details, but rather to draw conclusions on V-BLAST's inherent properties and requirements. It is believed that such an approach is an essential first approximation to the non-trivial problem of constructing a V-BLAST receiver within the computational and economical constraints of practical applications. General results may be used to facilitate the construction of any particular V-BLAST receiver.

Many of the questions above have a common theme: performance, data rate, bottlenecks. This theme is distilled into the term *algorithmic complexity*. In this thesis, complexity is defined simply as the number of operations an algorithm must execute to finish its task.

In order to obtain the most general results possible, some techniques that are common in the study of algorithms (for example, profiling) have been eschewed in favor of a simple, direct count of the quantity and type of operations the algorithm calls for. The reasons for this are explained in further detail in the following subsections.

4.1.1 The problem with profiling

When optimizing the execution of a computer program, a tool called a *profiler* is commonly used. A profiler watches over the program's execution and determines how much processor time was spent in each function, or even on each line of code. Such a tool does not provide an adequate degree of accuracy and granularity. For instance, it cannot distinguish between types of operations.

However, the main reason why profiling was not considered an option to study V-BLAST is that it is highly architecture-dependent. In other words, its results are tied to the processor where the program is running; thus, they are not useful for understanding the general behavior of V-BLAST.

4.1.2 The problem with the O(·) notation

In the field of algorithmic complexity, it is customary to use the so-called $\mathrm{O}(\cdot)$ notation (also known as the Big-O notation) to express complexity. Explained briefly, $\mathrm{O}(\cdot)$ is a compact, succinct way of expressing the asymptotic complexity of an algorithm when the size of the problem to be solved grows without limit.

For example, the time (or number of steps, or of instructions) needed by a certain algorithm A_1 to solve a problem of size n might be given by $4n^3 + 2n + 10$. As n tends to infinity, the term n^3 dominates; furthermore, all coefficients are disregarded and it is said that $A_1 \in \mathrm{O}(n^3)$. This captures an essential property of the algorithm: its complexity grows as the cube of its input.

When the problems to be solved are very large, the $\mathrm{O}(\cdot)$ notation can be useful in selecting a particular algorithm over another. For instance, if a matrix-inversion algorithm $A_1 \in \mathrm{O}(n^3)$ and another algorithm $A_2 \in \mathrm{O}(n^2)$, and the problem consists of inverting a matrix with 100,000 rows, it is a safe bet that A_2 is a better option (at least as far as complexity is concerned). However, for small problems, the $\mathrm{O}(\cdot)$ notation is not as useful. If A_1 takes $4n^3 + 2n + 10$ steps to solve the problem and A_2 requires $20n^2 + 25n$, then for $n = 5$, A_1 takes 145 steps against 625 for A_2. It is clear that A_1 is a better option for this particular problem than A_2, regardless of its cubic behavior.

The problems that a V-BLAST receiver has to solve are rather small. The largest problem studied here is that of finding the pseudo-inverse of a 16×8 matrix. Using the $\mathrm{O}(\cdot)$ notation to study the details of the V-BLAST algorithm would be useless at best and deceiving at worst. For this reason, the $\mathrm{O}(\cdot)$ notation is not used at all. Instead, the complexity of each variant of V-BLAST is studied in detail and the numbers obtained are based on accurate counts of every single operation performed, as explained in the next section.

4.1.3 Note on simulation details

All simulation in this thesis were done in double precision on 32-bit x86 Intel processors. The simulations were run until 2000 block errors were found. For low SNR the number of bit errors simulated is roughly 2000; the confidence interval will then be at least 95% when $BER = 10^{-3}$.

4.2 Computation of the Moore-Penrose pseudo-inverse

Simulation proves the common-sense observation that, by far, the most complex operation of V-BLAST is the Moore-Penrose pseudo-inverse (*MPPI*) of the channel matrix $\mathbf{H}$. There are several methods that can be used to find the MPPI of a matrix; four are studied here, and the least complex one is identified. Also, their numerical stability is evaluated. The four methods under consideration are detailed below. In all cases the channel matrix being inverted is $\mathbf{H} \in \mathbb{C}^{n_R \times n_T}$

i. Singular-value decomposition (SVD). The SVD of matrix $\mathbf{H}$ is defined as

$$\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^{\dagger},$$

where $\mathbf{U} \in \mathbb{C}^{n_R \times n_T}$ and $\mathbf{V} \in \mathbb{C}^{n_T \times n_T}$ are unitary matrices, and $\mathbf{D}$ is a diagonal matrix.

The MPPI of $\mathbf{H}$ is given by

$$\mathbf{H}^{+} = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^{\dagger}.$$

The SVD is interesting because it is numerically very stable when $\mathbf{H}$ is ill-conditioned[1]. Clearly, its main disadvantage is a large complexity.

ii. MPPI formula. When $(\mathbf{H}^{\dagger}\mathbf{H})^{-1}$ exists, its MPPI can be found with the following formula:

$$\mathbf{H}^{+} = (\mathbf{H}^{\dagger}\mathbf{H})^{-1}\mathbf{H}^{\dagger}. \tag{4.1}$$

[1]The *condition number* describes how close to singular a matrix is. Ill-conditioned matrices (those whose condition number is too large) can cause numerically unstable algorithms to produce erroneous results [15].

iii. Thin QR decomposition. The thin QR decomposition of matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \mathbf{QR},$$

where $\mathbf{Q} \in \mathbb{C}^{n_R \times n_T}$ has orthonormal columns, and $\mathbf{R} \in \mathbb{C}^{n_T \times n_T}$ is a lower-triangular matrix. One of the main benefits of this decomposition is that it can be done using a very simple algorithm, based on Gram-Schmidt orthogonalization [15]. Finding $\mathbf{H}^+$ is straightforward: column i of $\mathbf{H}^+$ is the vector $\mathbf{x}$ that solves

$$\mathbf{Rx} = (\mathbf{Q}^\dagger)_i, \tag{4.2}$$

where $(\mathbf{Q}^\dagger)_i$ is row i of $\mathbf{Q}^\dagger$.

iv. QR decomposition. Similar to thin QR, except that $\mathbf{Q} \in \mathbb{C}^{n_R \times n_R}$ is orthonormal; $\mathbf{R} \in \mathbb{C}^{n_R \times n_T}$ is lower-triangular. The process to find $\mathbf{H}^+$ is similar to the one described above.

The QR decomposition is more complex than its thin counterpart; it is still interesting, however, because $\mathbf{Q}$ and $\mathbf{R}$ can be updated (instead of re-calculated from scratch) when $\mathbf{H}$ has a rank-1 change, which means that one row or column is added or deleted. Recall from the V-BLAST algorithm that at each step a column is removed from $\mathbf{H}$; the QR decomposition with updates is perfectly suited for this situation.

4.2.1 Numerical stability results

No numerical stability problems were detected in simulation[2]. For identical data, channel realizations, and noise values, all four methods exhibit exactly the same *BER*, at least up to five decimal places; and all errors occur in the same bit positions. In cases where the data is not the same, all methods are within a fraction of a percentage point of each other.

The conclusion is that the choice of an MPPI calculation method can be based solely on performance.

4.2.2 Complexity measurements

The complexity of each MPPI calculation method was determined by simulation. A V-BLAST simulator that can be programmed to use each method was implemented; this simulator uses complexity counters to count and classify every single operation performed, whether algebraic (addition, multiplication, square root, etc.) or memory related (read or write).

The most general complexity measurement is simply the total number of operations required per received information bit. This number, denoted by O_b, gives a first approximation to the complexity of V-BLAST.

[2]All simulations were done in double precision on 32-bit x86 Intel Pentium processors.

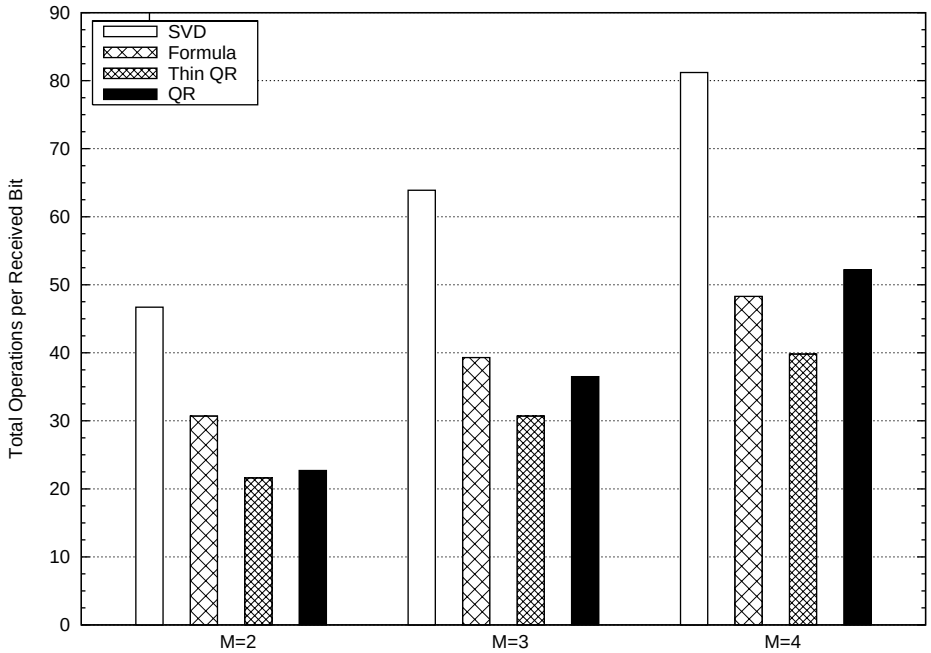

Figure 4.1: Total number of operations for a MIMO system with $n_T = 2$ and $n_R = 2, 4, 6$; $L = 10$.

Figures 4.1 through 4.4 show the values of O_b obtained for MIMO systems with several antenna combinations (the same that were studied in chapter 2), block size $L = 10$, and using 16-QAM[3]. In the figures, each bar type corresponds to a different MPPI calculation method: *SVD* for singular value decomposition, *Formula* for equation (4.1), and the two QR decomposition methods.

These results show that the singular value decomposition is not only much more complex than the other methods, but also that it is very sensitive to the value of n_R. Since its numerical stability has been shown to lack any advantage over other methods for this specific case, it is ruled out as a viable alternative.

The results of using equation (4.1) show that the simplicity of an expression has no bearing on the final complexity of its implementation. The QR decompositions, in particular, are more difficult to understand and implement; however, they require fewer operations.

The QR decomposition with updates has similar complexity to the thin QR for $n_T = n_R$; however, its complexity grows more rapidly with n_R. This is due to the different dimensions of the matrix $\mathbf{R}$ for each method; recall from equation (4.2) that calculating the MPPI involves solving n_T linear systems. When QR decomposition is used, the dimension of $\mathbf{R}$ at step i of V-BLAST is $n_R \times (n_T - i + 1)$, whereas in the case of the thin QR method, the size of $\mathbf{R}$ is $(n_T - i + 1) \times (n_T - i + 1)$. This explains the sensitivity of the QR method to the number of receive antennas;

[3]All simulations in this thesis were run until 2000 block errors were found.

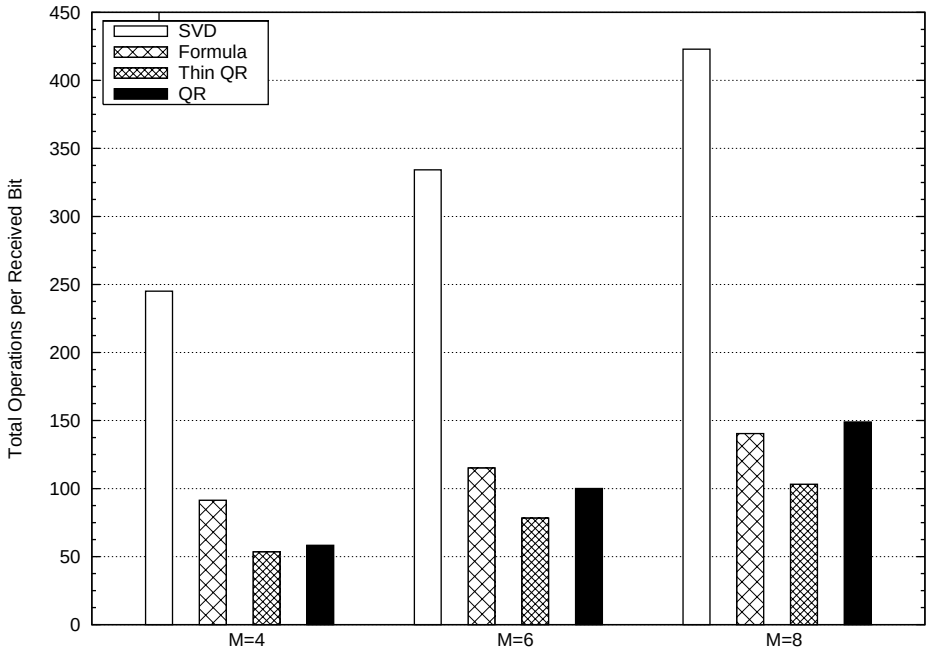

Figure 4.2: Total number of operations for a MIMO system with $n_T = 4$ and $n_R = 4, 6, 8$; $L = 10$.

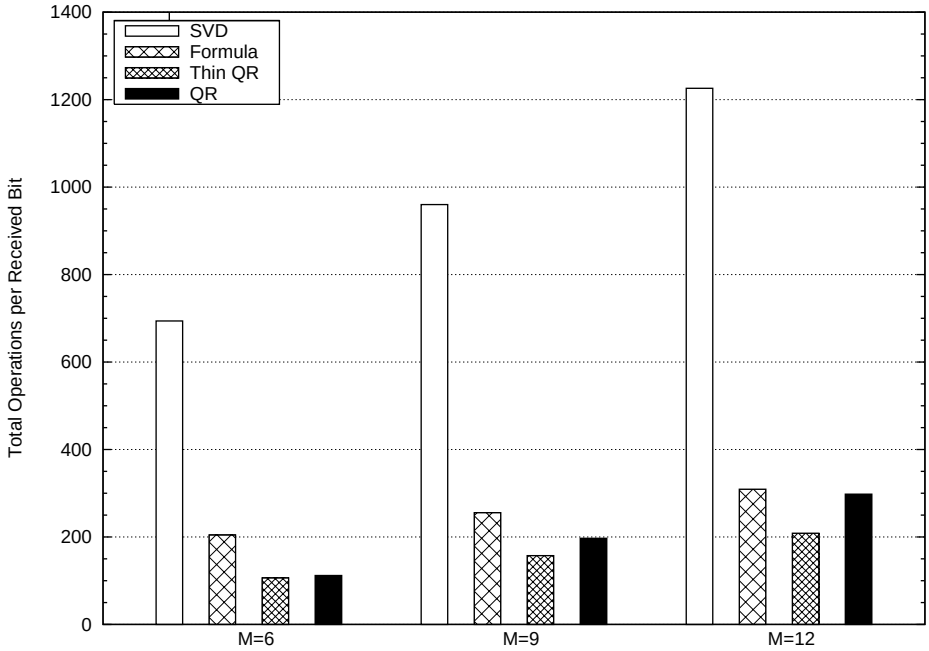

Figure 4.3: Total number of operations for a MIMO system with $n_T = 6$ and $n_R = 6, 9, 12$; $L = 10$.

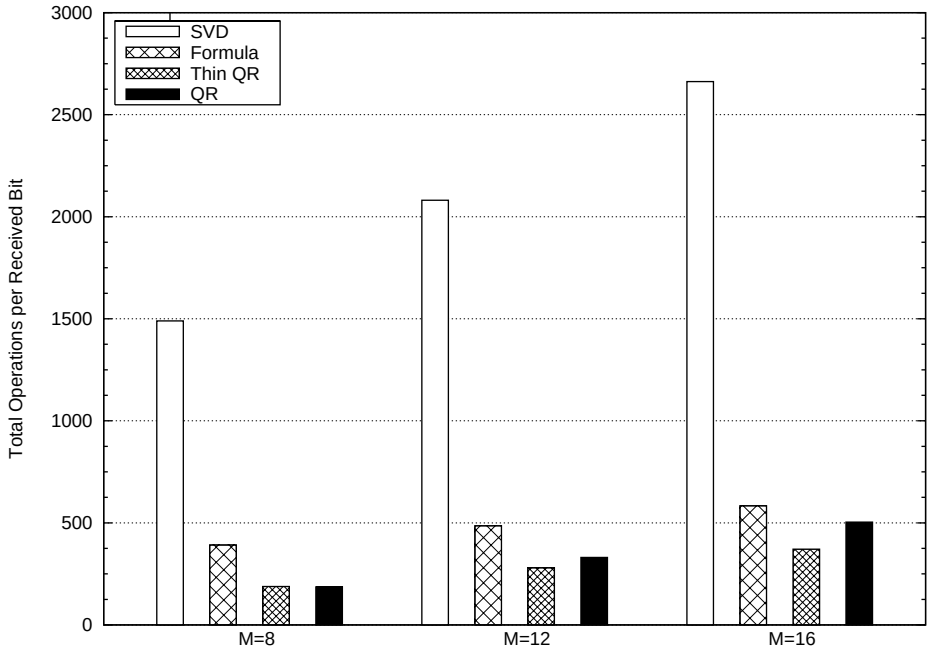

Figure 4.4: Total number of operations for a MIMO system with $n_T = 8$ and $n_R = 8, 12, 16$; $L = 10$.

ultimately, the advantage of being able to update $\mathbf{Q}$ and $\mathbf{R}$ instead of calculating them from scratch at each iteration proves insufficient to compensate for the size of $\mathbf{R}$. Thin QR decomposition's property of reducing both dimensions of $\mathbf{R}$ at each iteration will be further explored in the next chapter.

These results suggest that the thin QR decomposition method is the most attractive way to find the MPPI in a V-BLAST implementation. Henceforth, all following results are based on it.

It is worth noting, however, that even using thin QR, the operation count remains too large for high-speed applications. For example, in a medium-sized (4,6) system, $O_b \approx 80$. Therefore, a typical application like 10Base-T ETHERNET (which operates at 10Mbps) would require a receiver capable of executing 800 million operations per second, assuming for the moment that all operations are equivalent.

4.3 Complexity of thin QR V-BLAST

As noted in the previous section, the total number of operations per bit O_b only gives a general idea of the complexity; a more detailed view is necessary to understand how to reduce it.

This section aims to classify the operations required by V-BLAST according to their purpose, as a first step towards designing a receiver architecture that is suited to the task at hand.

n_R	$\mathbf{O_b}$	A_{mm}	A_o	P_{mm}	P_o	D_o	S_o	M_{mm}	M_{cm}	M_{zm}	M_o
2	21.6	18.5	15.1	18.5	4.8	1.4	0.2	23.1	5.3	0.5	12.1
3	30.7	19.5	12.7	19.5	5.1	1.5	0.1	24.4	5.6	0.4	10.9
4	39.8	20.0	11.4	20.1	5.2	1.5	0.1	25.1	5.8	0.3	10.3

Table 4.1: Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 2$ and $L = 10$.

The following types of operations are proposed. Each operation type is identified with a symbol.

- Additions
  - Additions made during matrix multiplications (A_{mm})
  - Other additions (A_o)
- Multiplications
  - Multiplications made during matrix multiplications (P_{mm})
  - Other multiplications (P_o)
- Divisions (D_o)
- Square Roots (S_o)
- Memory operations (write or read)
  - Memory operations made during matrix multiplications (M_{mm})
  - Memory operations made to copy matrices (M_{cm})
  - Memory operations made to reset (or zero) matrices (M_{zm})
  - Other memory operations (M_o)

O_b is then defined as

$$O_b = A_{mm} + A_o + P_{mm} + P_o + D_o + S_o + M_{mm} + M_{cm} + M_{zm} + M_o.$$

Note that this is only one of many possible classifications of the types of operations performed. Since V-BLAST and thin QR operate mostly on matrices, however, this classification seems attractive. For instance, the memory operations that relate to matrix clearing and copying are good examples of the kind of operations that can be done very quickly in a properly designed hardware implementation.

In tables 4.1 through 4.4 each operation type is expressed as a percentage of O_b, for $L = 10$. These results are a first attempt at identifying which types of operation should be optimized first.

Returning to the example of a (4,6) system, it can be seen in table 4.2 that around 50% of all operations (or approximately 40 operations per received bit) are performed during matrix multiplications. This means there is a maximum performance gain of

n_R	$\mathbf{O_b}$	A_{mm}	A_o	P_{mm}	P_o	D_o	S_o	M_{mm}	M_{cm}	M_{zm}	M_o
4	53.5	14.9	15.6	14.9	10.2	1.9	0.1	18.6	7.0	0.5	15.9
6	78.4	15.3	14.7	15.2	10.5	1.9	0.1	19.1	7.1	0.3	15.5
8	103.2	15.5	14.1	15.5	10.6	1.9	0.1	19.3	7.1	0.2	15.2

Table 4.2: Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 4$ and $L = 10$.

n_R	$\mathbf{O_b}$	A_{mm}	A_o	P_{mm}	P_o	D_o	S_o	M_{mm}	M_{cm}	M_{zm}	M_o
6	106.4	11.2	18.4	11.2	14.8	2.0	0.1	14.1	7.2	0.4	20.2
9	157.5	11.4	18.0	11.4	14.9	2.0	0.1	14.2	7.3	0.2	20.0
12	208.6	11.4	17.8	11.4	15.1	2.0	0.1	14.4	7.3	0.2	19.9

Table 4.3: Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 6$ and $L = 10$.

n_R	$\mathbf{O_b}$	A_{mm}	A_o	P_{mm}	P_o	D_o	S_o	M_{mm}	M_{cm}	M_{zm}	M_o
8	187.9	8.4	21.0	8.5	18.1	1.9	0.1	10.6	6.9	0.4	23.7
12	279.5	8.5	20.8	8.5	18.3	1.9	0.1	10.7	7.0	0.2	23.6
16	371.0	8.6	20.7	8.6	18.4	1.9	0.1	10.7	7.0	0.2	23.5

Table 4.4: Operations performed by V-BLAST classified by type, expressed as a percentage of O_b; $n_T = 8$ and $L = 10$.

50% to be obtained from optimizing matrix products. Perfect (or infinite) optimization would thus lower the ETHERNET receiver processing requirements considered above from 800 million operations per second to 400 million.

Furthermore, matrix clearing and matrix copying operations account for another 7.4% of the total. Optimization of these operations, added to that of the matrix products, would lower the required total to around 34 operations per received bit, or 340 million operations per second for 10Mbps reception.

4.4 The role of L in V-BLAST complexity

In chapter 3, V-BLAST was divided in two parts, the *block setup phase* and the *symbol estimation phase*. The block setup phase is done once per block, and the symbol estimation phase is repeated L times. Clearly, increasing L will lower the receiver complexity, at the expense of an increase in *BLER*. This section illustrates some general aspects of the behavior of V-BLAST as a function of L, without undertaking a more rigorous analysis.

Let O_{setup} be the number of operations required during the setup phase, and O_{est} the number required during the symbol estimation phase. Then, $O_b = O_{setup} + O_{est}$. If bit errors are independent and uniformly distributed, then *BLER* is approximated by:

$$BLER \approx 1 - (1 - BER)^B, \tag{4.3}$$

where $B = b \cdot n_T \cdot L$ is the number of information bits per block; recall that b is the number of bits per symbol. The minimum possible value of L is one, which means the *BLER* is lower-bound by $1 - (1 - BER)^{b \cdot n_T}$. Such a system would also exhibit the largest complexity.

For a given n_T, O_b increases in a roughly linear way with n_R; let $O_{setup} = x \cdot n_R$ and $O_{est} = y \cdot n_R$ for constants x, y that depend on the details of the particular V-BLAST implementation. Then,

$$\begin{aligned} O_b &\approx \frac{O_{setup} + L \cdot O_{est}}{B} && (4.4a) \\ &= \frac{O_{setup}}{n_T b L} + \frac{O_{est}}{n_T b} \\ &= \frac{1}{n_T b}\left(\frac{O_{setup}}{L} + O_{est}\right) \\ &\propto \frac{O_{setup}}{L} + O_{est} \\ &\propto n_R \cdot \left(\frac{x}{L} + y\right). && (4.4b) \end{aligned}$$

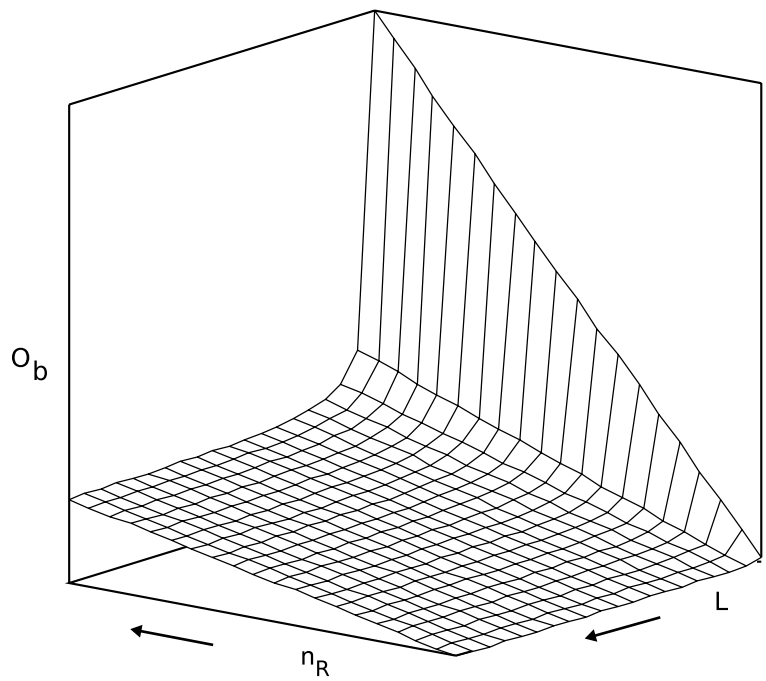

Figure 4.5: Behavior of O_b in V-BLAST as a function of n_R and L. The arrows indicate the direction of growth of n_R and L.

From equation (4.4b) it can be seen that an increase in L causes an inversely-proportional decrease in O_b; likewise, an increase in n_R causes a linear increase in O_b. Figure 4.5 shows the behavior of O_b for several combinations of n_R and L.

It is interesting to find how *BLER* improves as a function of O_b. Taking (4.3) and (4.4a) together it is found that:

$$BLER(O_b) \approx 1 - (1 - BER)^{\frac{n_R(x+Ly)}{O_b}}. \tag{4.5}$$

Figure 4.6 shows the behavior of *BLER* as a function of O_b; it can be seen that, after a point, increasing the complexity of the system will not significantly improve its *BLER* performance. In other words, reducing L in order to improve *BLER* brings diminishing returns after a certain point.

Figure 4.6 suggests that, for a given (n_T,n_R) system and a particular V-BLAST implementation, there is a kind of "operation point" with respect to *BLER* and L; this point can be chosen according to the desired cost (that is, O_b) and the desired block error rate.

4.4.1 Simulation results

Figure 4.7 shows the effect of L on V-BLAST for a (4,4) system. The shape of the curve is just as predicted by equation (4.4). As L increases, O_b tends to O_{setup}. It appears that for this particular system, a value of $L \approx 10$ would offer a good

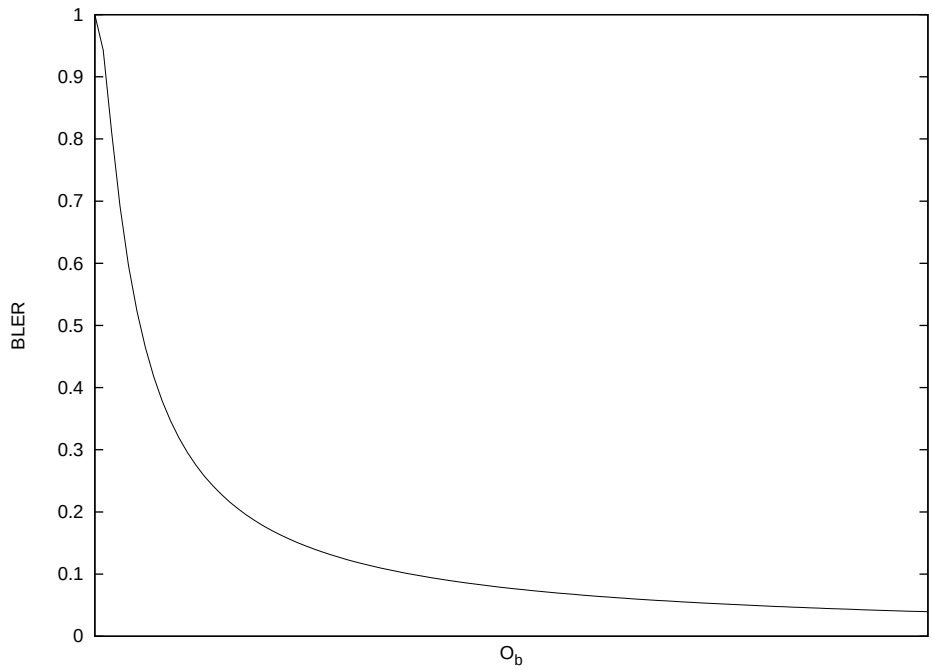

Figure 4.6: General shape of *BLER* as a function of O_b in a V-BLAST receiver.

compromise between block error rate and receiver complexity. Naturally, for a given channel, increasing L increases the data rate, which introduces yet another consideration.

4.5 V-BLAST error rate simulation results

In this section, simulation results of *BLER* as a function of average SNR are presented. Figures 4.8 through 4.11 show the performance curves for V-BLAST for several antenna combinations, block length $L = 10$, and 16-QAM.

Two important conclusions can be drawn from these figures. The first is that symmetrical (n_T,n_T) V-BLAST receivers have a poor *BLER* compared to asymmetrical antenna configurations. For example, the difference between systems with (8,8) and (8,12) antennas is 14dB at $BLER = 10^{-2}$. In contrast, increasing the number of receive antennas from $1.5n_T$ to $2n_T$ does not have as strong an effect: the difference between (8,12) and (8,16) is only 2dB at the same value of *BLER*.

Another interesting conclusion is that increasing n_R has a pronounced effect on the slope of the curves, which indicates an increase in the system's diversity. This confirms the result presented in Section 2.5, which establishes that V-BLAST diversity increases with $n_R - n_T$. Again, the effect is less pronounced as n_R increases.

These results have been compared with others available in the literature (for example, in [26]) to verify their correctness.

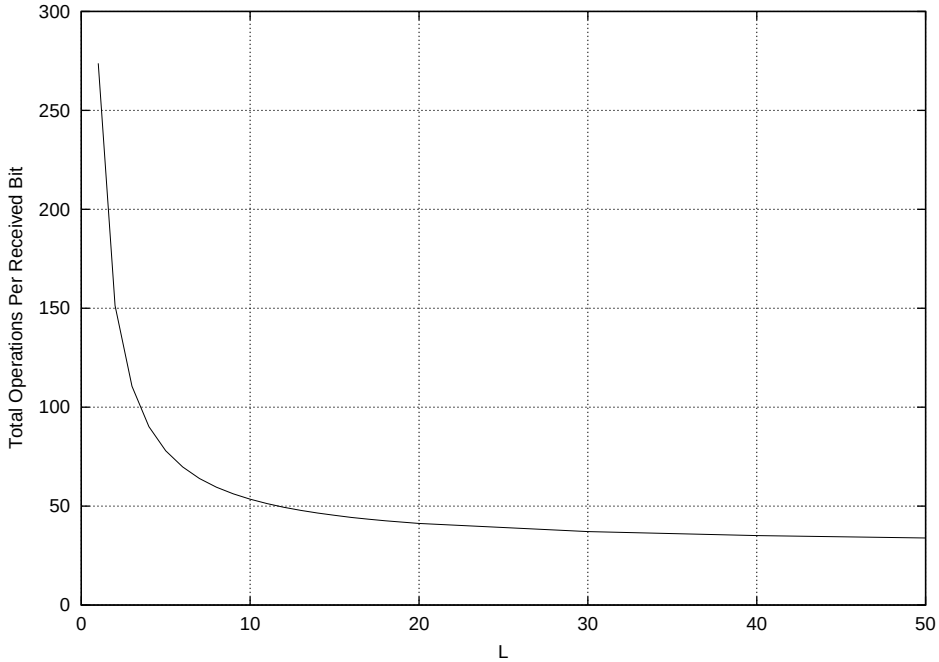

Figure 4.7: O_b as a function of L for a (4,4) V-BLAST receiver.

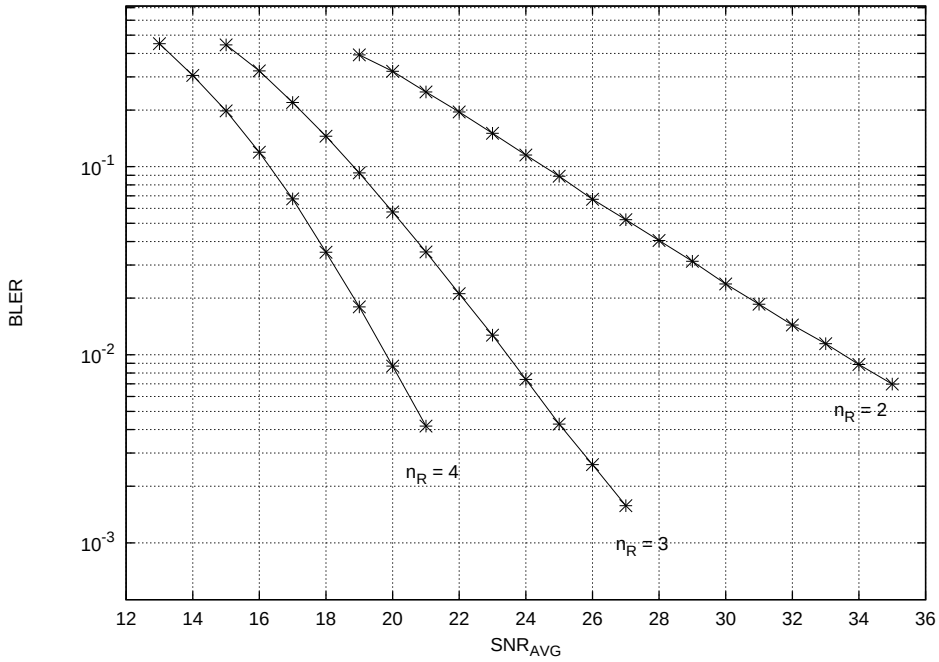

Figure 4.8: *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 2$, $L = 10$.

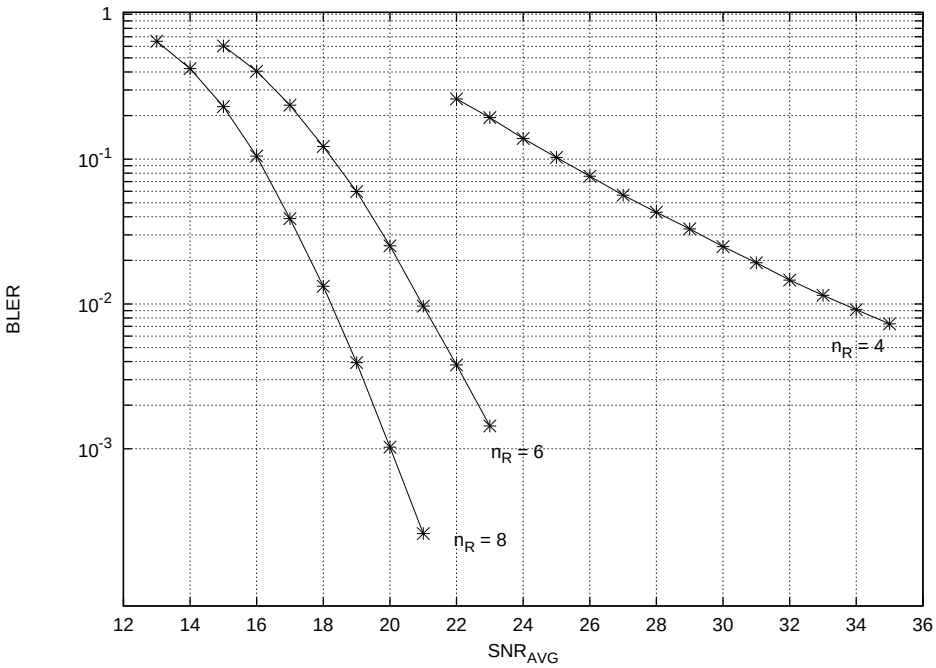

Figure 4.9: *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 4$, $L = 10$.

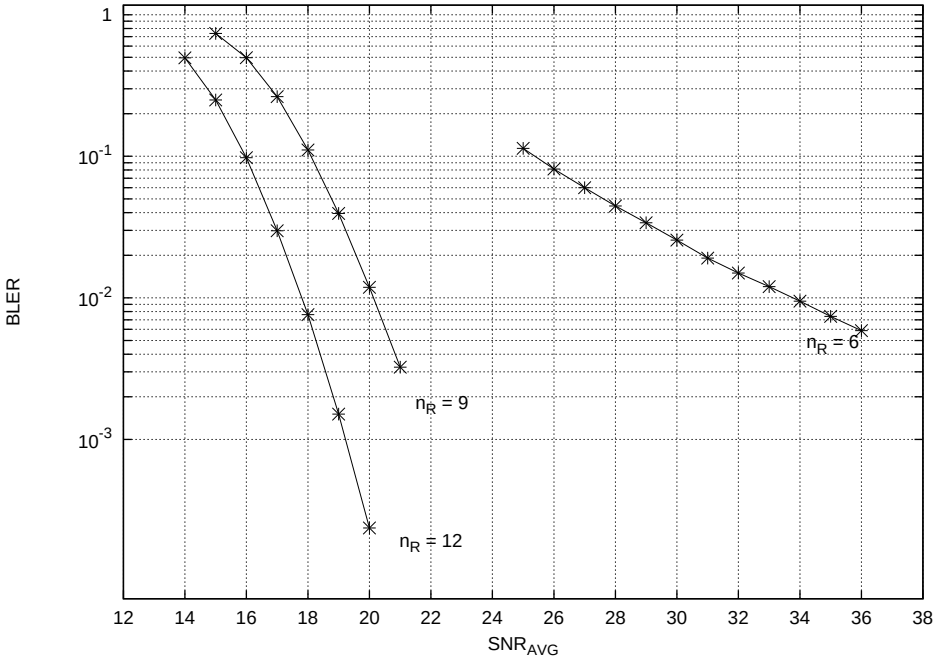

Figure 4.10: *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 6$, $L = 10$.

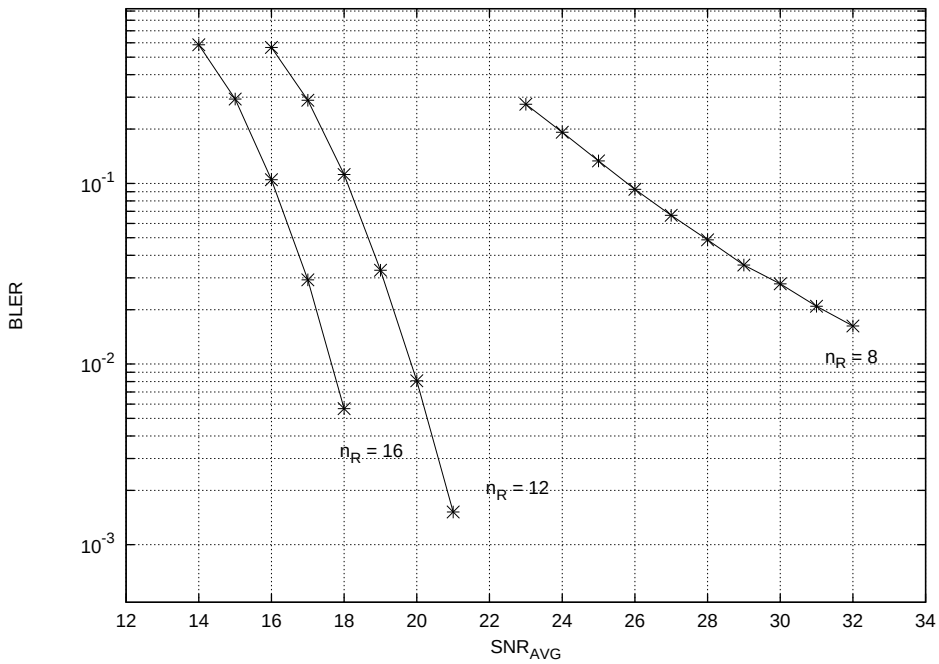

Figure 4.11: *BLER* as a function of average SNR for a V-BLAST receiver with $n_T = 8$, $L = 10$.

In view of the large spectral efficiency that V-BLAST makes available, these results are very interesting. They indicate that V-BLAST has the potential to be a viable candidate in the search for solutions to improve wireless communications.

4.6 Experimental results

The V-BLAST algorithm was implemented in a Texas Instruments TMS320C6711 floating-point digital signal processor (*DSP*) running at 150MHz and rated at 900 MFLOPS [27]. The algorithm's complexity was measured in instruction cycles using a code execution profiler; since the instruction cycle time is known to be $I_c = 6.7$ns, it is possible to estimate real-world performance, as well as to extrapolate the results to more recent processors.

Table 4.5 presents the instruction cycles needed per received information bit, and the corresponding data rate in kbps, for a 6711 DSP running the V-BLAST algorithm for a (2,3) MIMO system. Likewise, an extrapolation of the data rate achievable on a current-generation 6713 processor, which has a cycle time of 3.3ns and is rated at 1800MFLOPS, is shown.

Figure 4.12 shows the instruction cycles required by the 6711 DSP for the (2,3) V-BLAST receiver, as a function of block length L. The general behavior predicted by 4.6 and equation (4.4) is confirmed by experiment. A very rapid decrease in complexity is seen between $L = 1$ and $L = 20$, followed by a slow decrease between

L	**Clock cycles per bit**	**Data Rate** (kbps, $I_c = 6.7$ns)	**Data Rate** (kbps, $I_c = 3.3$ns)
5	4583	32.7	65.4
10	2558	58.6	117.2
15	1911	78.5	157.0
20	1580	94.9	189.8
40	1099	136.5	273.0
60	925	162.2	324.4
80	838	179.0	358.0
100	796	188.4	376.9

Table 4.5: Clock cycles and data rates as a function of L. Results for instruction cycle $I_c = 6.7$ns were obtained by running V-BLAST with $n_T = 2$ and $n_R = 3$ on a Texas Instruments 6711 DSP. Results for $I_c = 3.3$ns are an extrapolation to a current-generation DSP.

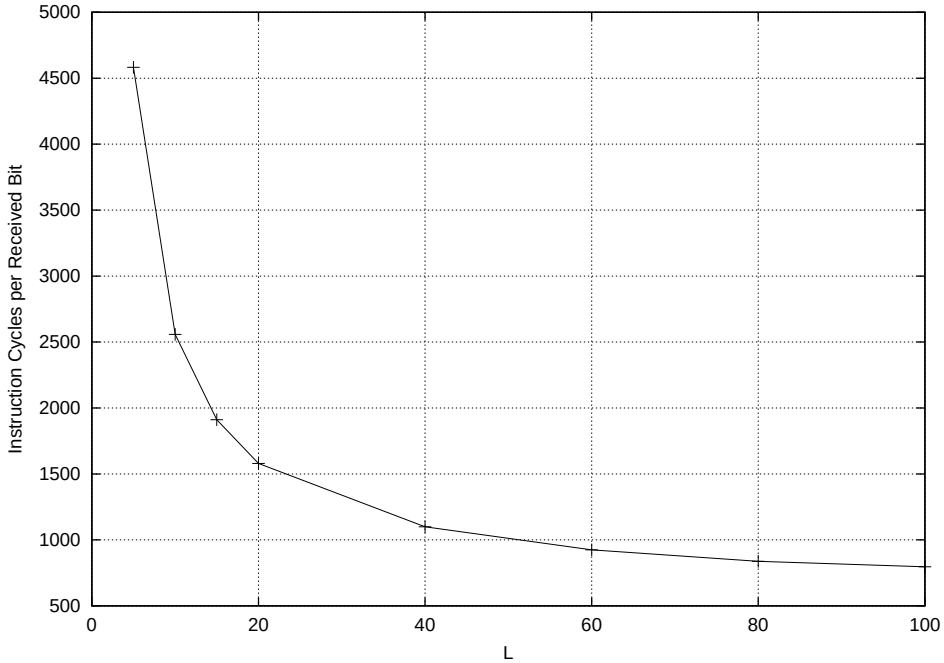

Figure 4.12: Instruction cycle count per received bit of a Texas Instruments 6711 DSP running V-BLAST with $n_T = 2$ and $n_R = 3$, as a function of L.

$L = 20$ and $L = 100$. This is, of course, reflected in the data rate: it practically doubles when L increases from 5 to 10, but increases merely 14% between $L = 60$ and $L = 100$.

Even a top-of-the-line, current general-purpose DSP is far from reaching ETHERNET data rates, even for the small (2,3) system under consideration here. These results clearly show that, if V-BLAST is to be used in practical applications, a drastic reduction in the number of operations needed per bit is essential. An alternative is to design a special-purpose hardware architecture; the disadvantages of this are high cost and implementation difficulty.

It should be emphasized, though, that these results are prospective in nature. The objective when exploring the performance of V-BLAST on this particular DSP is not to create a practical application, but rather to provide results on the feasibility of such a project.

4.7 V-BLAST memory requirements

Table 4.6 describes the memory requirements of V-BLAST when using the thin QR decomposition. The purpose of each matrix is stated, along with its dimensions and whether it is real or complex. When appropriate, the algorithm step number where the matrix is used is mentioned (see section 2.4).

Complex matrices require double the space of real ones. Taking this into account, the total requirements are given by

$$n_T(5 + 2n_T) + n_R(10 + n_T + n_T^2).$$

It is interesting that the memory requirements grow linearly with n_R, but grow with the square of n_T. The memory sizes are given in words; the actual memory needed by the algorithm in bits depends on the word size used. The channel matrix and the pseudo-inverse for the current step $\mathbf{A}$ need to be copied because certain operations destroy the matrix they operate on.

4.8 Loss of BER performance caused by sub-optimal ordering

As first mentioned in section 2.3, the optimal symbol estimation order is given by selecting, at each step, the symbol that has maximum SNR. The post-detection SNR of symbol a_{k_i} is given by

$$\gamma_{k_i} = \frac{|a_{k_i}|^2}{2N_0\|\mathbf{w}_{k_i}\|^2}.$$

An ordering constructed according to this equation is called *optimal*. (Strictly speaking, the optimal ordering should use the instantaneous power of the noise

Description	**Matrix Size**	**Real or Complex**
Stores vector $\mathbf{k}$ (step 4)	$1 \times n_T$	Real
Needed to establish optimal ordering (step 5)	$1 \times n_T$	Real
Estimated vector $\hat{\mathbf{a}}$ (step 12)	$1 \times n_T$	Complex
Received vector $\mathbf{r}$	$1 \times n_R$	Complex
Matrix $\mathbf{Q}$ (step 3)	$n_T \times n_R$	Complex
Matrix $\mathbf{R}$ (step 3)	$n_T \times n_T$	Complex
Channel matrix $\mathbf{H}$	$n_T \times n_R$	Complex
Copy of $\mathbf{H}$	$n_T \times n_R$	Complex
Copy of $\mathbf{A}$	$n_T \times n_R$	Complex
n_T pseudo-inverses	$n_T \times n_T$, $(n_T - 1) \times n_T$, $\ldots$, $1 \times n_T$	Complex

Table 4.6: Matrices required by V-BLAST. A description of their function, their dimensions, and whether they are real or complex.

affecting a_{k_i} instead of $2N_0$. However, this is too unrealistic to be worth considering).

When the signal constellation contains symbols with unequal energies, the ordering of the symbols' SNR cannot be accurately determined without knowledge of the energy of each symbol $|a_{k_i}|^2$. In this case, V-BLAST uses $1/\|\mathbf{w}_{k_i}\|^2$ to estimate the SNR; the ordering thus obtained is simply called *V-BLAST ordering*. It is, in fact, the best ordering that can be achieved given the information available to V-BLAST[4].

In this respect, constellations whose symbols have equal energy are especially attractive, because V-BLAST's detection ordering in this case is optimal. Their disadvantages, as is well known, are a smaller distance between symbols and more difficult labeling, when compared to lattice constellations like M-QAM. A more detailed analysis of the performance of different constellation types in a V-BLAST receiver remains to be done. Whether a coding scheme that improves the detection ordering exists, and how it would affect the complexity of the receiver, is still an open question. Some interesting attempts to improve the detection order are reported in [21, 28, 29, 30]

The loss caused by this sub-optimal ordering has been determined by simulation[5]. In figures 4.13 through 4.15, the *BER* performance between three ordering methods is compared: optimal ordering, V-BLAST ordering, and fixed ordering.

[4]This ordering is commonly called *optimal* in the literature.

[5]Note that simulating optimal ordering does not entail providing V-BLAST with the actual transmitted symbols; all it needs is their energy. What is being determined is just the performance loss due to sub-optimal ordering, not that due to other effects like error propagation.

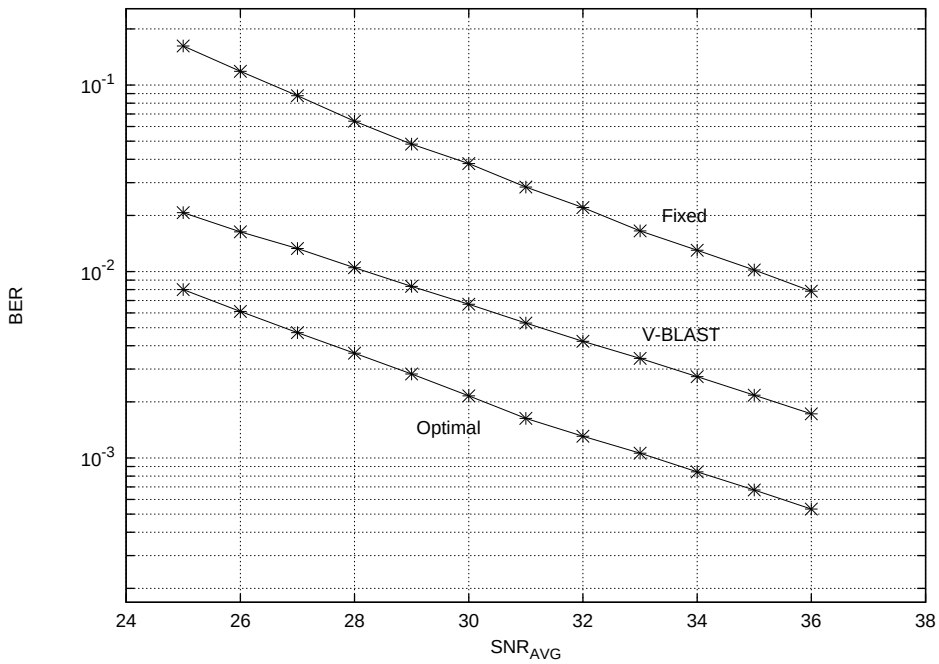

Figure 4.13: Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 6$, 16-QAM.

Fixed ordering does not take the symbols' SNR into account; symbols are estimated in a fixed, constant order. The signal constellation used is, as before, 16-QAM; symbols belonging to this constellation have one of three possible energies. The results presented are for $n_T = 6$ and $n_R = 6$, 9, and 12. The bit error rate is presented instead of the block error rate to remove any dependency on L.

The impact of the ordering method is more pronounced for $n_R = n_T$: the gain between fixed and V-BLAST ordering is 6dB, and between V-BLAST and optimal, a further 4dB. For $n_T = 6$ and $n_R = 12$, the gain has been reduced to 1dB and 0.8dB, respectively. This means that a large number of receive antennas can in large measure compensate for sub-optimal orderings.

In section 2.5 it was predicted that the ordering method would have no effect on diversity (that is, on the slope of the performance curves). In the simulations, it is seen that for $n_R > n_T$ there is in fact a small improvement on the slope as the ordering improves. This is not the case for $n_R = n_T$, where the slopes are practically identical for all three orderings.

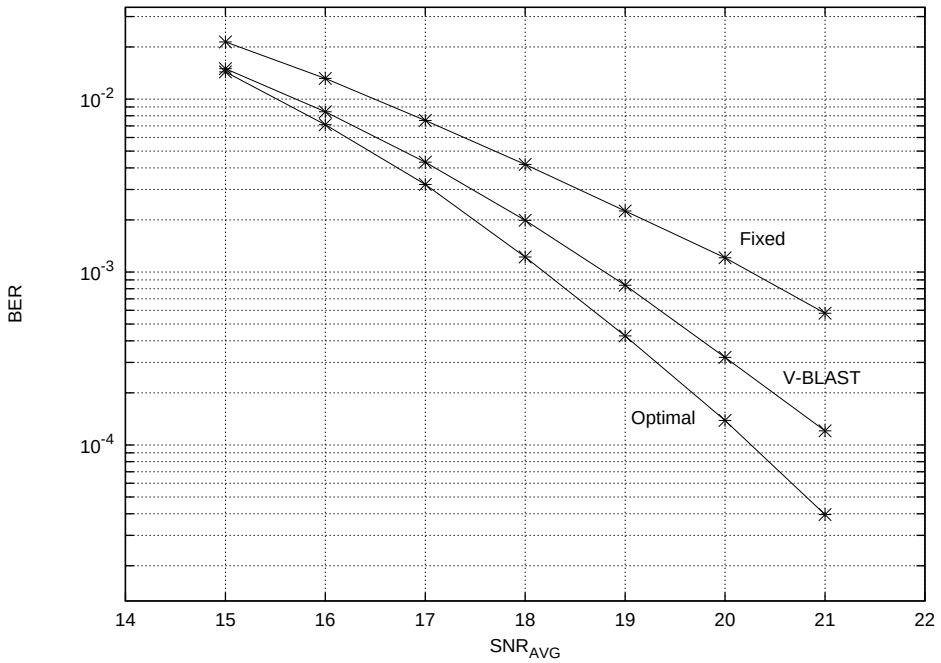

Figure 4.14: Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 9$, 16-QAM.

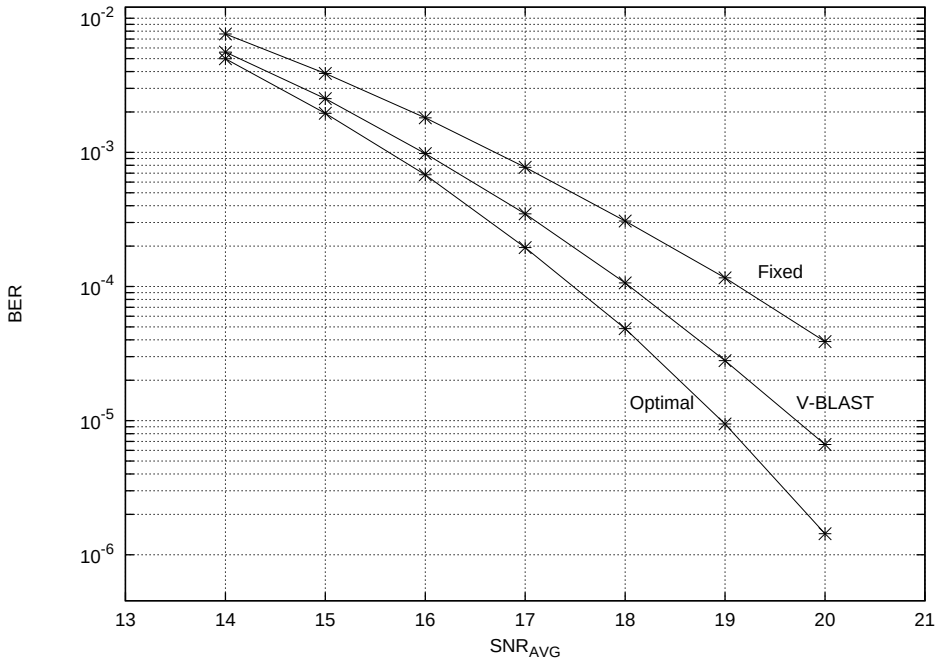

Figure 4.15: Comparison of *BER* for three different symbol detection orderings: fixed, V-BLAST, and optimal. $n_T = 6$, $n_R = 12$, 16-QAM.

Chapter 5

V-BLAST as a least-squares problem

5.1 Introduction

THE PREVIOUS chapter mentioned that the thin QR decomposition may be used to reduce the size of a part of V-BLAST from $(n_R \times n_T)$ to $(n_T \times n_T)$, potentially improving its performance. Let $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q} \in \mathbb{C}^{n_R \times n_T}$ is a matrix with orthonormal columns, and $\mathbf{R} \in \mathbb{C}^{n_T \times n_T}$ is upper-triangular. Then, the least-squares problem of finding the vector $\mathbf{x}$ that minimizes $\|\mathbf{Hx} - \mathbf{b}\|^2$ is equivalent to the problem of finding $\mathbf{x}$ that minimizes $\|\mathbf{Rx} - \mathbf{Q}^\dagger\mathbf{b}\|^2$. Since $\mathbf{R}$ is triangular, this problem is readily solved.

The QR decomposition has been applied to V-BLAST reception for a long time ([31] is one of the first times the idea is referenced explicitly). In this thesis, the technique is applied to V-BLAST in what is believed to be a novel way; namely, it is used in both phases of V-BLAST to speed up the calculation of both the pseudo-inverses and the symbol estimates without degrading the error rate. Some of the consequences of modifying V-BLAST in this way are explored; the modified algorithm is denoted V-LS. It is proved that V-BLAST's bit-error rate is not altered in any way; in fact, it is proved that the output of V-LS is exactly the same as the original's.

What is probably the fastest version of V-BLAST known to date is described in [32] (see also [33]). This algorithm, denoted here by V-SQR, also uses the thin QR decomposition, but it employs an heuristic to avoid having to calculate any pseudo-inverses (hence its speed) at the cost of a slightly higher error rate.

In this chapter the three algorithms are compared, and it is found that under certain circumstances, V-LS is even faster than V-SQR. Scenarios where the error rate of V-SQR degrades significantly are also described.

5.2 QR Decomposition in V-BLAST

V-BLAST can be seen as a least-squares solver of a system of linear equations perturbed by noise, with the added constraint that the solution must be an element of the constellation set S. If the noise vector $\mathbf{n}$ is zero, and the received vector is $\mathbf{r}$ (see equation (2.1)), then V-BLAST gives the least-squares solution to the problem of finding $\hat{\mathbf{a}}$ such that $\|\mathbf{H}\hat{\mathbf{a}} - \mathbf{r}\|^2$ has minimum norm.

In the presence of noise, V-BLAST will not always find the best solution in the least-squares sense; on the other hand, it has lower complexity than methods that find better solutions, such as lattice decoding or maximum-likelihood decoding.

The basic idea is to use QR decomposition to modify the original system model, $\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n}$, into

$$\begin{aligned} \mathbf{r} &= \mathbf{Q}\mathbf{R}\mathbf{a} + \mathbf{n} \\ \mathbf{Q}^\dagger\mathbf{r} &= \mathbf{R}\mathbf{a} + \mathbf{Q}^\dagger\mathbf{n} \\ \tilde{\mathbf{r}} &= \mathbf{R}\mathbf{a} + \tilde{\mathbf{n}}, \end{aligned} \tag{5.1}$$

The statistics of the noise are unchanged, since $\mathbf{Q}$ has orthonormal columns. The MIMO system described by equation (5.1) has a channel matrix $\mathbf{R}$ that is now square, and has smaller dimensions than the original channel matrix $\mathbf{H}$. This reduces the complexity of the two parts of the algorithm, the *block setup phase* and the *symbol estimation phase*, at the cost of performing the QR decomposition of $\mathbf{H}$, and calculating $\tilde{\mathbf{r}}$.

Making use of these ideas, a new algorithm, called V-LS, is proposed below.

Now, it will be proved that V-LS and V-BLAST as originally proposed are equivalent; that is, they produce exactly the same output.

Let $\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n}$. It will be proved that the vector of estimated symbols $\hat{\mathbf{a}}_\mathbf{V}$ produced by V-BLAST is equal to $\hat{\mathbf{a}}_\mathbf{L}$, the vector produced by V-LS.

The key point is the calculation of y in step $4.b$ of V-BLAST and in step $5.b$ of V-LS. If both algorithms compute the same y then they will produce the same output.

First, it is necessary to establish that the row norms of $\mathbf{R}^+$ equal those of $\mathbf{H}^+$. Let $\mathbf{x} = (\mathbf{R}^{-1})_j$ and $\mathbf{y} = (\mathbf{H}^+)_j$ be row j of $\mathbf{R}^{-1}$ and $\mathbf{H}^+$, respectively. The squared norm of $\mathbf{y}$ is given by

$$\begin{aligned} \|\mathbf{y}\|^2 &= \mathbf{y}\mathbf{y}^\dagger \\ &= \left(\mathbf{x}\mathbf{Q}^\dagger\right)\left(\mathbf{x}\mathbf{Q}^\dagger\right)^\dagger \\ &= \left(\mathbf{x}\mathbf{Q}^\dagger\right)\mathbf{Q}\mathbf{x}^\dagger \\ &= \mathbf{x}\mathbf{x}^\dagger \\ &= \|\mathbf{x}\|^2, \end{aligned}$$

Algorithm 3 V-LS

Input: an $n_R \times n_T$ matrix $\mathbf{H}$, a set of L $n_R \times 1$ vectors $\mathbf{r}_i$, $i = 1, \ldots, L$, and a signal constellation S.

Output: a set of L $n_T \times 1$ vectors $\hat{\mathbf{a}}_i$. Each vector $\hat{\mathbf{a}}_i$ is such that its elements are in S and $\mathbf{H}\hat{\mathbf{a}}_i = \mathbf{r}_i + \mathbf{v}_i$, where $\mathbf{v}_i$ is an error vector.

```
Compute an n_R × n_T matrix Q with orthonormal columns and an n_T × n_T
    upper-triangular matrix R, such that H = QR
A = R
for i = 1 to n_T do
    Find G_i = A^+
    Let k_i = argmin_j ||(G_i)_j||
    Let O_i equal to corresponding column in R
    Remove column k_i of A
end for
for j = 1 to L do
    Let x = Q^† · r_j
    for i = 1 to n_T do
        Let w equal to row k_i of G_i
        Let y = wx
        Let â_{j,O_i} = f_q(y)
        Let z equal to column O_i of R
        Let x = x − â_{j,o_i} z
    end for
end for
```

where the identity $\mathbf{H}^+ = \mathbf{R}^{-1}\mathbf{Q}^\dagger$ has been used. The order of symbol detection is, then, the same for both algorithms.

Let a_k be the first symbol to be detected, and let $\mathbf{w}$ be row k of $\mathbf{H}^+$. V-BLAST's estimate y_V is given by

$$\begin{aligned} y_V &= \mathbf{wr} \\ &= \mathbf{w}(\mathbf{Ha} + \mathbf{n}) \\ &= a_k + \mathbf{wn}. \end{aligned} \tag{5.2}$$

Let $\mathbf{v}$ be the kth row of $\mathbf{R}^{-1}$. Then, V-LS's estimate y_L is given by

$$\begin{aligned} y_L &= \mathbf{vx} \\ &= \mathbf{v}(\mathbf{Ra} + \mathbf{Q}^\dagger\mathbf{n}) \\ &= a_k + \mathbf{vQ}^\dagger\mathbf{n} \\ &= a_k + \mathbf{vRH}^+\mathbf{n} \\ &= a_k + \mathbf{wn}. \end{aligned} \tag{5.3}$$

From equations (5.2) and (5.3), it is clear that both algorithms produce the same estimate. The argument can be extended to the estimation of the remaining symbols.

It has been verified in simulation that the *BER* and *BLER* performance of V-BLAST and V-LS are in fact identical.

5.3 The Sorted QR algorithm

This variant of V-BLAST was first proposed in [32]. If the QR decomposition is used as in (5.1), $\hat{\mathbf{a}}$ can be found iteratively. For instance, $\hat{a}_{n_T}$ would be given by

$$\hat{a}_{n_T} = f_q\left(\frac{\tilde{r}_{n_T}}{R_{n_T,n_T}}\right),$$

and, in general, for $1 \leq i < n_T$,

$$\hat{a}_i = \frac{\tilde{r}_k - \hat{d}_k}{R_{k,k}},$$

where the term $\hat{d}_k$ can be considered as interference, and is given by

$$\hat{d}_k = \sum_{j=k+1}^{n_T} R_{k,j}\hat{a}_j.$$

The basic insight behind V-SQR is that maximizing the SNR at each step, as V-BLAST does, is equivalent to ordering the diagonal elements of $\mathbf{R}$ $R_{k,k}$ from minimum to maximum (that is, $R_{1,1} \geq R_{2,2} \geq \cdots \geq R_{n_T,n_T}$). Then, it proceeds to find an heuristic to construct such an $\mathbf{R}$, based on finding a permutation of the columns of $\mathbf{H}$ that cause the thin QR decomposition to produce a matrix with the desired properties. The algorithm may be written as follows; its inputs and outputs are identical to those of V-BLAST.

Algorithm 4 V-SQR

1: Find the sorted-QR decomposition of $\mathbf{H}$, store it in $\mathbf{Q}$ and $\mathbf{R}$.
2: **for** $i = 1$ to L **do**
3: Let $\mathbf{x} = \mathbf{Q}^\dagger \cdot \mathbf{r}_i$
4: Find $\hat{\mathbf{a}}_i$ that solves $\mathbf{R}\hat{\mathbf{a}} = \mathbf{x}$
5: Re-arrange $\hat{\mathbf{a}}_i$ to correspond to original ordering of columns of $\mathbf{H}$
6: **end for**

Step 5 is necessary since the ordering of the rows of matrix $\mathbf{R}$ does not correspond to the original ordering of the symbols in each vector. The sorted QR algorithm provides, as output, a vector that describes the permutations made on $\mathbf{H}$ and allows the rearrangement of $\hat{\mathbf{a}}_i$.

5.4 Complexity analysis and simulation results

In comparison with V-BLAST, the block setup phase of V-LS needs an extra QR decomposition; in exchange, the pseudo-inverses are performed on an $n_T \times n_T$ matrix. In turn, the symbol estimation phase has an extra vector-matrix multiplication, but the size of all other operations now depends exclusively on n_T instead of both n_T and n_R.

As for V-SQR, its modified thin QR decomposition is more complex than the regular one, since it involves many exchanges of the columns of $\mathbf{Q}$ and $\mathbf{R}$ as they are being calculated; it also has the extra overhead of rearranging the estimated symbols.

As was done in chapter 4, the effect of the block length L on the algorithms is studied. Since both V-LS and V-SQR have a similar structure to V-BLAST, it is expected that equation (4.4) still holds.

Complexity-wise, one important aspect of V-LS and V-SQR is that only two of their operations—the initial thin QR decomposition of $\mathbf{H}$, and the calculation of $\mathbf{x} = \mathbf{Q}^\dagger\mathbf{r}$—depend in any way on n_R. For this reason, it is desirable to examine how their complexity depends on n_R for constant n_T. These questions are studied in the next subsections. Also, the *BLER* performances of V-LS and V-SQR are compared.

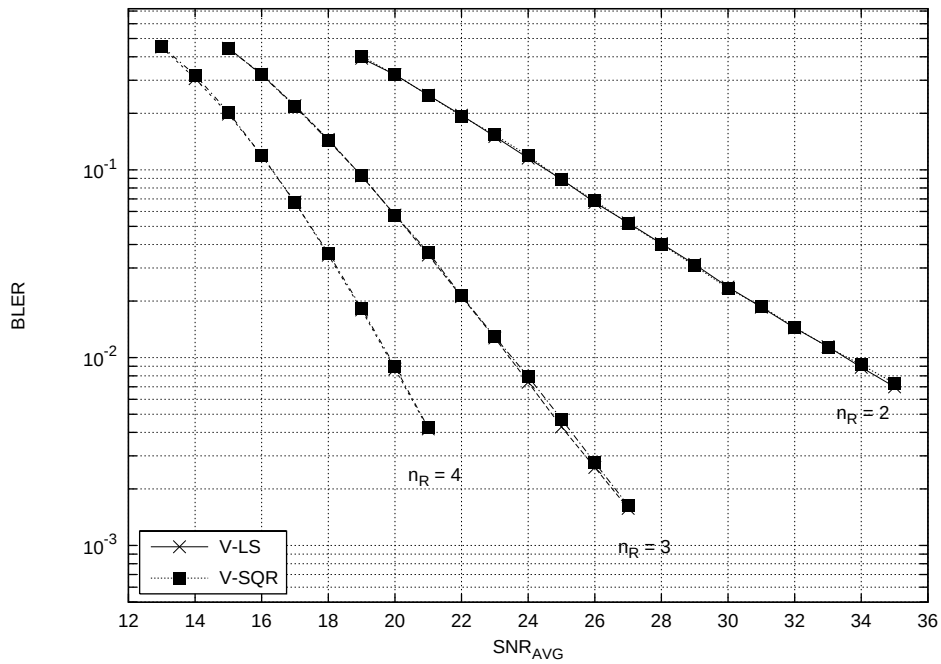

Figure 5.1: *BLER* as a function of average SNR (dB) for $n_T = 2$; $n_R = 2, 3, 4$; $L = 10$; 16-QAM.

5.4.1 BLER performance

Figures 5.1 to 5.4 present the *BLER* performance of V-LS and V-SQR under several combinations of n_T and n_R. Results for V-BLAST are not shown, since they are identical to those of V-LS. As in chapter 4, simulations were done with $L = 10$ and 16-QAM.

It is concluded that for values of n_R larger than n_T, the *BLER* performance of V-SQR is similar to that of V-BLAST. As n_T and n_R get closer, however, the difference begins to be significant; for $n_T = n_R = 8$, it is more than 2dB. It is unknown whether this is because V-SQR's ordering improves as n_R grows, or because the large number of receive antennas make the algorithm more resilient to errors in the detection order. When lowering the error rate is a priority, V-SQR is a viable alternative as long as $n_R > 1.5n_T$.

5.4.2 Complexity as a function of L

As in chapter 4, the number of operations required by each algorithm to receive one bit of information is called O_b; this number includes all arithmetic as well as memory operations.

Letting $n_T = 4$, the way O_b changes as a function of L has been determined. The contribution to the block setup phase complexity diminishes as L grows, while that of the symbol estimation phase grows. Results are presented in figures 5.5 through

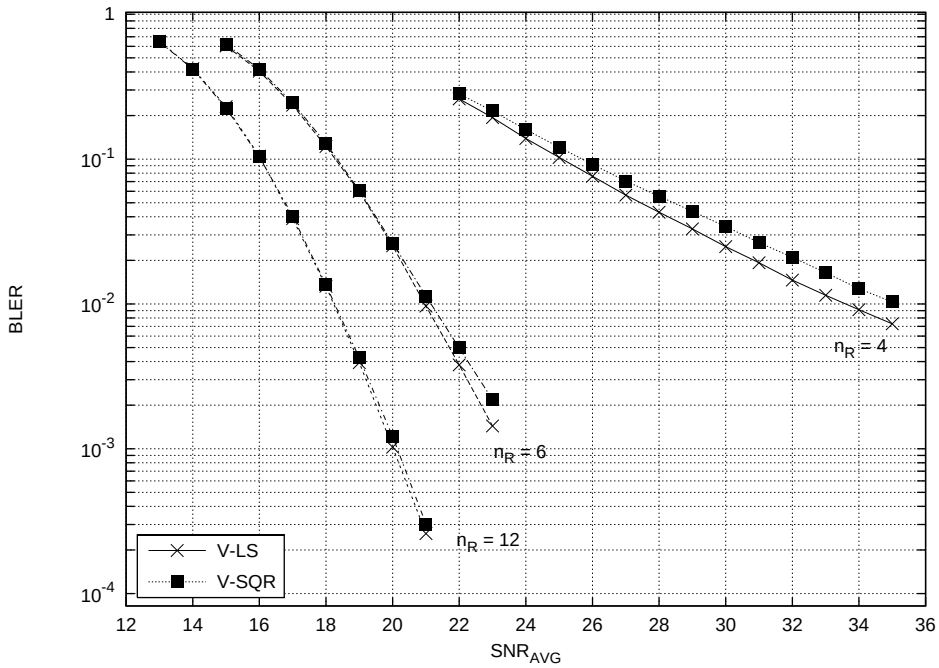

Figure 5.2: $BLER$ as a function of average SNR (dB) for $n_T = 4$; $n_R = 4$, 6, 8; $L = 10$; 16-QAM.

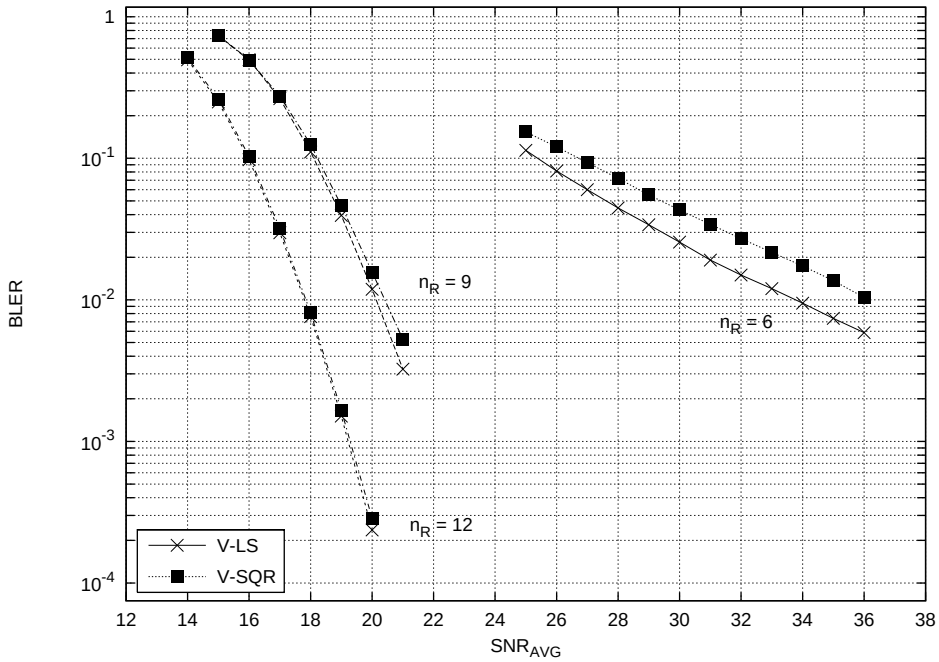

Figure 5.3: $BLER$ as a function of average SNR (dB) for $n_T = 6$; $n_R = 6$, 9, 12; $L = 10$; 16-QAM.

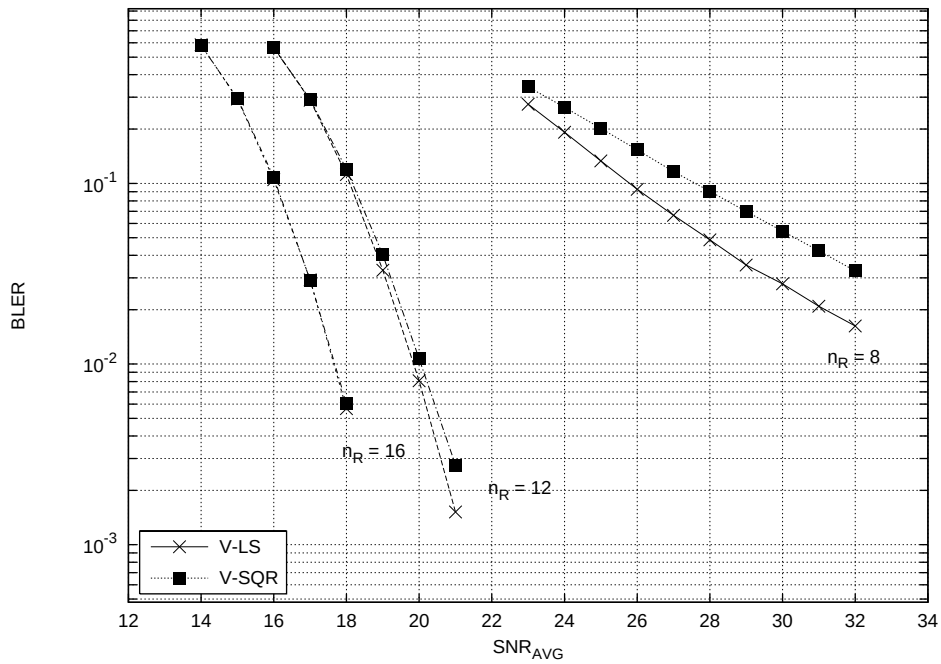

Figure 5.4: $BLER$ as a function of average SNR (dB) for $n_T = 8$; $n_R = 8$, 12, 16; $L = 10$; 16-QAM.

5.7; in each figure, a different value of n_R is used.

The conclusion is that V-SQR's complexity advantage decreases as L increases, with a more pronounced effect for large n_R. The reasons for this are the slight overhead in the symbol decoding phase of V-SQR, which does not exist in V-LS, and the reduced importance of the setup phase as L increases. If error-rate is the priority, and L is relatively large, then V-LS is a viable alternative, especially if n_R is close to n_T.

5.4.3 Complexity as a function of n_R

Again letting $n_T = 4$, it is determined how O_b changes when n_R is increased, for $L = 15$. Results are presented in figure 5.8.

It is clear that both V-LS and V-SQR have significantly lower complexity than V-BLAST. Furthermore, for sufficiently large n_R, V-LS has lower complexity.

The difference in the slope of O_b between the traditional V-BLAST and the other two algorithms in figure 5.8 is worth noting; it is caused by the lessened influence of n_R upon most operations performed by both V-LS and V-SQR.

Both V-LS and V-SQR have their strengths and weaknesses. Table 5.1 summarizes the conditions under which one preferable to the other.

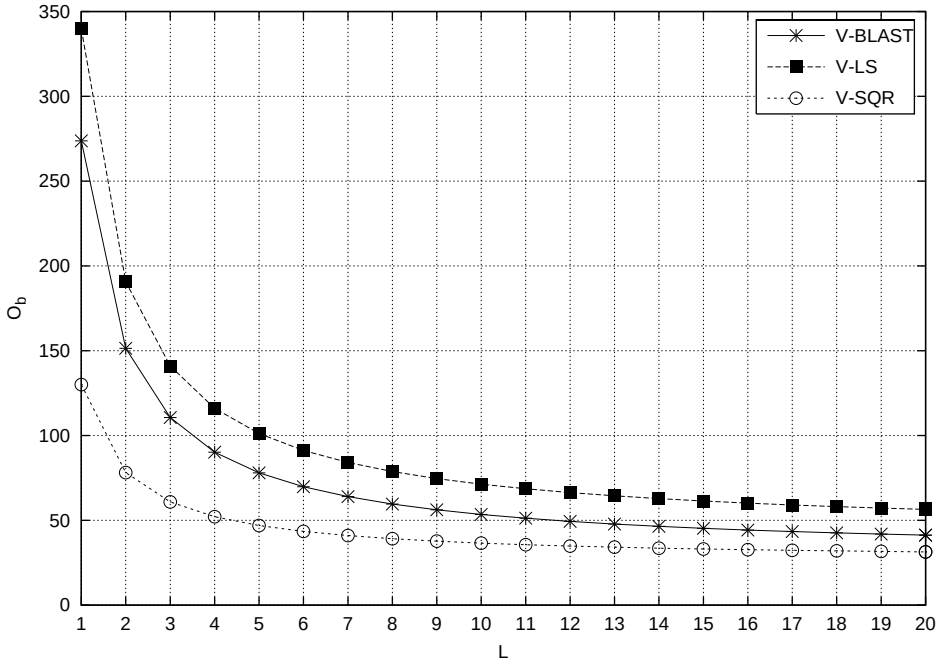

Figure 5.5: O_b as a function of L for $n_T = 4$; $n_R = 4$;

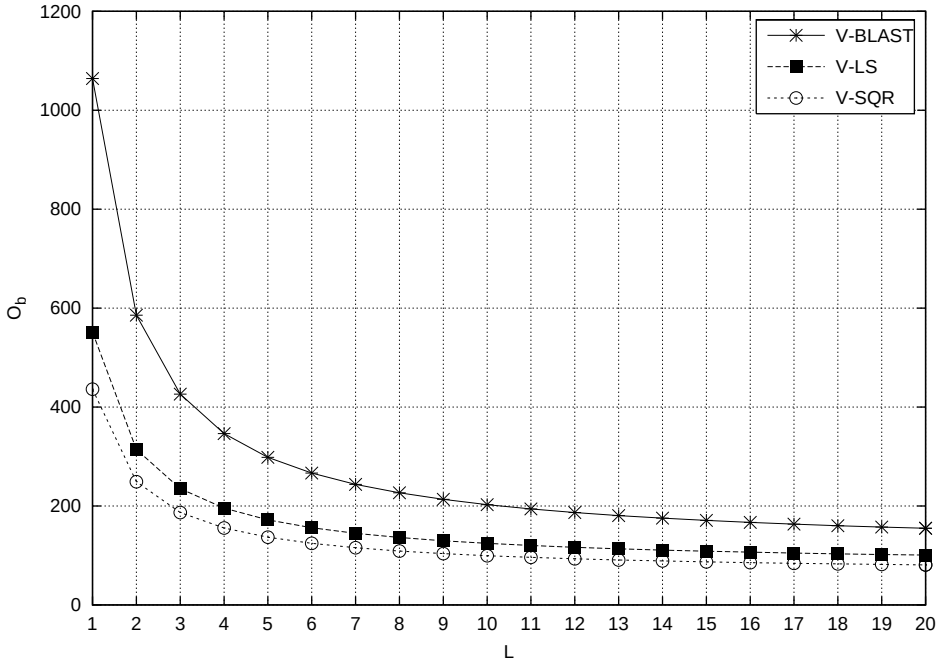

Figure 5.6: O_b as a function of L for $n_T = 4$; $n_R = 16$;

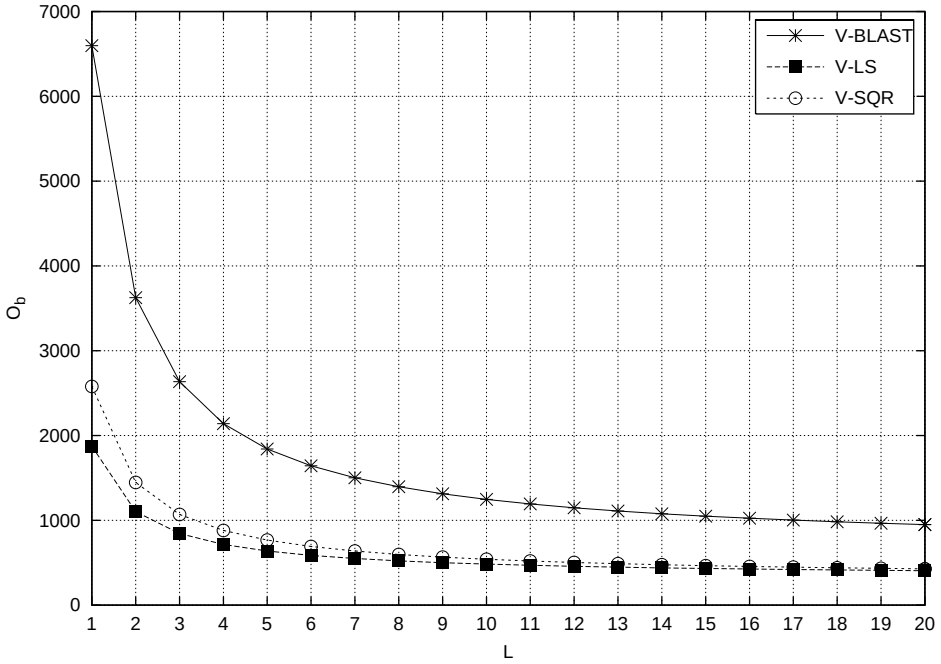

Figure 5.7: O_b as a function of L for $n_T = 4$; $n_R = 100$;

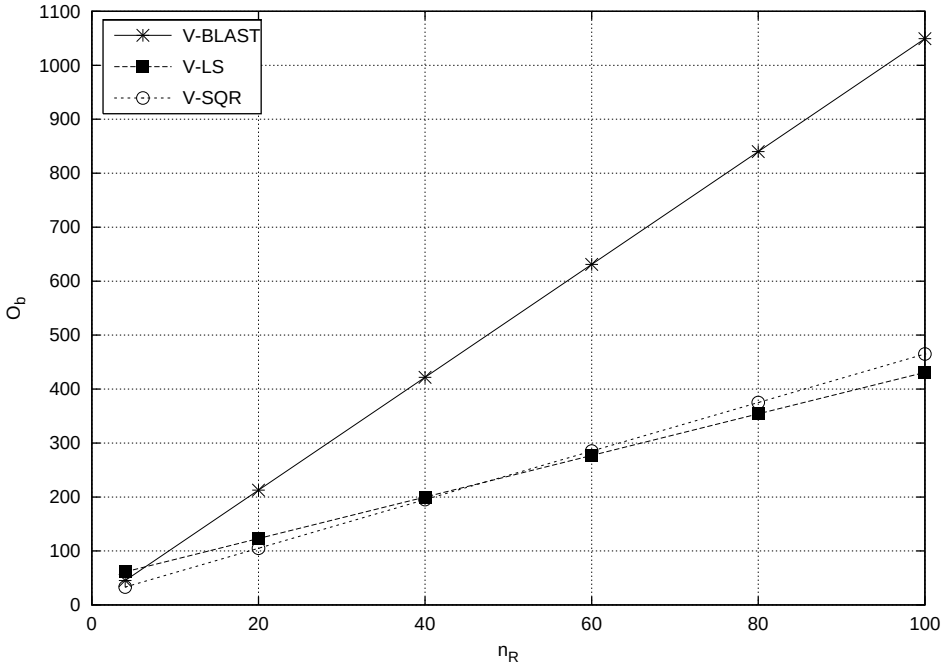

Figure 5.8: O_b as a function of n_R; $n_T = 4$; $L = 15$

n_R/n_T	**Priority** (*error rate* or O_b)	**Choice**
≈ 1	O_b	V-SQR
≈ 1	error rate	V-LS
$>> 1$	O_b	for sufficiently high n_R/n_T and L, best option is V-LS
$>> 1$	error rate	both are equal

Table 5.1: Summary of conditions under which each algorithm is a better option than the other.

5.5 Experimental Results

As in chapter 4, V-LS was executed on a Texas Instruments 6711 digital signal processor (DSP) for different sizes of L, with $n_T = 2$ and n_R equal to 3, 6 and 23. The results confirm the predictions that the complexity per bit diminishes with L and that V-LS is substantially faster than V-BLAST as n_R increases. For $n_R = 3$ there is little advantage in using V-SQR; for $n_R = 6$, however, the complexity is already reduced around 33%, and for $n_R = 23$ the reduction is around 45%.

The results are presented in figures 5.9 to 5.11, where the total number of clock cycles that the DSP requires per information bit received is shown as a function of the L. See chapter 4 for more details. Note that the scales of the ordinates in each figure are quite different.

5.6 Final words

Results on the behavior of each algorithm show that the choice of one algorithm over another is not straightforward; it will depend on the number of antennas in the system, the block length L, and on whether the application at hand requires low error rates or low complexity. Under some circumstances, V-LS, the novel algorithm proposed, is the most attractive.

Furthermore, as has been pointed out in chapter 4, these results are, so to speak, architecture-agnostic; it is expected that they will provide a sound foundation for the task of designing an architecture capable of making the promise of space-time coding a reality.

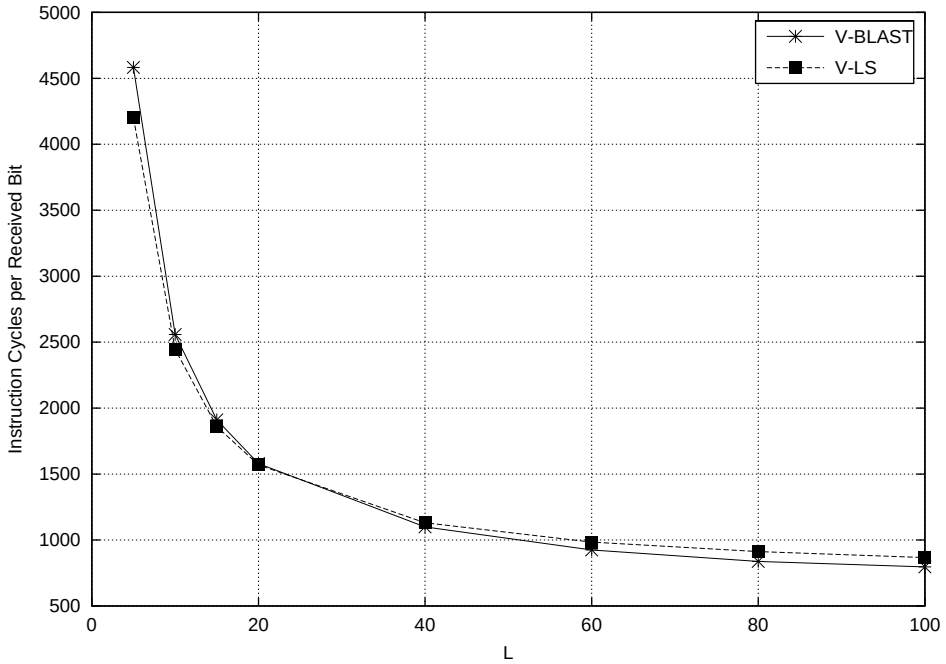

Figure 5.9: Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 3$; 16-QAM

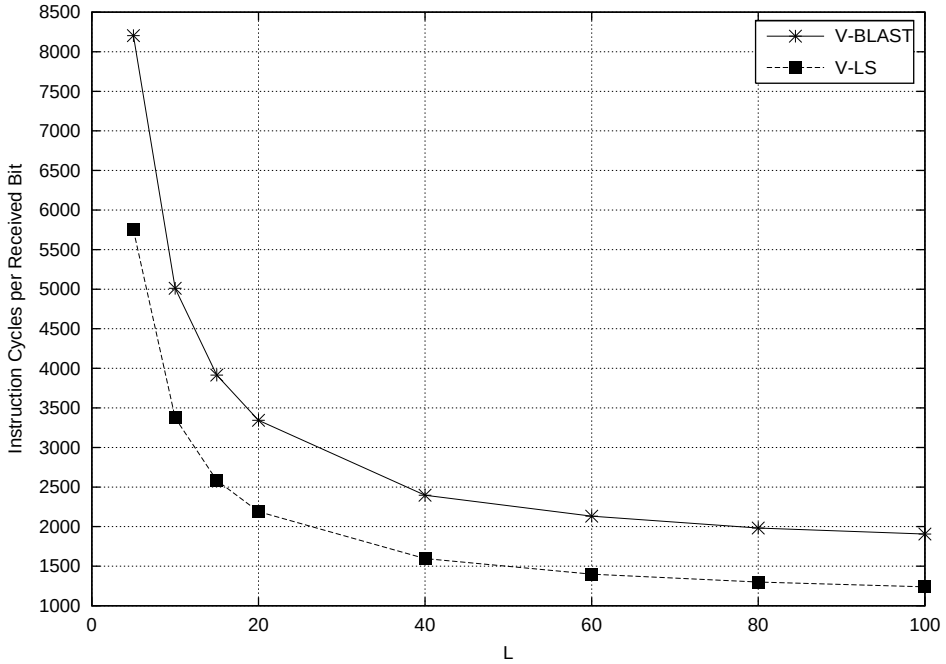

Figure 5.10: Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 6$; 16-QAM

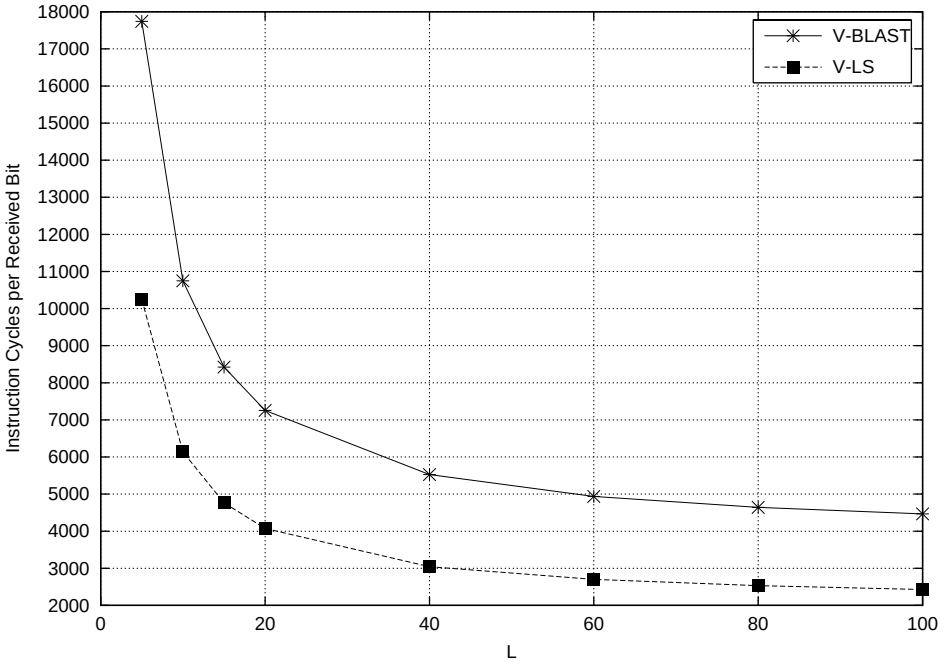

Figure 5.11: Comparison of instruction cycle count per received bit of a TI 6711 DSP running V-BLAST and V-LS with $n_T = 2$, $n_R = 23$; 16-QAM

Chapter 6

Lattice decoding applied to vertical space-time codes

6.1 Introduction

THE ALGORITHMS based on V-BLAST presented in previous chapters are interesting in their own right. The ordering-cancellation-nulling operations have proved to be an effective, simple method to exploit diversity and MIMO capacity under a variety of circumstances.

The real potential of vertical codes, however, has not yet been described. V-BLAST, by design, trades optimality for complexity. How sub-optimal is it really? Further, what is the real gain in complexity compared to optimal algorithms?

These questions are answered in this chapter and the next, for cases when the signal constellation can be arranged in a structure known as a lattice. Very fast algorithms exist that can be used to optimally receive vertical codes in such cases.

Geometrically, a *lattice* is a regular, periodic, infinite arrangement of points in an n-dimensional space. Figure 6.1 shows a lattice in two dimensions, a point $\mathbf{r} \in \mathbb{R}^2$ and the lattice point closest to $\mathbf{r}$.

More formally, let $\mathbf{G}$ be a matrix of real elements, with n rows and m columns, whose rows are linearly independent (which implies $n \leq m$). The lattice generated by $\mathbf{G}$ is defined as the set of vectors

$$\Lambda(\mathbf{G}) = \{\mathbf{uG} \colon \mathbf{u} \in \mathbb{Z}^n\}. \tag{6.1}$$

Matrix $\mathbf{G}$ is called the generator matrix of Λ, and its rows are called the basis vectors of the lattice. Note that the elements of Λ have dimensions $1 \times m$, but the dimension of the lattice is n. Given a lattice Λ and a vector $\mathbf{x} \in \mathbb{R}^m$, the problem of finding a vector $\hat{\mathbf{c}} \in \Lambda$ such that

$$\| \mathbf{x} - \hat{\mathbf{c}} \| \quad \leq \quad \| \mathbf{x} - \mathbf{c} \| \qquad \text{for all } \mathbf{c} \in \Lambda,$$

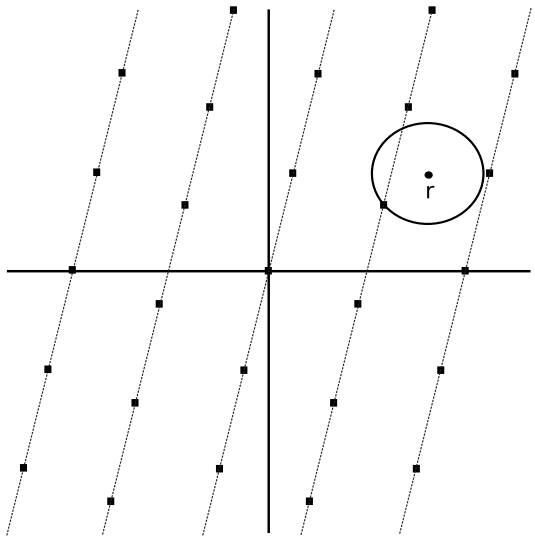

Figure 6.1: A lattice in two dimensions. The solid points represent the lattice points. The straight lines emphasize the periodic arrangement of the points. Also shown is a point $\mathbf{r} \in \mathbb{R}^2$, and the closest lattice point.

where $\|\cdot\|$ is the ℓ^2-norm, is called the *closest point problem*. This problem (and other, closely related ones) has numerous applications in mathematics and engineering. In the context of channel coding, the closest point problem is related to maximum-likelihood decoding. When the codewords can be arranged in a lattice, *lattice decoders* (algorithms optimized to exploit its regular structure), are faster than general ML decoders.

Because of their wide application and importance, there is a large body of knowledge about lattice decoding. Efficient algorithms have been found to implement it, and research in this field continues to be active. Given its close relationship to maximum-likelihood receivers, it promises considerably better error rates than V-BLAST. It is for these reasons that lattice decoding is investigated as an alternative for the reception of vertical layered space time codes.

In this chapter, a method is proposed to adapt lattice decoding to MIMO systems. A lattice decoding algorithm (based on ideas presented in [34]) is proposed for use with QAM constellations. Finally, a limitation of lattice decoding, present under certain conditions, is tackled in what is believed to be a novel way.

Chapter 7 presents results on the complexity and error rates of lattice decoding compared to V-BLAST.

6.2 The maximum-likelihood criterion

A brief recount of the maximum-likelihood criterion as applied to *additive, white Gaussian noise* (*AWGN*) channels is presented. Consider a communications system in which the transmitter sends vectors $\mathbf{a}_m = (a_{m,1}, a_{m,2}, \ldots, a_{m,n_T})^T$. All elements

$a_{m,i}$ belong to a signal constellation S, which has 2^b elements, where b is the number of information bits transmitted per signal $a_{m,i}$. Let the signal constellation S' be the set of all possible values of $\mathbf{a}_m$; S' has 2^{bn_T} elements.

Let the AWGN communications channel corrupt the transmitted vectors by the addition of noise, so that the received vector $\mathbf{r}$ is given by

$$\mathbf{r} = \mathbf{a}_m + \mathbf{n},$$

where $\mathbf{n}$ is a vector whose elements are taken from a Gaussian distribution with mean zero and power $N_0/2$.

After reception of each vector, the receiver has the task of estimating which of all possible vectors $\mathbf{a}_m$ was transmitted using its knowledge of $\mathbf{r}$. The *maximum-likelihood* (ML) receiver chooses its estimate $\hat{\mathbf{a}}$ as the vector element of S' that is closest to $\mathbf{r}$, that is,

$$\hat{\mathbf{a}} = \underset{\mathbf{a}_m \in S'}{\operatorname{argmin}} \parallel \mathbf{a}_m - \mathbf{r} \parallel^2 . \tag{6.2}$$

Equation 6.2 is known as the *ML criterion*. Assuming all vectors $\mathbf{a}_m$ are transmitted with equal probability, then the ML receiver is optimum in the sense that it minimizes the probability of error. If S' is a lattice, then lattice decoding and the ML criterion are equivalent.

6.2.1 The ML criterion in MIMO systems

The maximum-likelihood criterion is usually applied to AWGN channels, where its optimality has been proved. The applicability of this criterion to MIMO fading channels is not readily apparent. To verify that it still applies, note that the ML criterion is independent of the shape of the signal constellation in use. Let S_M be the set of signals

$$S_M = \{\mathbf{H}\mathbf{a}_m : \mathbf{a}_m \in S'\},$$

and let the signals in S_M be denoted by $\mathbf{s}_m$, $m = 1, \ldots, 2^{bn_T}$. A communication system that uses S_M in an AWGN, non-fading environment is described by equation (6.3). Such a system is theoretically indistinguishable from one that uses S' in a MIMO, Rayleigh-fading environment with channel matrix $\mathbf{H}$, so the ML criterion remains optimal (see also [1, pp. 2663-2664]).

$$\mathbf{r} = \mathbf{s}_m + \mathbf{v}. \tag{6.3}$$

6.3 Applying lattice decoding to V-BLAST

Finding the point in S_M closest to $\mathbf{r}$ can be a complex task: even though finding the distance between two points is simple, calculating and comparing many distances

one by one can be an unfeasibly lengthy task. In many applications, suboptimum but fast algorithms are often preferred to ML reception.

Lattice decoding exploits the structure of the lattice to accelerate the search for the closest point, making it several times faster than a general ML receiver. In order to use lattice decoding in a MIMO system, the following problems must be solved:

- since S_M is not, in general, a lattice, a suitable lattice representation must be found.
- a mapping between the lattice representation of S_M and S must be found, since ultimately the receiver has to produce estimates of $\mathbf{a}_m \in S'$.

6.3.1 Finding a lattice representation of S_M

The aim of this section is to represent the system in equation (6.3) in a way amenable to lattice decoding.

First, a real system model must be found. Recall from section 2.2.3 that MIMO systems can be expressed in real form; in equation (2.2), the system model is given by

$$\tilde{\mathbf{r}} = \tilde{\mathbf{H}}\tilde{\mathbf{a}}_{\mathbf{m}} + \tilde{\mathbf{n}},$$

where all terms are real. In this model the transmitted vectors are formed by a matrix-vector product; the lattice definition, however, requires its elements to be formed by a vector-matrix product. This is easily solved by redefining the signal constellation S_M as

$$S_M = \{\tilde{\mathbf{a}}_m^\dagger \tilde{\mathbf{H}}^\dagger : \mathbf{a}_m \in S'\}.$$

Note that now the elements of S_M are row vectors, and that $\tilde{\mathbf{H}}^\dagger$ meets the requirements for a generator matrix (described above).

It should be noted that redefining the system in this way produces a doubling of the dimensions of $\mathbf{H}$ and $\mathbf{r}$. This necessarily has an effect on the complexity of the decoder.

The definition of a lattice requires that the vectors that multiply the generator matrix have integer elements. A QAM constellation is defined as a set of 2-dimensional vectors whose elements are members of the sequence

$$\{\pm e_1, \pm 3e_1, \ldots, \pm(2n-1)e_1\},$$

for n an appropriate integer[1]. The factor e_1 is chosen so that $\mathcal{E}[\|\mathbf{a}_m\|^2] = n_T$; in other words, each element $a_{m,i}$ of $\mathbf{a}_m$ has energy one.

This means that e_1 can be factored out of $\mathbf{a}_m$, leaving an integer vector. Consider the translated signal constellation S_T:

[1]For instance, for 16-QAM, $n = 2$; for 64-QAM, $n = 4$

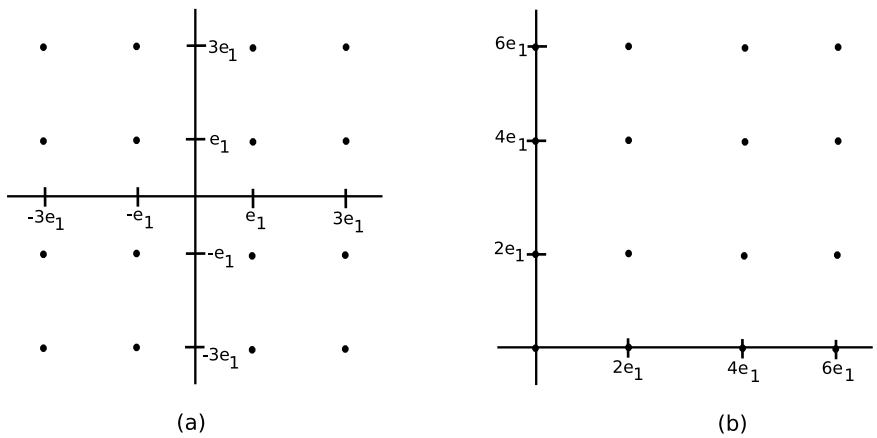

Figure 6.2: (a) A 16-QAM constellation. (b) The same constellation translated by $(2n-1)e_1$.

$$S_T = \{\mathbf{s}_m + \mathbf{t}\tilde{\mathbf{H}}^\dagger : \mathbf{s}_m \in S_M\},$$

where the *translation vector* $\mathbf{t}$ is such that all its elements are equal to $(2n-1)e_1$. To illustrate the effect of translating a constellation, figure 6.2 shows a 16-QAM constellation, and the same constellation translated by $(2n-1)e_1$.

Now, consider the lattice $\Lambda_S = \Lambda(2e_1\tilde{\mathbf{H}}^\dagger)$. All signals in S_T are also in Λ_S, which can then be used in lattice decoding. S_T is a subset of Λ_S, and is frequently called a *carving* of Λ_S. Note that the carving is contained in a contiguous region within Λ; within this region, there are no points that belong to Λ but not to S_T. If this condition were not met, the lattice decoder could return points outside S_T even in cases when $\mathbf{x}$ is very close to a constellation point.

Finally, since the lattice decoder operates on a translated lattice, it is also necessary to translate the received vector by the same amount.

In summary, a receiver that uses lattice decoding must build a generator matrix $\mathbf{G}$ from $\mathbf{H}$, and a vector $\mathbf{x}$ from $\mathbf{r}$, such that:

$$\mathbf{G} = 2e_1 \begin{bmatrix} \Re(\mathbf{H}) & \Im(\mathbf{H}) \\ -\Im(\mathbf{H}) & \Re(\mathbf{H}) \end{bmatrix}^T,$$

$$\mathbf{x} = \tilde{\mathbf{r}} + (2n-1)e_1\mathbf{G}.$$

Now the lattice decoder can operate on $\mathbf{G}$ and $\mathbf{x}$, to find a vector $\hat{\mathbf{u}}$ such that $\hat{\mathbf{u}}\mathbf{G}$ is the closest lattice point to $\mathbf{x}$.

6.3.2 Mapping $\hat{\mathbf{u}}$ to S

The vector produced by the lattice decoder, $\hat{\mathbf{u}}$, belongs to $\mathbb{Z}^{2n_T}$; it is an estimate of the vector $\mathbf{u}$ used in (6.1) to generate the lattice points. The mapping from $\hat{\mathbf{u}}$ to S is straightforward; it suffices to multiply by $2e_1$ and subtract $\mathbf{t}$.

Alternatively, the receiver could demodulate $\hat{\mathbf{u}}$ directly without mapping it back to S.

One important consequence of using a lattice decoder instead of a general ML algorithm is that the vector $\hat{\mathbf{u}}$ is an element of the lattice Λ_S, but not necessarily of S_T. Especially for low SNR, there is a high probability that the received point will lie outside the constellation, so a lattice point that does not belong to it will be returned as estimate.

Several possible remedies for this situation have been suggested [34]. The simpler option is to declare an erasure, at a cost in the probability of error. Another course of action is to identify the occurrences of this situation, and then expand the algorithm's search region until a constellation point is found. This has the obvious disadvantage of greatly increasing the complexity of the receiver.

A third method is to project the input vector into the boundary of the constellation before searching for the closest point. Besides being not quite optimal, this procedure again incurs a potentially high increase in complexity.

In this thesis, what is believed to be a novel heuristic to solve this problem is proposed. Although suboptimal, it has the advantage of having a negligible effect on the algorithm's complexity while having a lower error rate than simply declaring an erasure (or alternatively, choosing as estimate a fixed, predetermined constellation point).

The proposed method consists of quantizing every element of $\hat{\mathbf{u}}$ that is outside the allowed range to the closest allowed value. For example, if S is 16-QAM, then only those $\hat{\mathbf{u}}$ whose elements are 0, 1, 2 or 3 correspond to constellation points. All those elements of $\hat{\mathbf{u}}$ that are less than zero are quantized to zero, and those larger than 3 are quantized to 3. All valid values are left untouched.

Simulation results that compare this method to maximum-likelihood decoding are presented in chapter 7.

6.4 A lattice decoding algorithm for V-BLAST

The lattice decoding algorithm used here is based on the algorithm called ClosestPoint first proposed in [34]. This lattice decoder is believed to be the fastest available for general lattices. The algorithm follows.

The ClosestPoint algorithm is a front-end to algorithm Decode, which finds the closest point to $\mathbf{x_3}$ in $\Lambda(\mathbf{H_3}^{-1})$. Steps 1-3 perform several transformations to $\mathbf{G}$ and $\mathbf{x}$ that have the effect of improving *Decode*'s speed.

Decode's complexity depends on the length of the rows of $\mathbf{G}$. Step 1 is a basis reduction, which reduces these lengths. The most efficient reductions, like LLL or Korkine-Zolotareff, may reduce the execution time of Decode dramatically but are very complex themselves. Thus, they are more beneficial when many vectors are to be decoded in the same lattice. Note that $\mathbf{W}$ may also be the identity matrix, in which case $\mathbf{G}$ is not reduced at all. The next chapter provides more details on this

Algorithm 5 ClosestPoint

Input: a $2n_T \times 2n_R$ generator matrix $\mathbf{G}$, and a $1 \times 2n_R$ vector $\mathbf{x}$ to decode in $\Lambda(\mathbf{G})$
Output: the lattice point $\hat{\mathbf{x}} \in \Lambda(\mathbf{G})$ that is closest to $\mathbf{x}$.

1: Let $\mathbf{G_2} = \mathbf{WG}$, where $\mathbf{W}$ is an $2n_T \times 2n_T$ matrix with integer entries and determinant ± 1.
2: Compute a $2n_R \times 2n_T$ matrix $\mathbf{Q}$ with orthonormal columns and a $2n_T \times 2n_T$ lower-triangular matrix $\mathbf{G_3}$ with positive diagonal elements, such that $\mathbf{G_2} = \mathbf{G_3Q}$.
3: Let $\mathbf{x_3} = \mathbf{xQ}^T$.
4: Let $\mathbf{H_3} = \mathbf{G_3}^{-1}$.
5: Let $\mathbf{u_3} = \text{Decode}(\mathbf{H_3}, \mathbf{x_3})$
6: Return $\hat{\mathbf{x}} = \mathbf{u_3G_2}$.

subject and examines the performance benefit obtained by using the LLL reduction compared with no reduction.

Steps 2 and 3 are similar to what was done in the V-LS algorithm in chapter 5; the problem is reduced to size $2n_T \times 2n_T$. One way to obtain the matrices $\mathbf{Q}$ and $\mathbf{G_3}$ is to compute the thin QR decomposition of $\mathbf{G_2}^T$, which gives $\mathbf{Q}^T$ and $\mathbf{G_3}^T$.

The ClosestPoint algorithm has been modified to receive vertical layered space-time codes; the new algorithm is called V-CP and is written below.

As in previous chapters, the algorithm V-CP has been written to operate on a block of L received vectors, all of which are transmitted through the same channel $\mathbf{H}$. As was the case for V-BLAST, these algorithms can be divided in two phases: a setup phase that is carried out once per block, and a symbol estimation phase that is repeated L times.

Step 1 computes a matrix $\mathbf{G}$, and steps 2 and 3 find the translation vector $\mathbf{t}$ as explained in section 6.3.1. Steps 4 to 9 deal with the basis reduction. If the boolean variable R is set to **true**, V-CP will perform the LLL reduction on $\mathbf{G}$; otherwise no reduction is carried out. Matrix $\mathbf{W}$ is needed later in the decoding process.

In step 14, the received vector is translated using $\mathbf{t}$, and in step 15 the closest point $\hat{\mathbf{x}}$ is found. The heuristic to map points $\hat{\mathbf{x}}$ that do not belong to the signal constellation, introduced in 6.3.2, is executed in steps 16 to 22.

Algorithm 6 V-CP

Input: an $n_R \times n_T$ matrix $\mathbf{H}$, a set of L $n_R \times 1$ vectors $\mathbf{r}_i$, $i = 1, \ldots, L$, and a signal constellation S. A boolean variable R determines whether to perform the LLL reduction of the lattice basis or not.

Output: a set of L $n_T \times 1$ vectors $\hat{\mathbf{u}}_i \in \mathbb{Z}^{2n_T}$ from which the original information bits can be estimated.

```
Let G = 2e_1 H̃^T.
Let t_s be a 1 × 2n_T vector with elements equal to (2n − 1)e_1.
Let t = t_s H̃.
if R is true then
    Let G_2 = LLL(G)
    Let W = G_2 G^{-1}
else
    Let G_2 = G
    Let W be the identity matrix
end if
Compute a 2n_R×2n_T matrix Q with orthonormal columns and a 2n_T×2n_T lower-
    triangular matrix G_3 with positive diagonal elements, such that G_2 = G_3 Q.
Let H_3 = G_3^{-1}.
for i = 1 to L do
    Let x = (r̃_i + t)Q^T.
    Let x̂ = Decode(H_3, x) · W
    for j = 1 to 2n_T do
        if x̂_j > (2n − 1) then
            x̂_j = (2n − 1)
        else if x̂_j < 0 then
            x̂_j = 0
        end if
    end for
    û_i = x̂
end for
```

Chapter 7

Lattice decoding in MIMO systems: practical considerations

7.1 Introduction

In this chapter, results are presented that allow a comparison of the complexity and performance of three receiver algorithms studied in previous chapters:

- V-LS
- V-CP without basis reduction
- V-CP with basis reduction

Algorithm V-LS was introduced in chapter 5; it is a modification of the original V-BLAST that uses least-square techniques to lower its complexity without affecting the probability of error. V-CP is an adaptation of lattice decoding, introduced in chapter 6, to the reception of vertical space-time codes. Some important aspects of V-CP's behavior are examined here.

The bit and block error rates of V-LS and V-CP are compared in order to determine how suboptimal V-LS really is. Systems with $n_R \gg n_T$ are of particular interest, since V-BLAST has proven to be adept at exploiting space diversity in this situation. There is little doubt that, in cases where the highest priority is minimizing the probability of error, V-CP will prove superior to V-LS. The complexity estimates presented here point to the material cost of such a decision. The advantage of V-LS over V-CP in this sense is not always clear-cut, especially since the complexity of V-CP depends on the average SNR.

V-CP can optionally use basis reduction of the lattice generator matrix in an attempt to lower its complexity. There are two common kinds of reduction: the *LLL* reduction (named after Lenstra, Lenstra and Lovász) [35] and the *Korkine-Zolotareff* (*KZ*) reduction [36]. The KZ reduction finds a basis whose vectors are shorter than the corresponding LLL reduction; however, for the LLL reduction there

exists an algorithm that operates in polynomial time [37, pp. 83-89]. For this reason, LLL has been chosen to perform the reduction in V-CP.

It is important to remark that in [34], the complexity results presented *do not* include the complexity of the basis reduction. This surprising fact means that, even though it is proved that *Decode*[1] is faster when the generator matrix is reduced, it has not been proved that the *total* complexity (reduction plus search) is actually reduced. Results are presented regarding this matter.

The LLL reduction is also supposed to increase the numerical stability of V-CP, in cases where the channel matrix is ill-conditioned. This claim is not examined directly, but some observations made during simulations are presented.

In addition, the method used in V-CP to deal with points outside the signal constellation is compared to maximum-likelihood decoding. It is also compared to a simpler method that maps any point that lies outside the constellation to a fixed point inside it.

7.2 Error rate comparison

In this section the error rates of V-LS and V-CP are compared. The LLL basis reduction has no effect on the probability of error of V-CP, so it will be ignored. Figures 7.1 to 7.4 show a comparison of block error rates, and figures 7.5 to 7.8 compare bit error rates. In *BLER* results, the block length $L = 10$. As in other chapters, 16-QAM has been used throughout.

The systems being compared are the same as in previous chapters: $n_T = 2$, 4, 6, and 8, with $n_R = n_T$, $1.5n_T$, and $2n_T$.

Some interesting observations can be drawn from these figures. One is that for $n_R = 2n_T$ and $BLER = 10^{-2}$, the difference between V-LS and V-CP ranges between a small fraction of one decibel and roughly half a decibel. This means that V-LS is surprisingly good at exploiting antenna diversity.

For $n_T \approx n_R$, however, the situation is different. Lattice decoding provides an error rate that is clearly superior to V-LS. At $BLER = 10^{-2}$ and $n_T = n_R = 2$, the difference between them is 4dB; as the number of antennas grows, so does the difference, reaching 10dB for an (8,8) system.

Recall from chapter 3 that, when using V-BLAST in a symmetric (n,n) system, the first estimated symbol is not benefited by any diversity, in addition to being affected by the interference from all the other symbols. This explains the large difference between, for instance, a (2,2) and a (2,3) system: in the latter, every symbol benefits from at least some diversity.

On the other hand, V-CP performs a joint estimation of all symbols; in a sense, this means that the system's diversity is the same for the entire received vector. Compare the slope of the *BLER* curves for (n,n) systems: when using V-LS, the curve has all the signs of a receiver unable to exploit diversity; it is roughly a straight

[1]The algorithm underlying V-CP; see chapter 6.

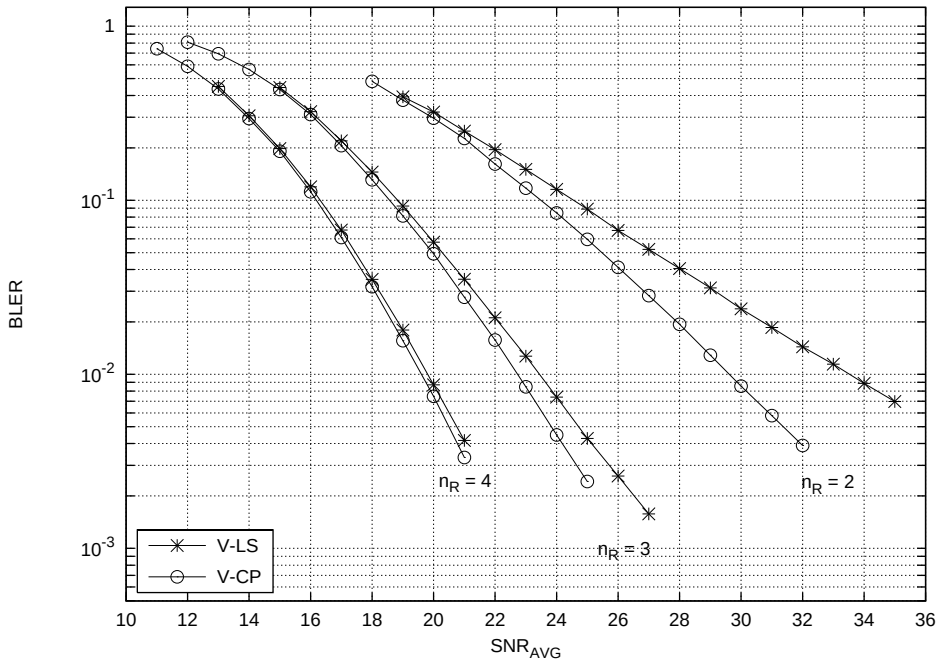

Figure 7.1: *BLER* comparison between V-LS and V-CP; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM.

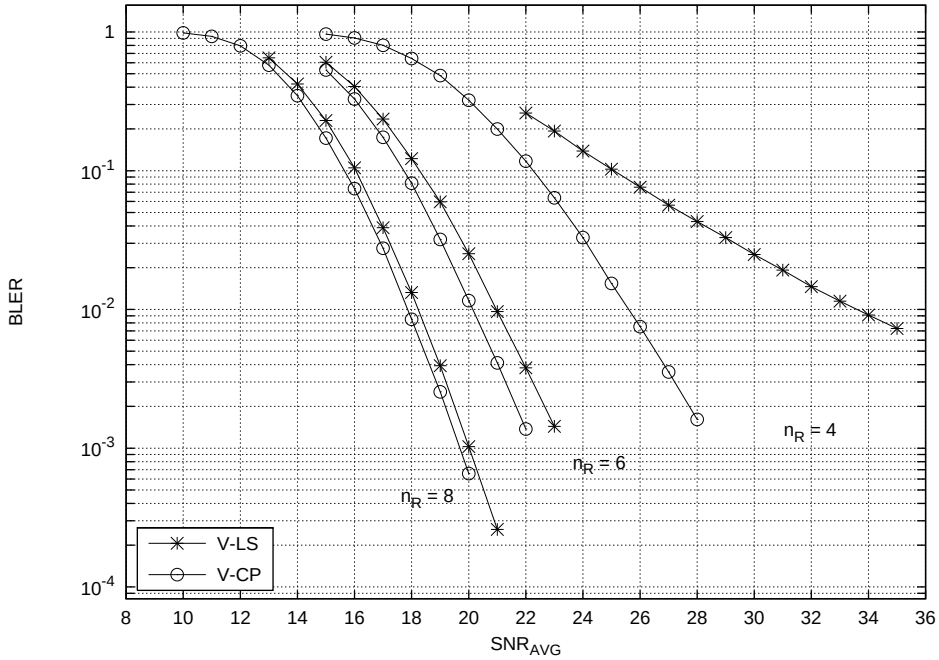

Figure 7.2: *BLER* comparison between V-LS and V-CP; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM.

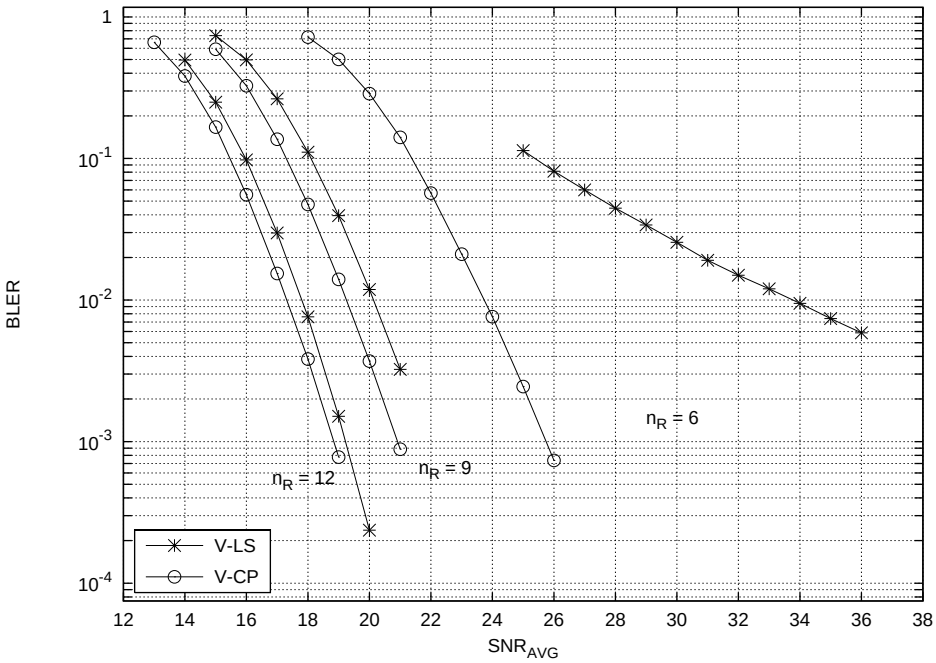

Figure 7.3: *BLER* comparison between V-LS and V-CP; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM.

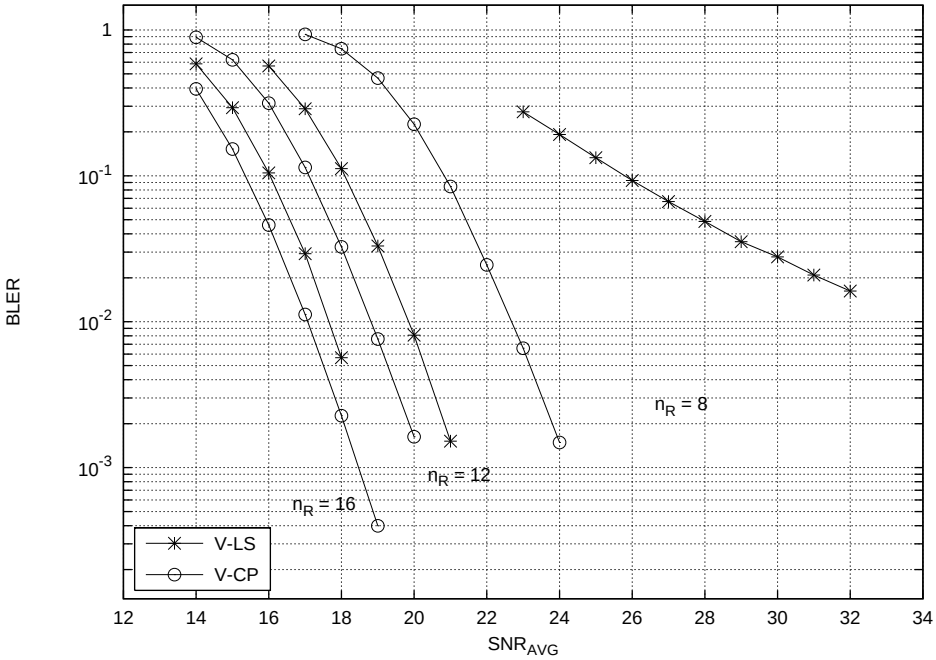

Figure 7.4: *BLER* comparison between V-LS and V-CP; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM.

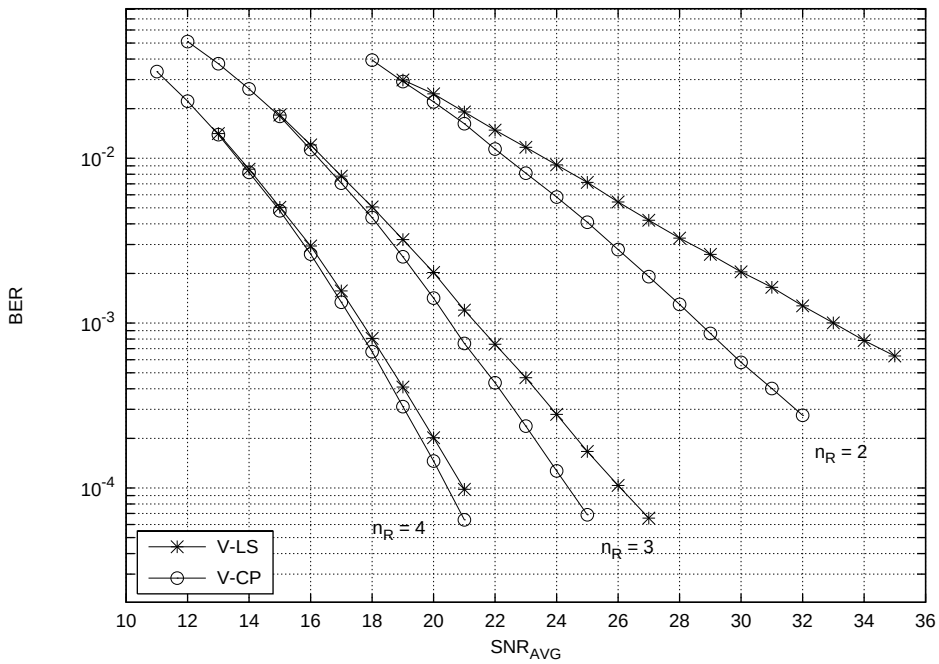

Figure 7.5: *BER* comparison between V-LS and V-CP; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM.

line. In contrast, that of the V-CP increases with average SNR, indicating a receiver that is able to exploit diversity.

Results for *BER* are presented in figures 7.5 through 7.8. The same behavior exhibited by *BLER* is seen here. For $n_R = 2n_T$, the gain of V-CP with respect to V-LS is less than one dB.

It is worth noting that, as n_T increases, the required average SNR to achieve a given *BER* is reduced. This means there is a double benefit to increasing the number of antennas: both a higher spectral efficiency and a reduced probability of error for the same average SNR.

7.3 Complexity comparisons

As in chapters 4 and 5, the algorithms' complexity is represented by O_b, the total number of operations performed per received information bit. Figures 7.9 through 7.12 present a comparison between the complexity of V-LS and V-CP, with no basis reduction performed, for $L = 10$ and 16-QAM.

O_b is presented as a function of the average SNR, since, as opposed to V-BLAST, the complexity of V-CP is not constant but depends on the noise power. The reason is that for large noise values, there is a higher probability that the received vector will be far from any lattice point, and the algorithm has to consider more candidate vectors.

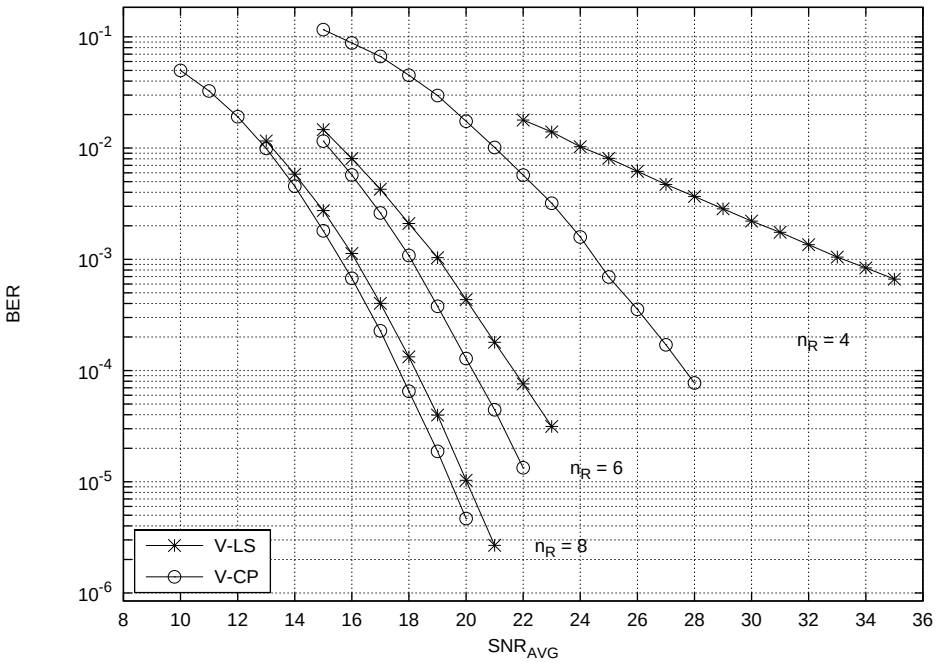

Figure 7.6: *BER* comparison between V-LS and V-CP; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM.

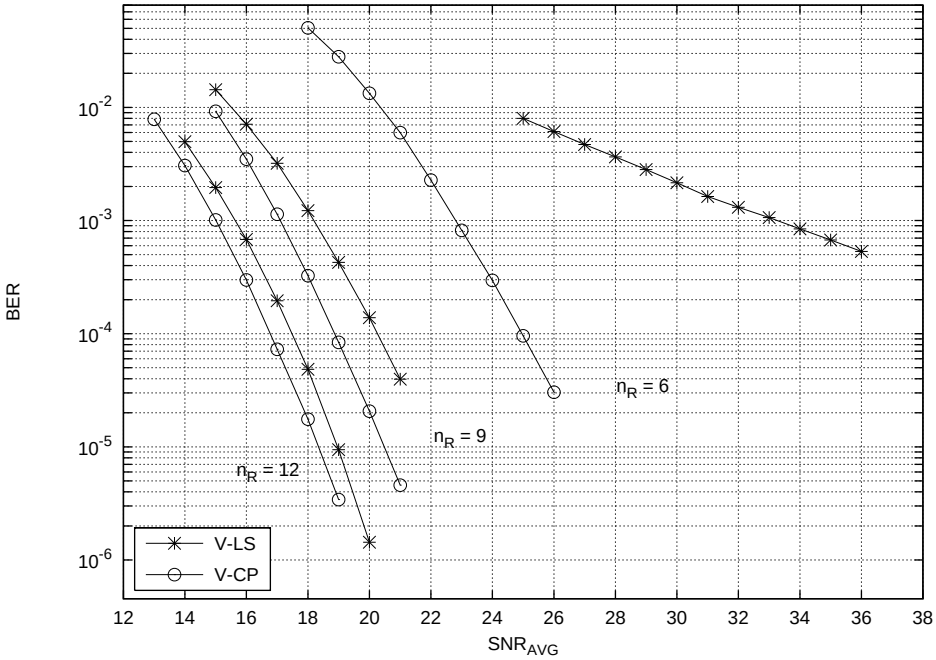

Figure 7.7: *BER* comparison between V-LS and V-CP; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM.

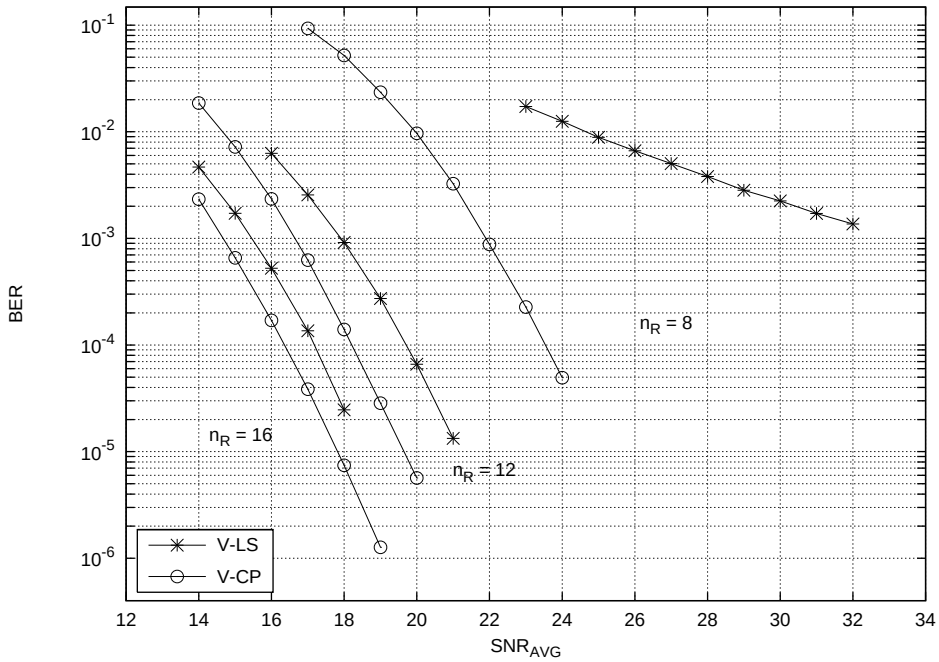

Figure 7.8: *BER* comparison between V-LS and V-CP; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM.

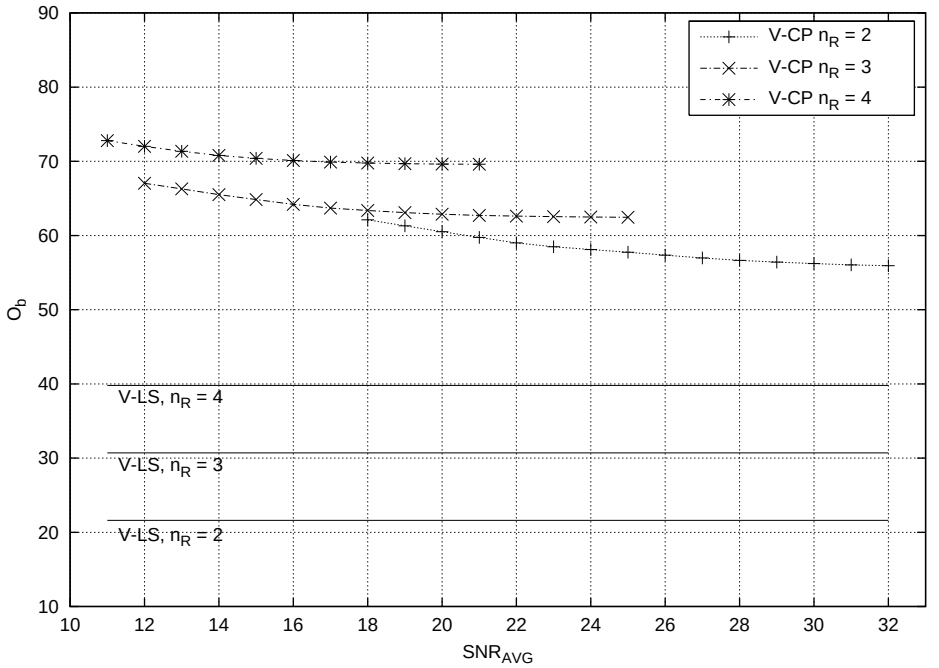

Figure 7.9: O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM

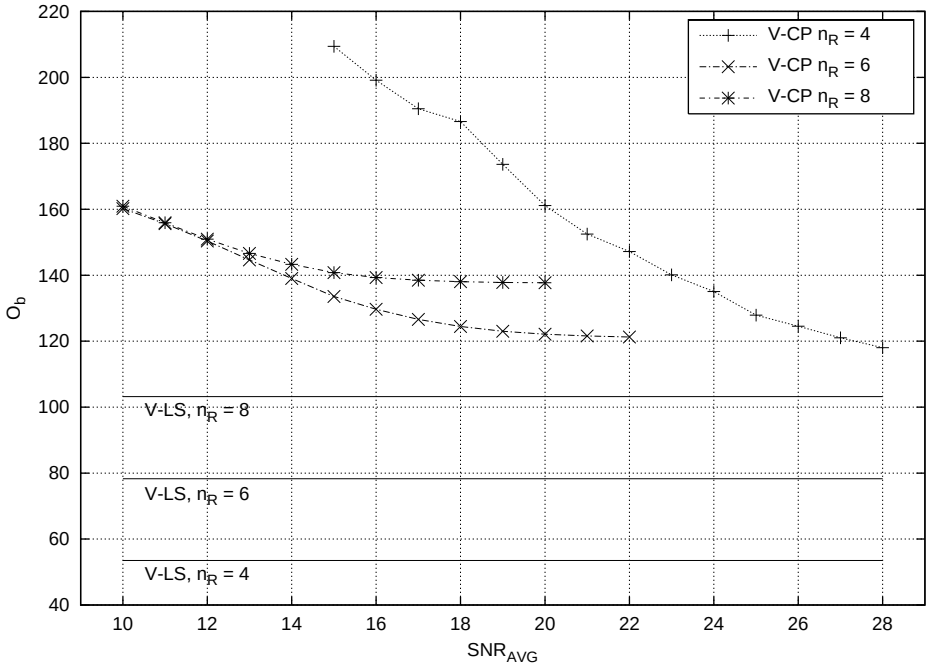

Figure 7.10: O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM

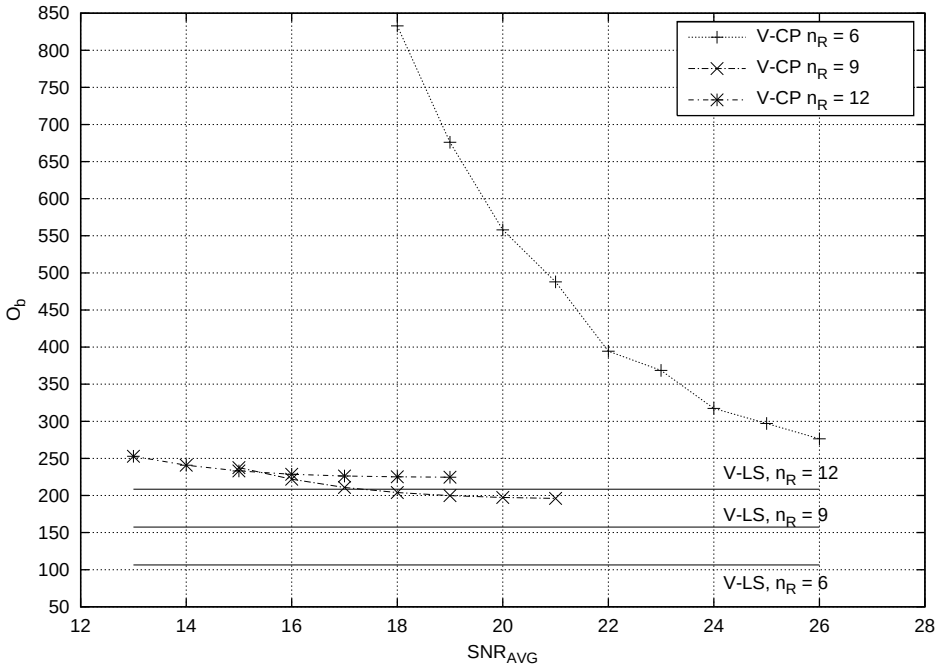

Figure 7.11: O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM

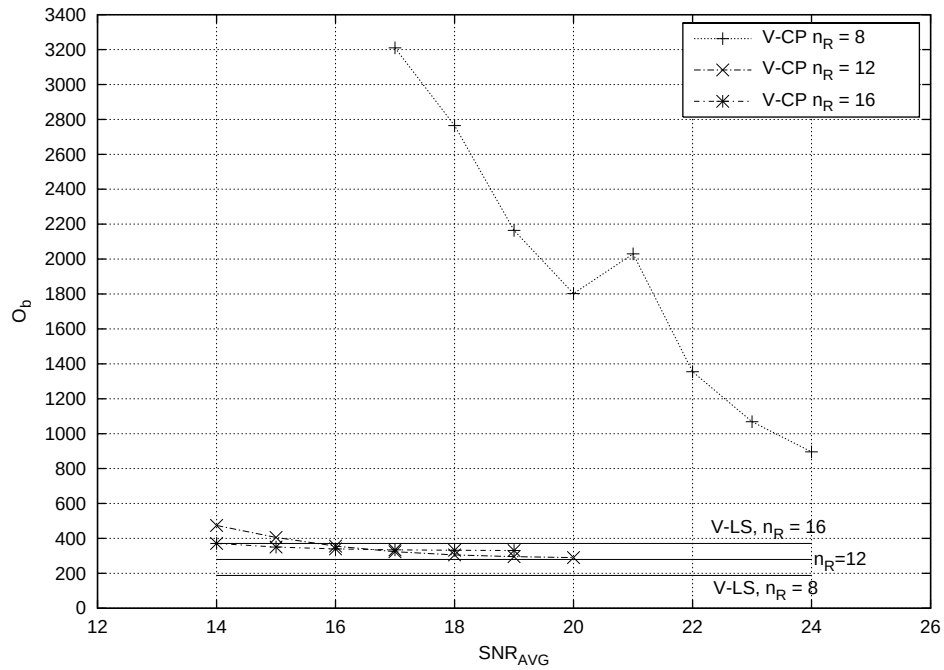

Figure 7.12: O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM

There are many interesting conclusions that can be drawn from the results presented in these figures.

The first, and perhaps the most striking, observation is that the complexity of V-CP for *smaller* systems can be significantly *higher* than for larger ones. The complexity curves show that larger systems have higher complexity only after the average SNR has surpassed a certain threshold value.

This effect can be clearly seen in figure 7.10, in which $n_T = 4$. At 10dB or less, the most complex system is (4,4), followed by (4,6) (for which $O_b = 161$), and by (4,8) ($O_b = 160$). When the average SNR reaches 15dB, the (4,8) system becomes more complex than the (4,6) one, but the (4,4) system is still more complex than both. It is only after the average SNR reaches approximately 27dB that the intuitive order is restored: the larger system becomes more complex. This phenomenon appears in all systems examined but becomes stronger as the number of antennas grows.

Only a hypothesis is proposed here on the possible causes of such behavior. The search size (that is, the number of points that V-CP must examine in order to find the closest one) increases as the noise power increases, and decreases as the dimension of the lattice grows. After the noise power reaches a certain point, however, the search size is dominated by the lattice dimension.

The reason for the search size growing when the noise power increases is that the received point will, with high probability, be found surrounded by many lattice points, none of them particularly close to it. Thus, the algorithm needs to examine

many points. This effect is countered by the dimension of the lattice; as the dimension grows, the volume occupied by any given number of lattice points increases, effectively separating the points, and providing a measure of noise immunity.

When the noise power is low, however, the received point will with high probability be found close to a lattice point, which will be quickly determined by any efficient algorithm. The noise plays a much smaller role in this case and the dominant factor in the search size is the lattice dimension.

The enormous complexity of V-CP for symmetrical systems is also worthy of note. Figure 7.12 is especially illustrative; it can be seen that the complexity for (8,8) completely dwarfs all other antenna combinations.

Figure 7.12 is an example of another phenomenon seen in simulation: the spike in complexity for the (8,8) system at 21dB. These spikes are spurious; a new simulation under the same conditions probably will not present it. The most likely explanation (unverified at this time) is that certain channel matrices cause V-CP to take much more time than usual.

The lack of bounds to the complexity of V-CP may prove to be a serious obstacle to its adoption in MIMO systems, where most applications require a constant data rate and, in consequence, fixed (or at least bound) complexity. There are two possible solutions to this problem. One is to stop the V-CP algorithm after a fixed amount of time, and have it return its best estimate at that time. This will have an effect on the probability of error; how strong an effect will depend on the frequency of the problematic channel matrices.

A second possibility is to employ the LLL reduction: such complexity spikes have not been observed when the generator matrix has been reduced. On one hand, this gives support to the hypothesis that the spikes are caused by problematic generator matrices, since the LLL reduction lends numerical stability to *Decode*. On the other hand, as will be seen later in this chapter, the use of LLL reduction does not have a clearly positive effect on the receiver's complexity.

This phenomenon illustrates another point: V-CP's complexity can be modeled as a random variable. The complexity needed to decode a vector depends on the channel matrix and the instantaneous realization of the noise. The complexity results presented here are averages; a more thorough understanding of V-CP requires obtaining second-order estimator statistics. Besides helping to understand the behavior of the complexity spikes, these statistics might be used to determine the peak processing power needed to ensure the algorithm will finish its task with a given probability, and under a given average SNR.

It is also interesting to see that V-CP's complexity tends to an asymptote as the average SNR grows. It was noted before that, as the noise power diminishes, the received point gets closer to a lattice point and the algorithm is able to find this point very quickly. It is conjectured that the variance of the complexity will also diminish under such circumstances.

Figure 7.13 is a "close-up" of figure 7.12; $n_T = 8$ and $n_R = 12$ and 16. It clearly shows that there is a value of average SNR (around 16.5dB) below which

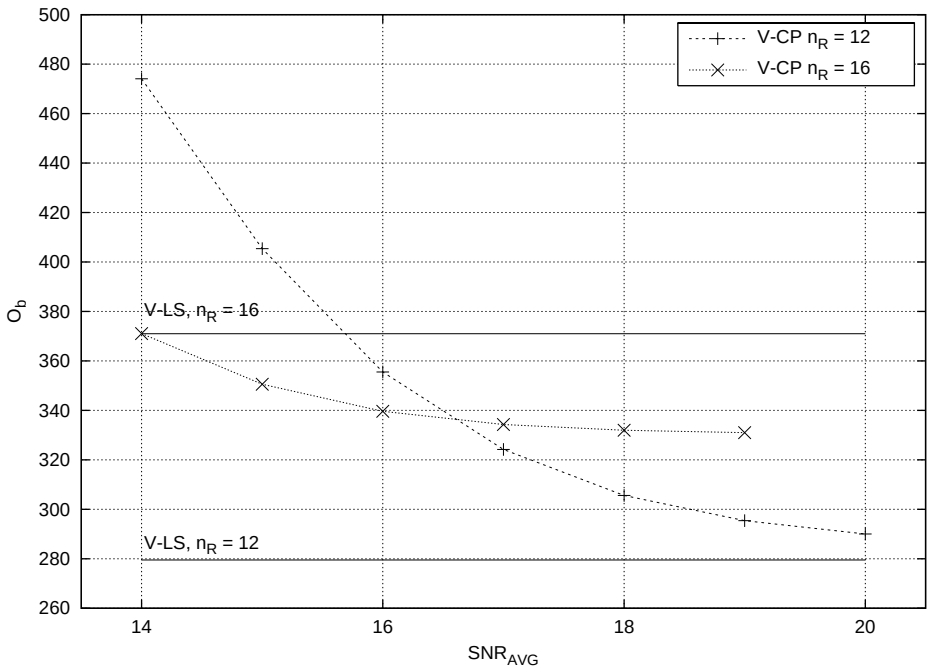

Figure 7.13: O_b as a function of average SNR for V-LS and V-CP (without basis reduction); $n_T = 8$, $n_R = 8$, 12, $L = 10$, 16-QAM

the smaller (8,12) system is more complex than the larger (8,16) one. It also shows, remarkably, that V-CP's complexity compares favorably with that of V-LS for some values of average SNR. The conclusion is that, in high SNR situations, V-CP might be preferable to V-LS, since it offers a better error rate at lower complexity.

7.3.1 LLL reduction complexity

The complexity of V-CP with and without the LLL matrix reduction is examined. The results presented account for the total receiver complexity: both the reduction and the actual vector processing. Figures 7.14 through 7.17 show the results for $L = 10$.

The results are clear; in most cases, the speed increase in *Decode* is insufficient to compensate for the complexity increase caused by the LLL reduction. There are only two exceptions: for a (6,6) system with average SNR less than 21dB, and for an (8,8) system, performing the LLL reduction is actually beneficial. Note, in figure 7.17, that the complexity spike visible in the case of no reduction does not appear when the reduction is performed. Note that both cases were simulated under the exact same noise and channel matrices, and for the same duration.

The LLL reduction increases the complexity of the algorithm's setup phase while decreasing that of the vector estimation phase; for this reason, the more vectors are estimated in a block, the more beneficial to the receiver's complexity it will prove to

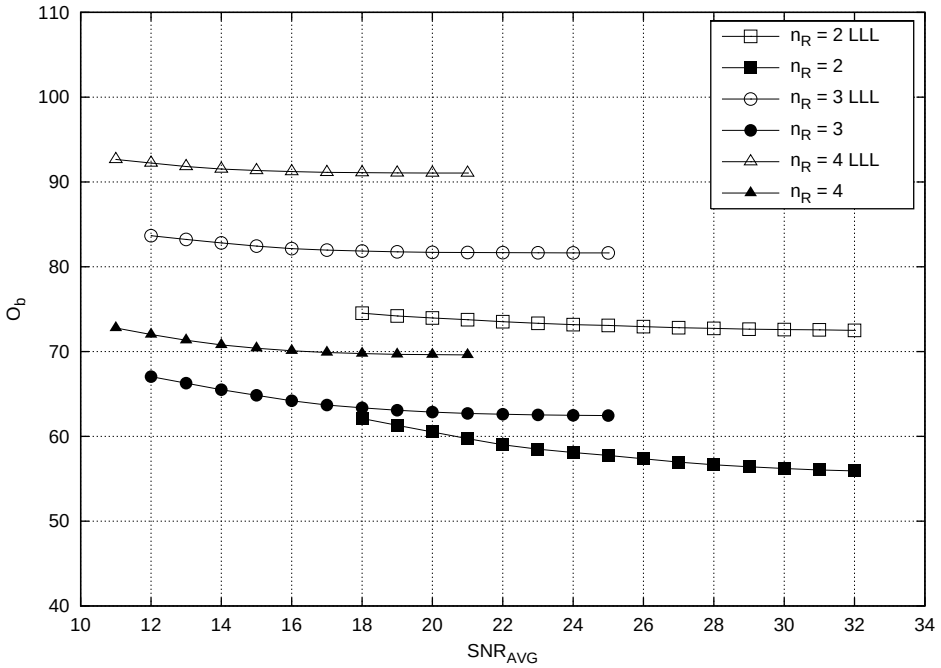

Figure 7.14: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 10$, 16-QAM

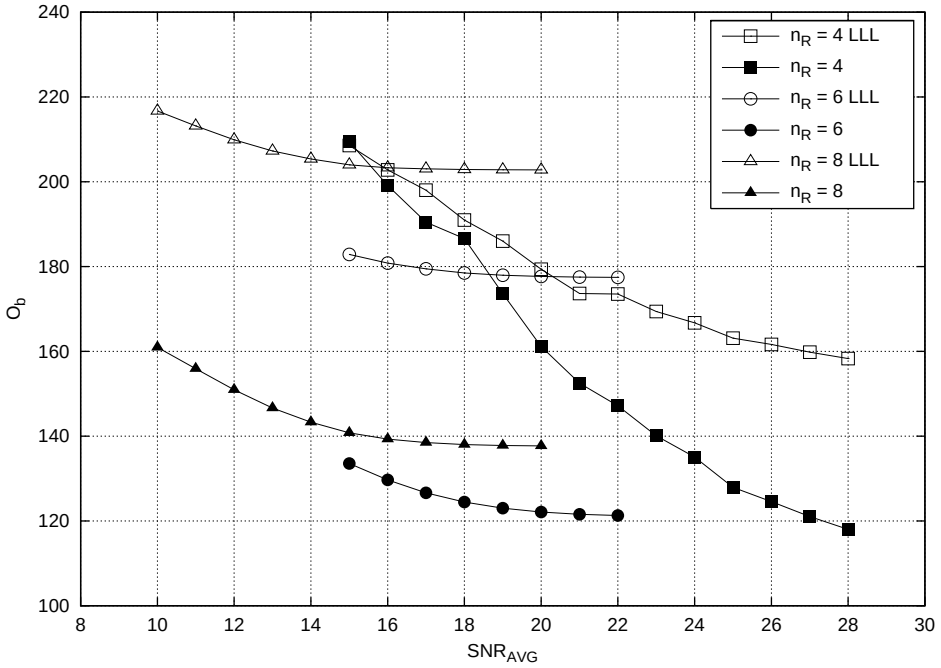

Figure 7.15: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 10$, 16-QAM

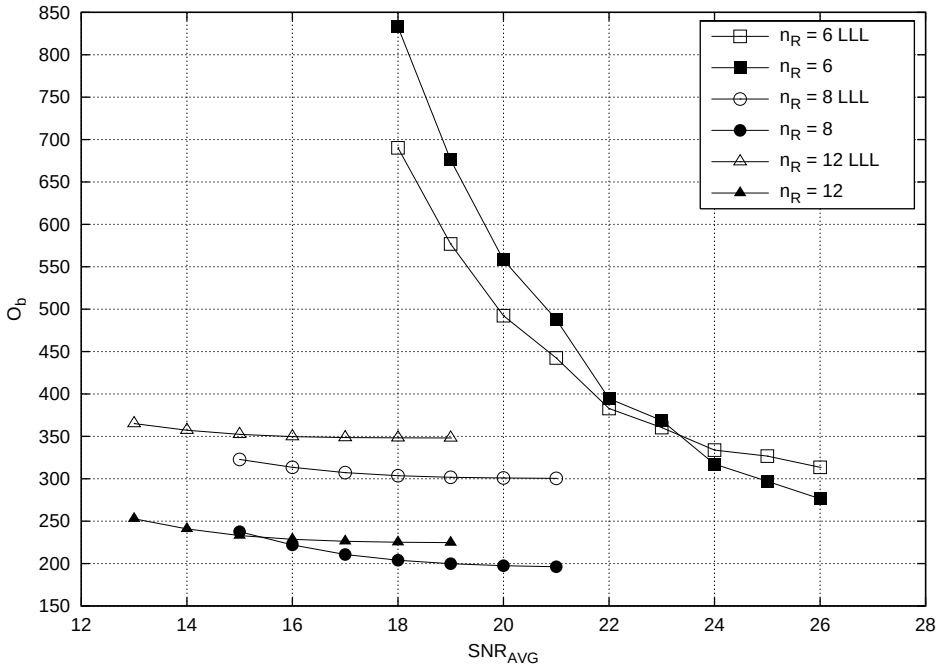

Figure 7.16: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 10$, 16-QAM

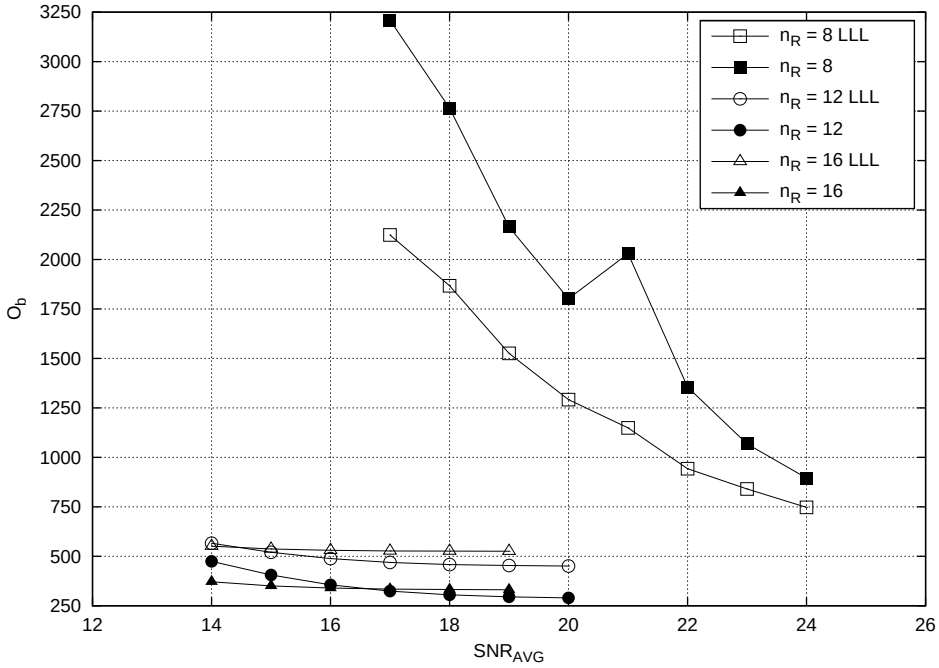

Figure 7.17: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 10$, 16-QAM

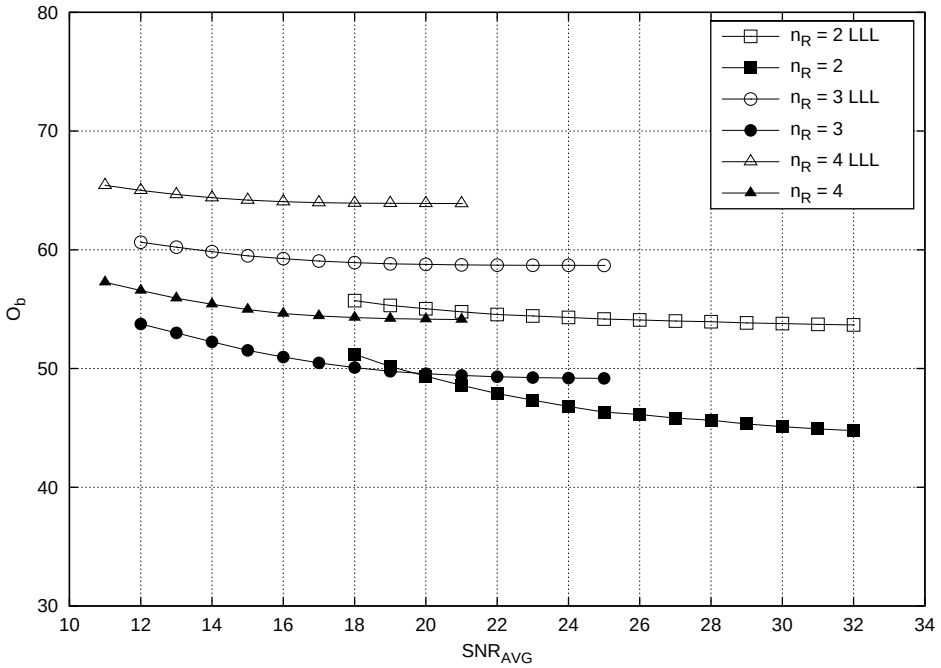

Figure 7.18: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 2$, $n_R = 2$, 3, and 4, $L = 100$, 16-QAM

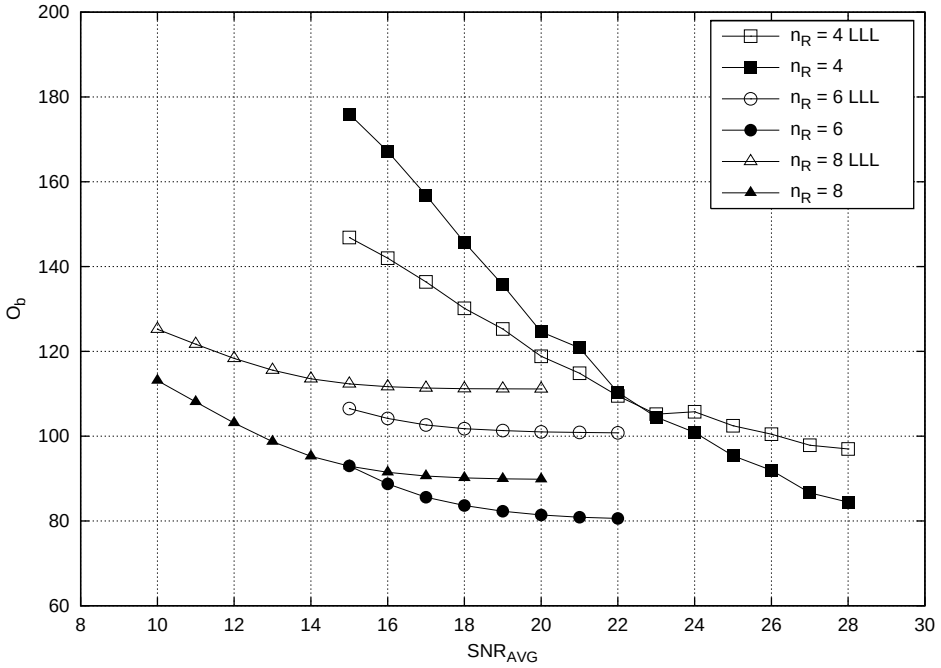

Figure 7.19: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 4$, $n_R = 4$, 6, and 8, $L = 100$, 16-QAM

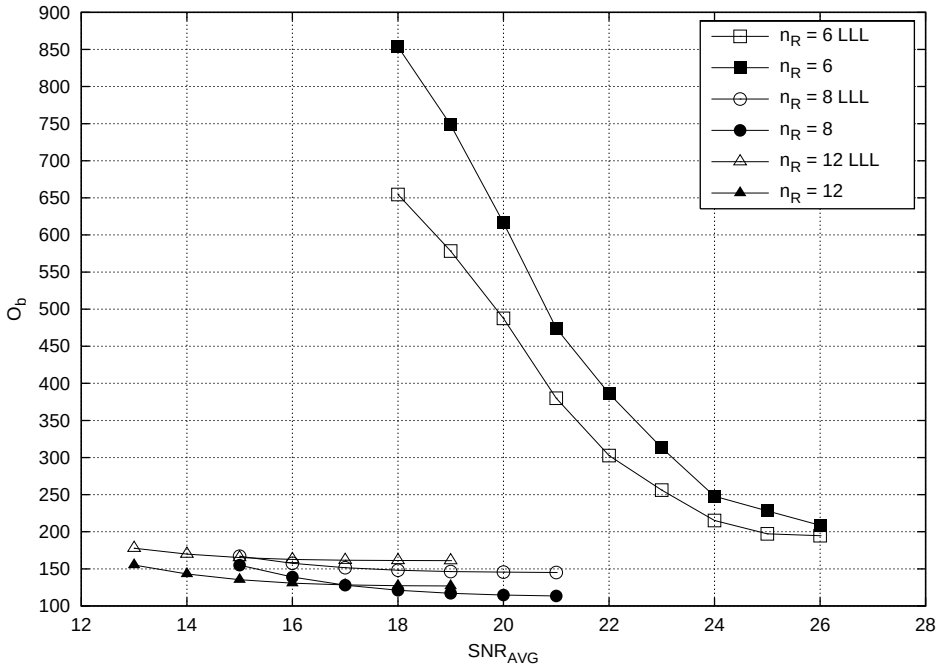

Figure 7.20: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 6$, $n_R = 6$, 9, and 12, $L = 100$, 16-QAM

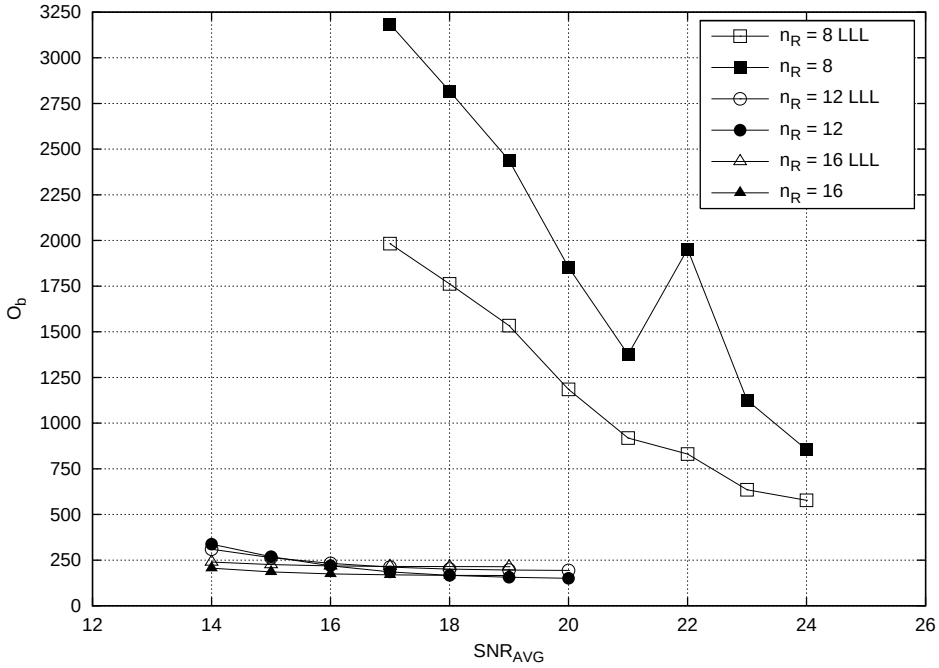

Figure 7.21: Complexity comparison between V-CP with and without LLL matrix reduction; $n_T = 8$, $n_R = 8$, 12, and 16, $L = 100$, 16-QAM

be. Figures 7.18 to 7.21 are similar to those presented above except that the block length L has been increased to 100.

As can be seen, the situation has improved, but not much: from figure 7.14, a (2,3) system ($L = 10$) without reduction has a complexity of $O_b \approx 65$ (at 20dB), and with reduction it increases to $O_b \approx 85$. When $L = 100$ the complexity figures are approximately 48 and 58, respectively. In most cases, the LLL reduction proves so costly that it ends up increasing the receiver's total complexity despite decreasing *Decode*'s, even for $L = 100$. Note that in a MIMO system, increasing L has the cost of increasing *BLER*.

In summary, the only apparent benefit to performing the LLL reduction, in most cases, is the suppression of the complexity spikes.

7.4 V-CP compared to maximum-likelihood

As explained in chapter 6, a shortcoming of lattice decoders when applied to MIMO receivers is that, especially in low SNR, they can produce an estimate that does not belong to the constellation in use. The algorithm V-CP uses a novel method to obtain a valid constellation point even when this happens. In this section, this method is compared to maximum-likelihood decoding, on one hand, and to the simpler method, called *V-Fixed*, which consists in simply assigning a fixed constellation point as estimate.

Figures 7.22 to 7.27 present the bit error rates obtained, for $n_T = 2$ with $n_R = 2, 3,$ and 4, and for $n_T = 4$ with $n_R = 4, 6,$ and 8. Of special interest are situations of low average SNR since, if noise power is low, then most points returned by *Decode* will be constellation points.

First, it is established that, in all cases, V-CP is superior to simply assigning a fixed constellation point as estimate, at negligible cost in complexity. At $BER = 10^{-3}$, the gain goes roughly from half a decibel for a (2,2) system to 1.5 decibels for a (4,8) system.

Second, it is determined how far from optimum V-CP is. In general, V-CP gets closer to ML as n_R grows with respect to n_T. For a (2,2) system, the difference between the two is roughly 2 decibels; for a (2,4) system, only a fraction of a decibel separates them. In a (4,8) system they are nearly indistinguishable for $BER = 10^{-3}$ or less.

7.5 Conclusions

The rule of thumb that claims V-BLAST-based algorithms are faster, while lattice decoding has better error rates, has been proved to be not quite true. For large numbers of receive antennas, V-BLAST's error rates are very close to lattice decoding; for certain values of average SNR, n_R, and n_T, lattice decoding is faster than V-BLAST.

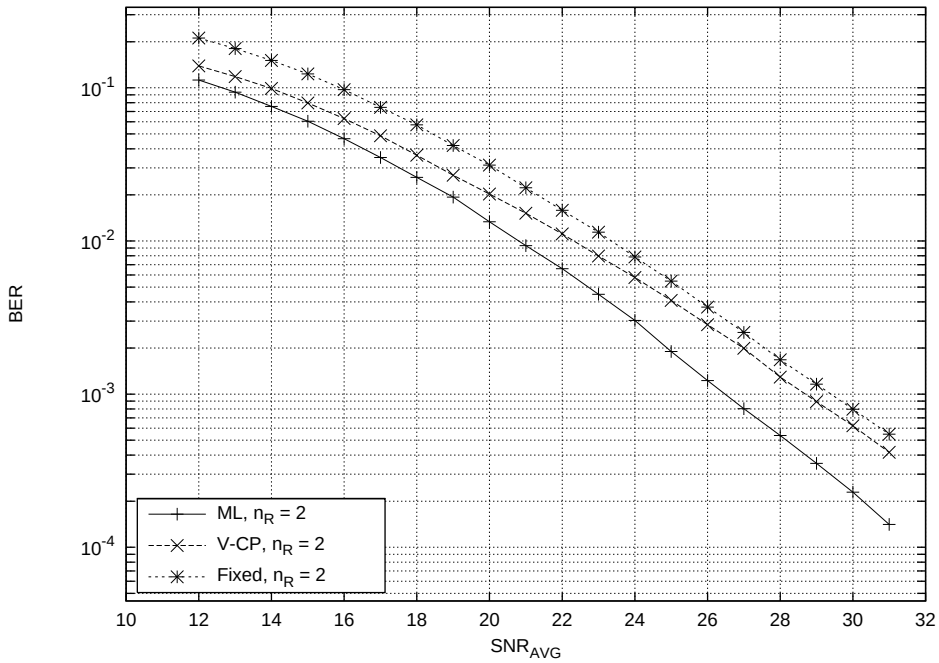

Figure 7.22: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 2$, 16-QAM

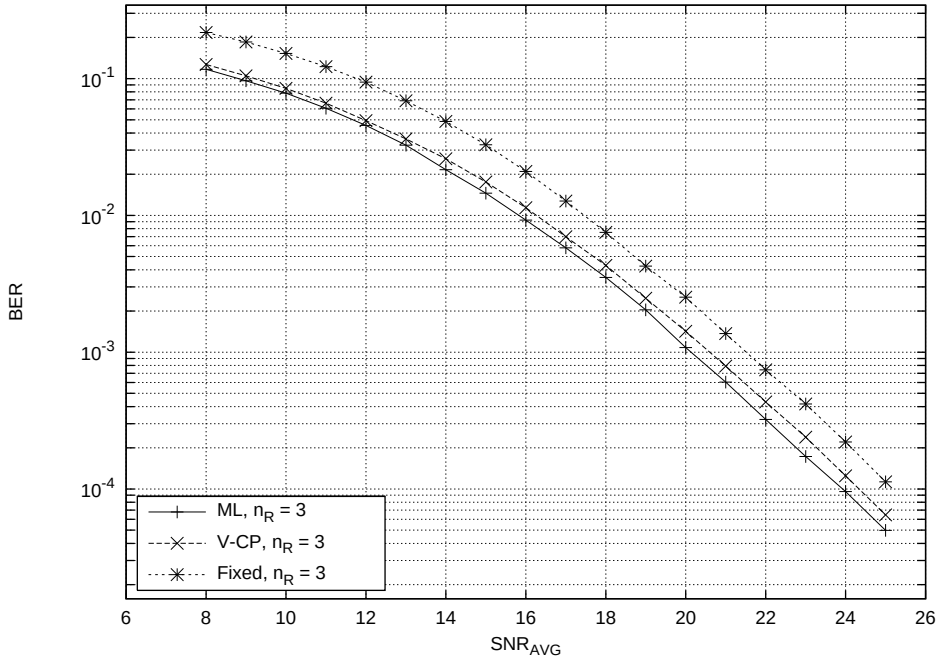

Figure 7.23: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 3$, 16-QAM

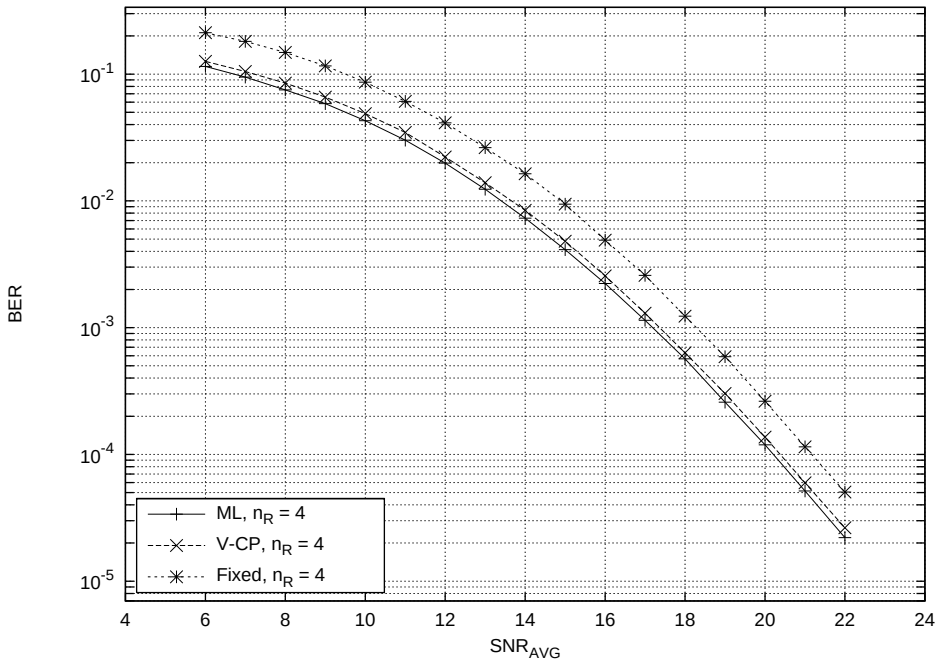

Figure 7.24: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 2$, $n_R = 4$, 16-QAM

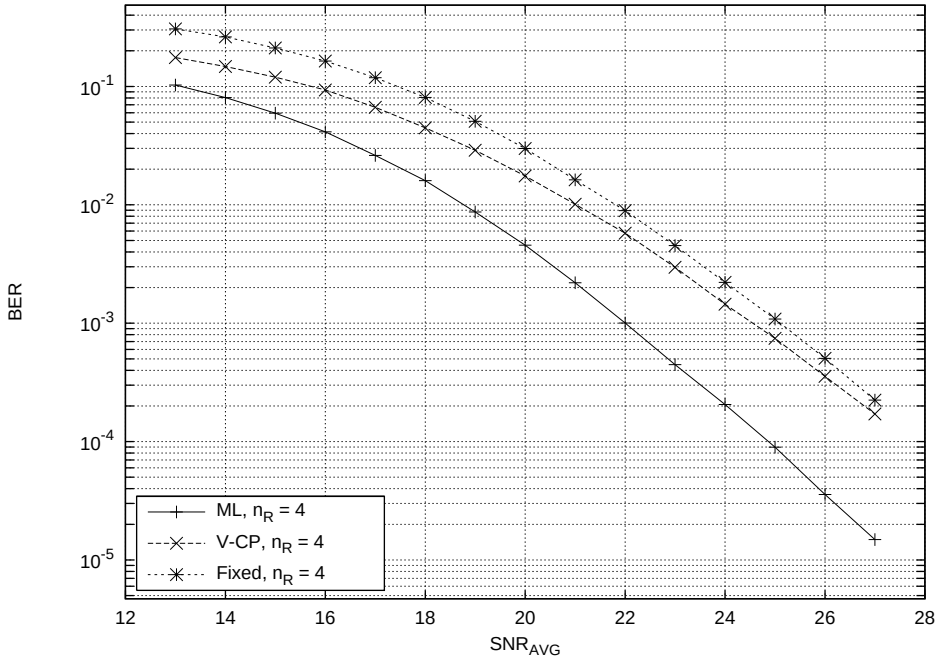

Figure 7.25: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 4$, 16-QAM

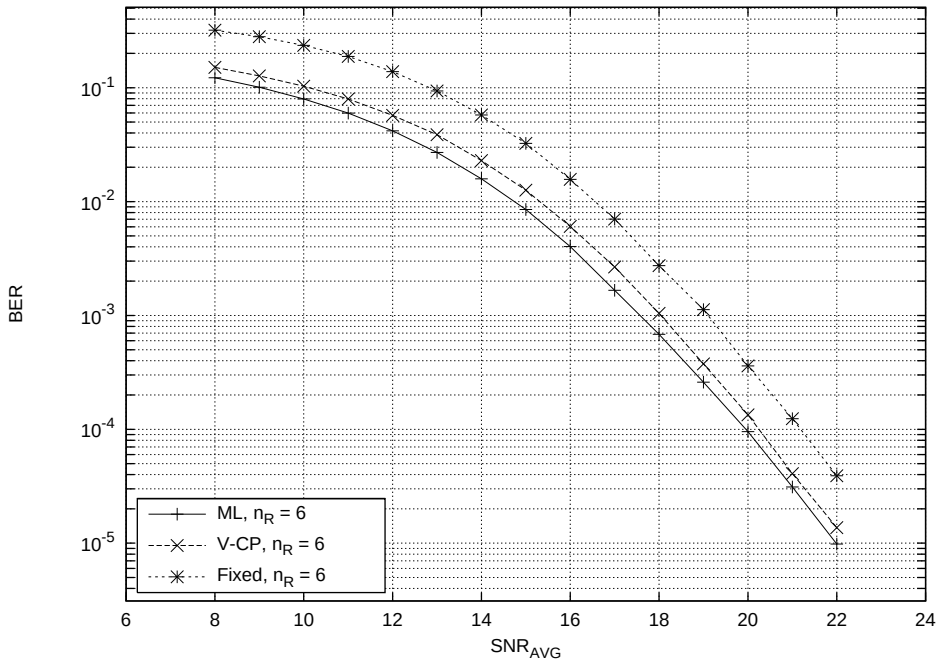

Figure 7.26: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 6$, 16-QAM

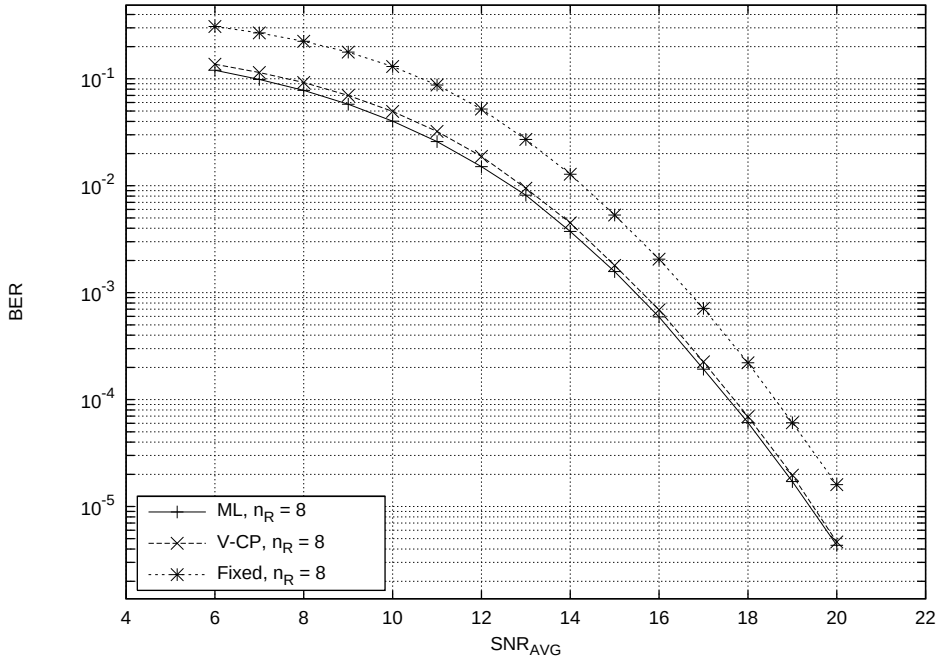

Figure 7.27: *BER* comparison between maximum-likelihood, V-CP, and V-Fixed; $n_T = 4$, $n_R = 8$, 16-QAM

The choice of one over the other will depend on the specific details of the implementation. If maximizing error rate performance is a clear-cut priority, then V-CP is the best choice. In more realistic situations, however, there are other factors to consider. One is the range of average SNR values at which the receiver will be required to operate. Another is the variance in complexity from one vector to another and from one block to the next that the system must tolerate.

The method employed in V-CP to map vectors outside the lattice back to a constellation point has been justified. At practically no cost in complexity, this method has been proved to be almost indistinguishable from maximum likelihood decoding for systems where n_R is greater than n_T; for symmetrical systems, the loss of 2 decibels with respect to ML can still be acceptable in implementations where the extra complexity of ML decoding is not desirable.

Chapter 8

Conclusions

At the outset, the objective for this thesis was to explore the performance and complexity of MIMO communication systems based on layered space-time codes. After analyzing, implementing, simulating and comparing a large number of algorithms, all the while gathering enormous quantities of data, it is time to look at the results and draw conclusions.

It is believed that the complexity analysis presented in this thesis are important, or at least useful, because they provide information that is needed for future practical implementations of layered space-time codes that meet economic constraints of cost, power consumption, and reliability. Among other things, these analyses might help identify bottlenecks in the algorithms, or operations that can be parallelized.

The results, though, go beyond raw numbers. Interesting, non-trivial behaviors have been observed in the algorithms, and explanations have been put forth where possible. Equally important, some light has been shed on the question of whether some algorithms are better than others and under which circumstances. In particular, hard facts have shown that there is no clear-cut division between the sub-optimal-but-fast and the optimal-but-slow algorithms. It has been amply demonstrated that the amount of space an algorithm takes up on paper has no bearing on its complexity.

In the process of studying the algorithms, there has been opportunity to offer novel ideas, modifications and improvements. The limitation of V-BLAST due to its inability to find better detection orderings has been exposed; least-squares techniques have been used in the development of a new variant of V-BLAST, V-LS; lattice decoding has been modified so that it can be used to receive vertical codes, using a novel technique to cope with a limitation of this type of algorithms.

Beyond their theoretical properties, one factor that will prove to be of importance to the wide-spread adoption and use of vertical space-time codes is the feasibility of implementing receiver algorithms in off-the-shelf, inexpensive hardware. A digital signal processor would appear to be especially well-suited for this task, because of its ability to perform vector and matrix products with great speed. In order to illuminate this aspect of the problem, both V-BLAST and V-LS have been imple-

mented in a modern Texas Instruments DSP; results confirm that such a DSP is indeed suitable for the task, providing a data rate of a few hundreds of kilobits per second for small antenna arrays. Reaching higher rates requires more work to be done in optimizing and improving the existing algorithms.

In summary, the contributions that have been made in this thesis are the following.

- The effect that the lack of knowledge of each symbol's energy has on the estimation ordering of V-BLAST has been identified and quantified. This opens the door to innovations in the design of signal constellations that help V-BLAST achieve the optimum detection order.

- Some general conclusions on the characteristics of V-BLAST's complexity have been reached. The dependence of the complexity on the block length and the number of receive antennas has been determined. The trade-offs between complexity, array size, error rate, and block length have been identified.

- Least-squares techniques have been applied to the reception of vertical space-time codes, resulting in a new algorithm, V-LS, that is considerably less complex than V-BLAST while having exactly the same error rate.

- V-LS has been compared to the sorted-QR V-BLAST algorithm, V-SQR, and situations where each is preferable have been identified.

- Four different methods of calculating the matrix pseudo-inverse operation in V-BLAST have been studied, and their complexity and effect on the algorithm's stability have been determined. The thin QR decomposition has been found to be the most attractive method to perform this calculation.

- Lattice decoding has been adapted to the reception of vertical space-time codes, and an algorithm, called V-CP, has been proposed. Lattice decoders have the drawback of sometimes estimating a vector that is not part of the signal constellation. V-CP includes a novel solution to this problem, which has a negligible effect on the algorithm's complexity.

- V-LS has been compared to V-CP, with some remarkable results that indicate that V-BLAST–based algorithms are not always faster, nor do they have much worse error rates, than optimum decoders.

- The error-rate performance of V-CP has been compared to maximum-likelihood decoding. Although sub-optimal, V-CP has been shown to be close to optimum in many situations, and practically indistinguishable from optimum in others.

- Some peculiarities in the behavior of V-CP have been identified, such as complexity spikes, dependence of complexity on SNR, and lattices of smaller dimensions being more difficult to decode than larger ones for low values of SNR. These behaviors could significantly affect a practical application of V-CP and related lattice decoding algorithms.

- The benefit obtained from reducing the channel matrix before executing V-CP has been evaluated. It has been determined that, in most cases, the reduction has a negative impact on the complexity of the receiver; its only apparent benefit is in improving the stability of the algorithm.

- An extensive, architecture-agnostic, prospective study of the complexity of V-BLAST, V-LS, V-SQR, and V-CP has been carried out. It is expected that the results obtained will serve as the basis of future developments in the implementation of these algorithms in practical applications.

- Prospective studies of the feasibility of using a commercial, general purpose digital signal processor have been carried out for the V-BLAST and V-LS algorithms. Results are encouraging, since performance was good even though the DSP used is one generation behind and is not particularly optimized for this task.

- A simulator platform for MIMO systems has been developed; it is detailed in appendix B. Consisting of 5000 lines of code, it is a powerful tool to explore new algorithms. All performance and complexity results presented in this thesis were calculated with its help.

Much remains to be done in this area. Even though work in MIMO systems and space-time codes can be traced back to at least 1998, it still has not left the research labs. The perennial problem of receiver complexity is still present. Code design and construction is still in large part an open field, as is constellation design. The hypothesis that receivers have perfect channel state information needs to be tested, as well as all the assumptions about the channel model.

Chapitre 9

Résumé en français de la thèse

9.1 Introduction

À l'heure actuelle, la recherche mondiale dans le domaine des communications numériques sans-fil est de plus en plus élaborée. L'activité de développement cherche à s'adapter aux besoins en communication de chaque individu. La quantité d'information disponible augmente de façon exponentielle.

Les systèmes sans-fil possèdent de forts avantages : on n' a pas besoin de câbler chaque bâtiment, et le signal transmis est capable de couvrir une grande surface, ce qui facilite la diffusion de l'information.

Le canal radio a une capacité qui est rapidement dépassée par le volume d' information que l'on souhaite transmettre. Aujourd'hui, l' allocation de la ressource « spectre» est considérée comme un problème prioritaire ; ressource qui devient de plus en plus rare et en conséquence de plus en plus chère.

Heureusement, des techniques qui promettent une utilisation plus efficace du spectre ont été proposées. Ces techniques, connues sous le nom de *codage espace-temps*, exploitent une nouvelle forme de diversité, la diversité *spatiale*, en conjonction avec la diversité temporelle pour obtenir une efficacité spectrale sans précédents.

La conception de ce type de codes, et d'algorithmes de réception, présente une grande difficulté. Par ailleurs, trouver des récepteurs qui peuvent être implantés de façon fiable et à des coûts raisonnables, est une condition nécessaire pour exploiter ce type de codes.

Dans cette thèse la conception de ce type de récepteurs a été étudiée. Pour cette étude, on a choisi les codes connus sous l' appellation *vertical layered codes* ; leurs performance a été déterminée et plusieurs algorithmes de réception ont été analysés et comparés.

9.2 Codes espace-temps : état de l'art

9.2.1 Description du problème

Une caractéristique des systèmes de communication sans fil est que le signal émis suit de multiples chemins avant d'arriver à l'antenne de réception. Chacune de ces réflexions du signal arrive à des temps différents et subit des atténuations différentes. Quand ces réflexions s'ajoutent de façon destructive, le phénomène connu comme évanouissement (*fading*) apparaît. Pour avoir des communications fiables sur le canal sans fil, il faut mettre en place de méthodes pour mitiger ces effets. La technique la plus répandue est celle qui exploite la diversité.

Récemment, le même phénomène qui donne lieu au *fading* a été utilisé pour le combattre. Les réflexions multiples peuvent être utilisées pour créer de la diversité ; aucune autre ressource (comme bande passante ou puissance émise) n'est nécessaire. Ces techniques sont utilisées ne pas seulement pour fournir de la diversité, mais aussi pour augmenter le débit.

9.2.2 Définition d'un système MIMO

Les systèmes qui utilisent de multiples antennes en émission et en réception sont connus comme des systèmes *multiple-input, multiple-output* (MIMO).

Dans un système MIMO, n_T antennes sont utilisées comme émetteurs, et n_R antennes comme récepteurs. Les donnés sont séparés en n_T groupes, et chaque groupe est transmis par une antenne différente. Toutes les antennes émettrices sont synchronisées par rapport aux symboles, utilisent la même bande de fréquences, et la même constellation $S = \{s_1, s_2, \cdots, s_{2^b}\}$; b est le nombre de bits d'information véhiculés par chaque signal en S. L'énergie moyenne par symbole est définie E_s. Dans la suite, on suppose que $n_R \geq n_T$.

Le canal est supposé être non-sélectif en fréquence avec des évanouissements lents. Il est représenté par une matrice $\mathbf{H}$, avec n_R files et n_T colonnes. L'élément h_{ij} est la fonction de transfert du canal entre l'émetteur j et le récepteur i. Les éléments de $\mathbf{H}$ sont supposés être des variables aléatoires complexes gaussiennes, indépendantes et avec distributions identiques, de moyenne nulle et puissance 0.5 par dimension. Le canal est supposé être constant pendant la transmission d'un *bloc* de taille $L \times n_T$ symboles ; il change d'un bloc à l'autre. On supposera dans la suite qu'il n'a pas de mémoire entre blocs. Un canal avec ces caractéristiques est connu comme un canal à évanouissements par bloc (*block fading*). Il est supposé que le récepteur a estimé le canal sans erreur. Ce type de canal modélise raisonnablement un canal à l' intérieur d' un bâtiment (modèle *indoor*).

Soit $\mathbf{a} = (a_1, a_2, ..., a_{n_T})^T$ le vecteur de symboles émis ; tous les éléments de $\mathbf{a}$ appartiennent à la même constellation S, et ils partagent la même puissance moyenne $E_s = \mathcal{E}[|a_i|^2]$, $1 \leq i \leq n_T$. Le vecteur reçu $\mathbf{r}$ peut être écrit comme :

$$\mathbf{r} = \mathbf{Ha} + \mathbf{n}, \tag{9.1}$$

où $\mathbf{n}$ est un vecteur de bruit ; ses éléments sont des variables aléatoires gaussiennes indépendantes symétriques circulaires de moyenne nulle et puissance N_0 par dimension.

9.2.3 Mesures de performance de systèmes MIMO

La performance (évaluée par le taux d'erreur) des systèmes MIMO peut être mesuré comme un taux d'erreur par bit (*Bit Error Rate*, ou *BER*). Cependant, comme le canal découpe le flux de donnés en blocs, il est aussi intéressant d'étudier le taux d'erreur par bloc (*BLock Error Rate*, ou *BLER*. Une erreur de bloc est définie comme l'apparition de au moins une erreur dans le bloc.

La probabilité d'erreur est normalement calculée en fonction du rapport signal sur bruit moyen (*SNR moyen*, ou γ), qui est défini comme le rapport de la puissance reçue sur chaque antenne et la puissance de bruit qui affecte chaque composante du vecteur reçu :

$$\gamma = \frac{\mathcal{E}[|\mathbf{Ha}|^2]}{\mathcal{E}[|\mathbf{n}|^2]}, \tag{9.2}$$

ou $|\cdot|$ est la norme ℓ^2. Soit $\mathbf{s} = \mathbf{Ha}$; alors,

$$\begin{aligned} \mathcal{E}[|\mathbf{s}|^2] &= \mathcal{E}\left[\sum_{k=1}^{n_R} |s_k|^2\right] \\ &= E_s \cdot n_T \cdot n_R. \end{aligned} \tag{9.3}$$

Comme $\mathcal{E}[|\mathbf{n}|^2] = 2 \cdot n_R \cdot N_0$, et en utilisant (9.3) et (9.2), le SNR moyen peut s'écrire sous la forme :

$$\gamma = \frac{E_s \cdot n_T}{2N_0}. \tag{9.4}$$

On suppose, dans la suite, que $E_s = 1$. Cela implique que le SNR moyen vaut :

$$\gamma = n_T/(2N_0).$$

9.2.4 Limites sur la performance et capacité des systèmes MIMO

Dans un premier temps, la capacité d'un système à une seule antenne est comparée avec celui d'un système MIMO. Pour un système à bruit gaussien additif et évanouissements de type Rayleigh, la capacité est donné par :

$$C = \log(1 + \gamma|h|^2). \tag{9.5}$$

Ici, h est un scalaire connue par le récepteur. Si le SNR moyen est grand, une augmentation de 3dB en γ donne un gain dans l'efficacité spectrale de un bit par seconde par Hertz (bps/Hz).

L'équation (9.5) représente un système où toute l'énergie disponible est émise dans un seul canal qui existe entre l'émetteur et le récepteur. On considère un système où, sans augmenter la puissance émise, n signaux sont émis sur n canaux indépendants. Ça veut dire que $\mathbf{H} = I_n$. Dans un tel cas, la capacité est donné par :

$$C_n = n \cdot \log(1 + (\gamma/n)). \tag{9.6}$$

L'équation (9.6) a une importance capitale, parce qu'elle fournit la justification de l'étude et l'utilisation des systèmes MIMO. Grâce à l'utilisation de plusieurs canaux, la capacité augmente d'un rapport donné par l'équation (9.5). Asymptotiquement, quand le nombre de canaux augmente, la capacité a comme limite $\gamma/\ln(2)$, ce qui implique une croissance linéaire en fonction de γ.

Bien évidemment, dans le cas général $\mathbf{H} \neq I_n$ et le canal présente des évanouissements et de l'interférence entre les signaux. Ces problèmes peuvent être résolus par l'utilisation du codage ou d'autres techniques conçues pour cet effet.

L'expression générale de la capacité d'un système MIMO est :

$$C = \mathcal{E}[\log\det(I_{n_R} + (\gamma/n_T) \cdot \mathbf{H}\mathbf{H}^\dagger)]. \tag{9.7}$$

On doit remarquer que la capacité est une variable aléatoire. Quand n_T est constant et n_R augmente, la capacité est donné par :

$$\begin{aligned} \lim_{n_R \to \infty} C &= n_T \log(1 + \gamma \frac{n_R}{n_T}) \\ &= n_T \log\left(1 + \frac{1}{n_T} \cdot (\gamma \cdot n_R)\right). \end{aligned} \tag{9.8}$$

Il est évident que, si n_R augmente, on atteint la même capacité que celle d'un système avec n_T canaux indépendants et sans évanouissements, chacun avec un SNR moyen égal à γn_R.

Capacité de quelques combinaisons d'antennes

La probabilité de *coupure* (*outage probability*) d'un canal à évanouissements est la probabilité que la capacité du canal dans un instant quelconque soit inférieure qu' une certaine capacité théorique C_{th}, pour des valeurs fixes de n_T, n_R, et γ.

Afin de déterminer la probabilité de coupure, la *fonction de distribution complémentaire cumulative* (*complementary cumulative distribution function*, ou *ccdf*) est utilisé. La ccdf est la probabilité que la capacité dans un instant quelconque soit

plus grande qu'un seuil spécifié. Les figures 2.7 à 2.18 présentent la ccdf de plusieurs combinaisons des antennes.

Remarques importantes sur la capacité moyenne et la probabilité de coupure

Un aspect très intéressant des courbes de capacité moyenne est la variation de la pente quand n_R augmente. Les systèmes à une seule antenne présentent, comme on l'attendait, une augmentation de 1 bps/Hz pour 3dB de puissance supplémentaire. Par contre, pour les systèmes MIMO, la capacité moyenne augmente approximativement de n_T bps/Hz chaque fois qu'on double la puissance émise. Cela est équivalent à avoir n_T canaux indépendants.

Même pour des probabilités de coupure très faibles (environ 1%), et pour un faible nombre d'antennes, les systèmes MIMO offrent une augmentation énorme de la capacité par rapport aux systèmes à une seule antenne.

9.2.5 Une note sur la diversité

Une question intéressante que l'on peut se poser est : quelle est la diversité générée par les systèmes MIMO ? Rappelons-le, ce type de systèmes profitent de la diversité spatiale et temporelle, mais au même temps, les antennes de réception sont partagés par plusieurs antennes d'émission, ce qui introduit une interférence supplémentaire et qui, nécessairement, aura un effet négatif sur la performance du système.

Il est possible d'obtenir une expression générale pour la capacité. Quand n_R est très grand, la capacité est donné dans (9.8) :

$$\lim_{n_R \to \infty} C = n_T \log \left(1 + \frac{1}{n_T} \cdot (\gamma \cdot n_R) \right).$$

Cela suggère que n_T voies de diversité ont été générées ; en outre, le SNR moyen a été multiplié par un facteur n_R. Ceci veut dire que l'ordre de diversité qui est potentiellement disponible dans un système (n_T, n_R) est égal à $D = n_T \cdot n_R$.

9.2.6 Codes espace-temps

Les codes espace-temps ont comme but d'exploiter l'énorme capacité offerte par les systèmes MIMO en combinant les diversités spatiale et temporelle. Il existe trois types de codes espace-temps : les codes ST en treillis, les codes ST en bloc, et les codes ST en couches (*layered*). Cette thèse s'intéresse à un type particulier de *layered codes*, connus sous le nom de «codes espace-temps verticaux».

Historiquement, les *layered codes* ont été les premiers codes proposés dans le contexte des systèmes MIMO. Aujourd'hui, ils représentent la technique la plus prometteuse pour augmenter la capacité des systèmes sans fils. Ils offrent une efficacité

spectrale très grande, leur conception est relativement simple ainsi que l'architecture des récepteurs capables de les décoder. En plus, ils ont des performances en termes du taux d'erreur qui s'adaptent bien à la grande majorité des applications sans-fil.

Un code espace-temps associe une groupe de b bits d'information avec un vecteur de code $\mathcal{A} = \mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_L$. $\mathcal{A}$ est un mot du code espace-temps. $\mathcal{A}$ représente un bloc, et en conséquence il subit un évanouissement constant. Dans ces conditions, on a déterminé que la probabilité d'erreur par paires d'un récepteur à maximum de vraisemblance est bornée par :

$$P(\mathcal{C} \to \mathcal{E}) \leq \left(\prod_{i=1}^{r} \lambda_i \right)^{-n_R} (1/2N_0)^{-rn_R}, \tag{9.9}$$

ou r est le rang de la matrice $\mathbf{A}$, qui dépend des mots de code $\mathcal{C}$ et $\mathcal{E}$, et λ_i sont les valeurs propres non nulles de $\mathbf{A}$.

L'exposant du SNR dans l'équation (9.9) est appelé l'*avantage de diversité* du code ; sa valeur maximale est $n_T \cdot n_R$. D' autre part, $(\lambda_1 \lambda_2 \cdots \lambda_r)^{1/r}$ s'appelle l'*avantage de codage*, est il donne une approximation du gain obtenu par l'utilisation d'un système codé par rapport à un système non codé qui opère avec le même avantage de diversité.

On doit remarquer que, même dans le pire scénario (c'est à dire quand $r = 1$), il y a un avantage de diversité égal à n_R, auquel il faut rajouter l'avantage de codage.

La conception des codes espace-temps présente un compromis entre la taille de la constellation, la diversité, et le débit. Pour le cas d'une avantage de diversité maximale (égale à $n_T \cdot n_R$), si la constellation utilisée est de cardinalité 2^M signaux, le débit maximal est M bits par seconde par Hertz.

La conception de bons codes espace-temps doit prendre en compte ce compromis, ainsi que la complexité des récepteurs. Aujourd'hui, ce domaine, champ de recherches intensives, est dans une phase prospective. Des codes très puissants et efficaces sont connues, mais en général ils sont extrêmement difficiles (voire impossible) à décoder.

Les *Layered Space-Time Codes*

Les codes espace-temps en couches (*layered space-time codes*, ou *LSTC*) ont été introduits par Foschini et Gans ([13], [14]). L'idée fondamentale est de séparer un système (n_T, n_R) dans n_T systèmes (1, n_R). Les symboles émis sont estimés un à la fois. Pendant la phase d'estimation de chaque symbole, les symboles qui n'ont pas été encore estimés sont considérés comme de l'interférence. Le but est de pouvoir utiliser toutes les connaissances développées pour les systèmes à diversité spatiale (1, n_R). Au même temps, une partie non négligeable de la capacité du canal est exploitée. Cette méthode est connue comme *Bell Labs layered space-time*, ou *BLAST*.

Les avantages de BLAST sont sa simplicité et son efficacité spectrale. Les récepteurs BLAST sont aussi moins complexes que ceux pour d'autres types de codes. Ses performances sont adéquates pour plusieurs applications. Son principal désavanta-

ge, par rapport aux autres types de codes espace-temps, est leur performance reste moins bonne.

On doit remarquer que la complexité des récepteurs est l'un des obstacles les plus importants pour l'adoption des techniques espace-temps.

9.3 Les principes de BLAST Vertical(VBLAST)

9.3.1 Introduction

Même si la formulation de l'algorithme V-BLAST est très simple, il a été très difficile de trouver des expressions pour mesurer ses performances tel que la probabilité d'erreur ou la probabilité de coupure. Très récemment, quelques résultats, pour des cas très particuliers, ont été reportés. De même, des résultats particulièrement intéressants seront reportés dans ce rapport malgré que l' objectif fondamental de cette thèse soit orienté vers l'étude de la complexité algorithmique et la recherche de nouveaux algorithmes de réception.

9.3.2 Codage V-BLAST

Les codes verticaux répètent les symboles émis une seule fois. Pour cette raison, ils n'exploitent pas la diversité temporelle.

9.3.3 Détection V-BLAST : ordre, suppression et annulation

La détection V-BLAST consiste à estimer chaque symbole émis en séquence. À chaque pas de l'estimation, les symboles déjà estimés sont soustraits du signal reçu $\mathbf{r}$ (c'est à dire, *supprimés*), et les symboles inconnus sont considérés comme de l'interférence (c'est à dire *annulés*).

Il faut encore déterminer l'ordre d'estimation. L'un des aspects clef de V-BLAST est que des ordres différents produisent des taux d'erreur différents. Soit $\mathcal{K}_i$ un ordre d'estimation, définit comme un ensemble ordonné de nombres entiers k, $1 \leq k \leq n_T$. Évidemment, il y a $n_T!$ ordres possibles. L'un d'entre eux est optimal dans le sens qu'il produit le taux d'erreur minimal. Soit l'ordre optimal $\mathcal{K}_o = \{k_1, k_2, \ldots, k_{n_T}\}$. La méthode utilisée pour déterminer $\mathcal{K}_o$ est expliqué dans la suite. Pour l'instant, on suppose que l'ordre optimal a été trouvé. La détection de a_{k_i} se fait en deux pas :

I. Suppression des symboles. On suppose $i > 1$. Soit $\mathbf{h}_1, \mathbf{h}_2, \ldots \mathbf{h}_{n_T}$ les colonnes de la matrice du canal $\mathbf{H}$. Si les symboles $a_{k_1}, a_{k_2}, \ldots, a_{k_{i-1}}$ on été déjà estimés, alors leur contribution à $\mathbf{r}$ peut être supprimée :

$$\mathbf{r}_{k_i} = \mathbf{r} - \hat{a}_{k_1}\mathbf{h}_{k_1} - \hat{a}_{k_2}\mathbf{h}_{k_2} - \cdots - \hat{a}_{k_{i-1}}\mathbf{h}_{k_{i-1}}. \tag{9.10}$$

II. Annulation de l'interférence. Les symboles $a_{k_{i+1}}, a_{k_{i+2}}, \ldots, a_{k_{n_T}}$ n'ont pas été encore estimés. Cependant, grâce à la connaissance de $\mathbf{H}$, ils peuvent être annulés. L'annulation est synonyme à faire une pondération linéaire du vecteur reçu de telle façon qu'un certain critère soit satisfait. Les critères les plus courants sont le critère du forçage à zéro (*zero-forcing*) et le critère de l'erreur quadratique minimale (*minimum mean-squared error* ou *MMSE*).

Le critère *MMSE* offre une meilleure performance en taux d'erreur que le critère *zero-forcing* dans le cas où le SNR moyen est faible. Cependant, il a un désavantage important : le récepteur doit connaître la valeur de N_0. Pour les SNR moyen grands, MMSE et *zero-forcing* présentent des performances équivalentes. Donc, on se concentrera dans la suite sur le critère *zero-forcing*.

Le principe du *zero-forcing* est de trouver $\mathbf{w}_{k_i} \in \mathbb{C}^{1\times n_R}$ tel que :

$$\mathbf{w}_{k_i}\mathbf{H} = [0, \ldots, 0, 1, 0, \ldots, 0]$$

où l'élément égal à 1 se trouve dans la position k_i. Alors, $\hat{a}_{k_i}$ est donné par :

$$\begin{aligned} \hat{a}_{k_i} &= \mathbf{w}_{k_i}\mathbf{r}_{k_i} + \mathbf{w}_{k_i}\mathbf{n} \\ &= \mathbf{w}_{k_i}\mathbf{H}\mathbf{a} + \mathbf{w}_{k_i}\mathbf{n}. \end{aligned} \tag{9.11}$$

Cette procédure peut être interprétée d'un point de vue géométrique comme la projection de $\mathbf{r}_{k_i}$ sur un vecteur orthogonal à l'espace vectoriel $n_T - i$ dimensionnel qui est généré par les colonnes de $\mathbf{H}$, ce qui correspond aux symboles qui n'ont pas encore été estimés.

L'ordre optimal. L'ordre optimal est donné par le SNR moyen de chaque a_{k_i} : le symbole avec le SNR le plus fort doit être détecté en premier lieu. Il sera suivi du symbole le plus fort parmi ceux qui restent à détecter. Cette procédure se répète jusqu'à ce que tous les symboles aient été détectés. C'est à dire, pour chaque i, a_{k_i} doit avoir un SNR plus grand que a_{k_i}, a_{k_i+1}, ..., $a_{k_{n_T}}$.

Le SNR post-détection de a_{k_i} peut être déterminé directement à partir de l'équation (9.11) :

$$\gamma_{k_i} = \frac{|a_{k_i}|^2}{2N_0\|\mathbf{w}_{k_i}\|^2}, \tag{9.12}$$

où $\mathbf{w}_{k_i}$ est le vecteur utilisé pour annuler les symboles inconnus $a_{k_{i+1}}, a_{k_{i+2}}, \ldots, a_{k_{n_T}}$. Pour déterminer k_i il faut trouver le SNR moyen de tous les symboles et puis trouver le plus fort.

L'efficacité spectrale de V-BLAST

Si chaque antenne émet un symbole par seconde par hertz de bande passante, l'efficacité spectrale de V-BLAST est donné par :

$$\Phi = b \cdot n_T,$$

ou b a été définit comme le nombre de bits d'information par symbole. Alors, le débit maximal que l'on peut atteindre est $b \cdot n_T \cdot B$, ou B est la bande passante. À titre d'exemple, un système MIMO avec $n_T = 8$, une constellation 16-QAM, et une bande passante de 30kHz, a une efficacité spectrale de 32bps/Hz et un débit de 960kbps.

Capacité, probabilité de coupure et diversité de V-BLAST

Des expressions générales pour la capacité, la probabilité de coupure, et la probabilité d'erreur de V-BLAST, ont démontré être très difficiles à trouver. Cependant, quelques résultats fournissent une justification pour son étude, utilisation et adoption.

Si on suppose que n_T est très grand, en appelant $\alpha = n_T/n_R$ le rapport entre le nombre d'antennes en émission et le nombre d'antennes en réception, la capacité de V-BLAST mesuré en bps/Hz/dimension est donné de façon approximative par :

$$C_v \approx \alpha \cdot \log(1 + \gamma \cdot (\alpha^{-1} - 1). \tag{9.13}$$

La probabilité de coupure P_o est donnée en fonction du débit R et du SNR moyen γ. V-BLAST a comme but de décomposer le canal MIMO en sous-canaux indépendants. Chaque sous-canal subit l'effet du bruit et de l'interférence des autres sous-canaux. La coupure est définie comme l'événement pour lequel le débit est plus grand que la capacité d'un sous-canal quelconque, une fois que l'interférence a été prise en compte.

Soit $r = R/n_T$ le débit de chaque sous-canal. La probabilité de coupure est donc :

$$P_o = 1 - \prod_{i=1}^{n_T} G(i, \gamma, r). \tag{9.14}$$

Il faut remarquer que (9.14) ne tient pas compte du bénéfice obtenu par le choix de l'ordre d'estimation optimal. Dans ce sens, cette performance peut être considérée comme la pire possible.

L'ordre de diversité pour le i-ème canal est $(n_R - n_T + i)$. Ce résultat reste valable indépendamment que l'ordre optimal soit utilisé ou non. Le bénéfice de l'ordre optimal se traduit par une augmentation du SNR moyen de chaque sous-canal. C'est à dire, l'ordre n'a aucun effet sur la pente asymptotique de la probabilité de coupure.

Ces résultats, vus dans l'ensemble, montrent les bénéfices potentiels de V-BLAST est fournissent une justification pour approfondir son étude.

9.4 Considérations pratiques dans une implantation de V-BLAST

9.4.1 Introduction

Implanter V-BLAST pour des applications pratiques, soit sous forme matérielle, soit sous forme logicielle, n'est pas une tâche facile. Parmi les questions à régler on trouve :

- Quel est la meilleure méthode pour trouver la pseudo-inverse de la matrice du canal ? On doit prendre en compte la stabilité numérique de l'algorithme ainsi que sa rapidité ;

  Quelle est la complexité algorithmique de V-BLAST ?
- Quels débits peut-on atteindre avec les processeurs modernes ?
- Quelles sont les opérations les plus complexes que faut-il réaliser ?
- Quelle quantité de mémoire est requise par V-BLAST ?
- Quelle performance en termes de *BLER* peut-on atteindre avec les différentes configurations du système ?
- Quelle est l' influence du choix de L par rapport à la rapidité et la performance de V-BLAST ?

Les réponses à ces questions peuvent varier par rapport aux facteurs, en général indépendants de l'algorithme lui-même, comme la architecture matérielle utilisé, ou bien l'habilité des concepteurs et des outils informatiques utilisés pour optimiser son exécution. Le but dans ce travail n'est pas de se concentrer sur de tels détails d'implantation, mais plutôt d'obtenir des conclusions des propriétés et des demandes inhérentes de l'algorithme. Une telle analyse est d'une importance fondamentale puisqu'elle fournira les critères préalables à la conception d'une implantation matérielle d'un récepteur, très souvent soumis à des contraintes économiques draconiennes. L'utilité de ce type de résultats plus généraux est de pouvoir être utilisés pour faciliter l'implantation de V-BLAST dans n'importe quelles conditions et sous n'importe quelles contraintes.

Plusieurs de ces questions partagent certains aspects : la performance, le débit, la complexité. Ces aspects sont contenus dans la notion de *complexité algorithmique*. Dans ce travail, la complexité est définit tout simplement comme le nombre total d'opérations qu'un algorithme doit réaliser avant de conclure sa tâche.

9.4.2 Calcul de la pseudo-inverse de Moore-Penrose

On a trouvé, par simulation, que l'opération la plus complexe de V-BLAST est le calcul de la pseudo-inverse de Moore-Penrose (*MPPI*) de la matrice du canal $\mathbf{H}$. Il y a plusieurs méthodes pour trouver la MPPI d'une matrice. Quatre d'entre elles sont

étudiés ici, et la moins complexe est identifiée. D'autre part, leur stabilité numérique est évaluée. Les quatre méthodes considérées sont :

i. Décomposition en valeurs singulières (SVD). La SVD est intéressante parce qu'elle a une grande stabilité numérique quand la matrice $\mathbf{H}$ est mal conditionnée. Évidemment, son désavantage est sa grande complexité.

ii. Formule pour MPPI. Quand $(\mathbf{H}^{\dagger}\mathbf{H})^{-1}$ existe, sa MPPI peut être trouvé avec la formule suivante :

$$\mathbf{H}^{+} = (\mathbf{H}^{\dagger}\mathbf{H})^{-1}\mathbf{H}^{\dagger}. \tag{9.15}$$

iii. Décomposition QR mince(Thin QR decomposition). La décomposition QR mince d'une matrice $\mathbf{H}$ est donné par :

$$\mathbf{H} = \mathbf{QR},$$

ou $\mathbf{Q} \in \mathbb{C}^{n_R \times n_T}$ a des colonnes orthonormales, et $\mathbf{R} \in \mathbb{C}^{n_T \times n_T}$ est une matrice triangulaire inférieure. L'un des avantages le plus important de cette décomposition est qu'elle peut être réalisée en utilisant un algorithme très simple, basé sur l'orthogonalization de Gram-Schmidt.

iv. Décomposition QR. Similaire à la QR mince, sauf que $\mathbf{Q} \in \mathbb{C}^{n_R \times n_R}$ est orthonormale ; $\mathbf{R} \in \mathbb{C}^{n_R \times n_T}$ est triangulaire inférieure.

La décomposition QR est plus complexe que QR mince ; son intérêt est que $\mathbf{Q}$ et $\mathbf{R}$ peuvent être mis à jour, sans avoir besoin d'être recalculées, quand $\mathbf{H}$ subit un changement de rang 1.

Résultats sur la stabilité numérique

Aucun problème de stabilité numérique n'a été trouvé en simulation. Comme conclusion, le choix d'une méthode de calcul du MPPI peut être fait basé seulement sur la complexité.

9.4.3 Mesures de complexité

La complexité de chaque opération MPPI a été déterminée par simulation.

Les figures 4.1 à 4.4 montrent les valeurs obtenues pour O_b pour des systèmes MIMO à plusieurs combinaisons d'antennes, utilisant une modulation 16-QAM.

Ces résultats suggèrent que la décomposition QR mince est la méthode la plus attractive pour trouver la MPPI dans une implantation de V-BLAST. En conséquence, tous les résultats qui suivent sont basés sur cette décomposition.

Cependant, il faut remarquer que la complexité est toujours trop grande pour des applications à haut débit.

9.4.4 Complexité de V-BLAST avec QR mince

Comme il a été signalé dans la section précédente, le nombre d'opérations par bit, O_b, donne une idée générale sur la complexité. Dans la suite on verra comment réduire cette complexité.

Cette section a pour but de classer les opérations requises par V-BLAST.

Les types suivants d'opérations sont proposés. Chaque type d'opération est identifié avec un symbole.

- Additions
  - · Additions faites pendant une multiplication de matrices (A_{mm})
  - · D'autres additions (A_o)
- Multiplications
  - · Multiplications faites pendant une multiplication de matrices (P_{mm})
  - · D'autres multiplications (P_o)
- Divisions (D_o)
- Racines carrées (S_o)
- Opérations de mémoire (écriture ou lecture)
  - · Opérations de mémoire faites pendant une multiplication de matrices (M_{mm})
  - · Opérations de mémoire faites pendant une copie de matrices (M_{cm})
  - · Opérations de mémoire faites pendant une opération de remise à zéro d'une matrice (M_{zm})
  - · D'autres opérations de mémoire (M_o)

O_b est définit comme :

$$O_b = A_{mm} + A_o + P_{mm} + P_o + D_o + S_o + M_{mm} + M_{cm} + M_{zm} + M_o.$$

Il faut remarquer que ceci n'est qu'un classement possible des opérations réalisées par V-BLAST.

Dans les tables 4.1 à 4.4 chaque type d'opération est donnée comme un pourcentage d'O_b, pour $L = 10$. Ces résultats sont un premier pas vers l'identification des types d'opérations qui devraient être optimisées en premier terme.

9.4.5 Le rôle de L dans la complexité de V-BLAST

V-BLAST est divisé en deux parties, la *phase de préparation* et la *phase d'estimation des symboles*. La phase de préparation est réalisée une fois par bloc, et la phase d'estimation des symboles est répétée L fois. Bien évidemment, une réduction de L entraîne une diminution de la complexité du récepteur, mais au même temps une augmentation de *BLER*. Cette section présente quelques aspects du comportement de V-BLAST en fonction de L.

Soit O_{setup} le nombre d'opérations requises pendant la phase de préparation, et O_{est} le nombre d'opérations requises pendant la phase d'estimation des symboles ($O_b = O_{setup} + O_{est}$). Le *BLER* est donné par :

$$BLER \approx 1 - (1 - BER)^B, \tag{9.16}$$

où $B = b \cdot n_T \cdot L$ est le nombre de bits d'information par bloc. La valeur minimal possible de L est 1, ce qui veut dire que *BLER* a une borne inférieure donnée par $1 - (1 - BER)^{b \cdot n_T}$.

Pour un n_T donné, O_b augmente de façon linéaire avec n_R. Soient $O_{setup} = x \cdot n_R$ et $O_{est} = y \cdot n_R$, pour x, y des constantes qui dépendent de l'implantation de V-BLAST. Alors,

$$O_b \approx \frac{O_{setup} + L \cdot O_{est}}{B} \tag{9.17a}$$

$$\propto n_R \cdot \left(\frac{x}{L} + y\right). \tag{9.17b}$$

À partir de l'équation (4.4b) on peut voir qu'une augmentation de L entraîne une diminution (inversement proportionnelle) de O_b. La figure 4.5 montre le comportement de O_b pour plusieurs combinaisons du n_R et L.

Il est intéressant de remarquer comment le *BER* provoque une amélioration du *BLER* en fonction de O_b. En utilisant (4.3) et (4.4a) on peut trouver que :

$$BLER(O_b) \approx 1 - (1 - BER)^{\frac{n_R(x+Ly)}{O_b}}. \tag{9.18}$$

La figure 4.6 montre le comportement du *BLER* comme fonction de O_b. On s'aperçoit qu'après un certain seuil, une augmentation de la complexité du système n'entraîne pas une amélioration de son taux d'erreur.

La figure 4.6 suggère que, pour un système (n_T,n_R) et une implantations particulière de V-BLAST, il y a ce que l'on pourrait appeler un « point d'opération optimal » par rapport à la performance en *BLER* souhaitée et en fonction du choix de L.

9.4.6 Résultats de simulation

La figure 4.7 montre l'effet de L sur V-BLAST pour un système (4,4). La forme de la courbe de performance obtenue est en correspondance avec la performance prédite par l'équation (9.17). Quand L augmente, O_b a une tendance asymptotique vers O_{setup}.

Résultats de simulation sur le taux d'erreur de V-BLAST

Dans cette section, les résultats de simulation sont présentés concernant le *BLER* en fonction du SNR moyen. Les figures 4.8 à 4.11 montrent ces résultats de performance pour plusieurs combinaisons des antennes, largeur du bloc $L = 10$, et

modulation 16-QAM. On peur arriver à deux conclusions importantes. La première est que les récepteurs V-BLAST symétriques (n_T,n_T) ont un *BLER* très mauvais par rapport aux configurations asymétriques. Une autre conclusion intéressante est qu'une diminution de n_R a un effet très prononcé sur la pente des courbes. Ça veut dire qu'il y a augmentation de la diversité du système. Ceci confirme le résultat selon lequel la diversité de V-BLAST augmente avec $n_R - n_T$. L'effet est moins prononcé pour des valeurs importantes de n_R.

Tenant compte de la très grande efficacité spectrale fournit par V-BLAST, ces résultats sont extrêmement intéressants. Ils indiquent que V-BLAST a le potentiel d'être un candidat dans la recherche de solutions pour améliorer les communications sans fils.

9.4.7 Résultats expérimentaux

L'algorithme V-BLAST a été implanté sur un processeur de signaux numériques (*DSP*) en virgule flottante TMS320C6711 qui opère à 150MHz et qui est capable de fournir jusqu'à 900 MFLOPS. La complexité de l'algorithme a été mesuré en cycles d'instruction en utilisant un *profiler* d'exécution de code. Comme le temps de cycle d'instruction est connu et égal à $I_c = 6.7$ns, il est possible d'estimer le rendement du DSP dans la pratique.

Le tableau 4.5 montre les cycles d'instruction qui sont requises par chaque bit d'information reçu, et le débit correspondant en kbps, quand le DSP exécute l'algorithme V-BLAST pour un système MIMO (2,3).

La figure 4.12 montre les cycles d'instruction requises par le même DSP en fonction de la longueur du bloc L. Le comportement général prédit par la figure 4.6 et l'équation (9.17) sont confirmés par l'expérience.

Même un DSP, courant l'état de l'art, est loin d'être capable de fournir un débit équivalant a ETHERNET (10Mbps) pour le petit système MIMO (2,3) considéré. Ces résultats montrent clairement qu'il faut réduire le nombre d'opérations requises par V-BLAST si on veut appliquer cette technique dans un système commercial.

9.5 V-BLAST comme un problème de moindres carrés

9.5.1 Introduction

La décomposition QR mince peut être utilisé pour réduire la taille de V-BLAST de $(n_R \times n_T)$ à $(n_T \times n_T)$, ce qui donne une réduction potentielle de la complexité. Soit $\mathbf{H} = \mathbf{QR}$, où $\mathbf{Q} \in \mathbb{C}^{n_R \times n_T}$ est une matrice avec des colonnes orthonormales, et $\mathbf{R} \in \mathbb{C}^{n_T \times n_T}$ est triangulaire supérieure. Alors, le problème des moindres carrés, c'est à dire, trouver le vecteur $\mathbf{x}$ qui minimise $\|\mathbf{Hx} - \mathbf{b}\|^2$, est équivalent au problème

de trouver $\mathbf{x}$ qui minimise $\|\mathbf{Rx} - \mathbf{Q}^{\dagger}\mathbf{b}\|^2$. Comme $\mathbf{R}$ est triangulaire, ce problème est résolu facilement.

Dans cette thèse, on propose une utilisation de cette technique d'une façon nouvelle : elle est utilisée dans les deux phases de V-BLAST pour accélérer le calcul des pseudo-inverses et aussi de l'estimation des symboles sans pourtant augmenter le taux d'erreur. L'algorithme modifié est nommé dans la suite *V-LS*.

La version de V-BLAST, qui est probablement la plus rapide connue, est nommé ici *V-SQR*. Elle utilise aussi la décomposition QR mince, mais d'une façon sous-optimale pour éviter le calcul des pseudo-inverses.

Dans ce chapitre, ces trois algorithmes sont comparés. Il a été trouvé que, sous certaines conditions, V-LS est plus rapide que V-SQR. Des scénarios où le taux d'erreur de V-SQR est considérablement dégradé, sont aussi décrits.

9.5.2 Décomposition QR dans V-BLAST

V-BLAST peut être considéré comme un algorithme pour résoudre un système d'équations linéaires qui ont été perturbés par le bruit. La solution du système doit appartenir à la constellation S. Si le vecteur de bruit $\mathbf{n}$ est zéro, et le vecteur reçu est $\mathbf{r}$ (voir équation (2.1)), alors V-BLAST donne la solution de moindres carrés au problème de trouver $\hat{\mathbf{a}}$ tel que $\|\mathbf{H}\hat{\mathbf{a}} - \mathbf{r}\|^2$ a une norme minimale.

En présence du bruit, V-BLAST ne trouve pas toujours la meilleur solution dans le sens de moindres carrés ; par contre, il a une plus basse complexité que d'autres méthodes qui trouvent de meilleur solutions.

L'idée consiste à utiliser la décomposition QR pour modifier le système $\mathbf{r} = \mathbf{Ha} + \mathbf{n}$, de la façon suivante :

$$\begin{aligned} \mathbf{r} &= \mathbf{QRa} + \mathbf{n} \\ \tilde{\mathbf{r}} &= \mathbf{Ra} + \tilde{\mathbf{n}}. \end{aligned} \tag{9.19}$$

Les statistiques du bruit ne sont pas changées par cette opération, parce que les colonnes de $\mathbf{Q}$ sont orthonormales. En utilisant ces idées, un nouvel algorithme, nommé V-LS, est proposé. V-BLAST et V-LS produisent exactement les mêmes résultats.

9.5.3 L'algorithme QR ordonné

Si la décomposition QR est utilisé comme décrit par (5.1), $\hat{\mathbf{a}}$ peut être trouvé d'une façon itérative. Par exemple, $\hat{a}_{n_T}$ est donné par :

$$\hat{a}_{n_T} = f_q\left(\frac{\tilde{r}_{n_T}}{R_{n_T,n_T}}\right),$$

et, en général, pour $1 \leq i < n_T$,

$$\hat{a}_i = \frac{\tilde{r}_k - \hat{d}_k}{R_{k,k}},$$

où le terme $\hat{d}_k$ peut être considéré comme de l'interférence ; il est déterminé par :

$$\hat{d}_k = \sum_{j=k+1}^{n_T} R_{k,j}\hat{a}_j.$$

L'idée fondamentale du V-SQR est que la maximisation du SNR à chaque pas, comme fait par V-BLAST, est équivalent à ordonner les éléments diagonaux de $\mathbf{R}$ $R_{k,k}$ du minimum au maximum. Alors, V-SQR fournit une procédure pour construire une telle matrice $\mathbf{R}$. Cette procédure entraîne une permutation des colonnes de $\mathbf{H}$.

9.5.4 Analyse de la complexité et résultats de simulation

Comparé à V-BLAST, la phase de préparation de V-LS réalise une décomposition QR supplémentaire. Par contre, les pseudo-inverses sont réalisées sur une matrice $n_T \times n_T$. Quant à la phase d'estimation des symboles, on doit calculer une multiplication vecteur-matrice supplémentaire, mais toutes les autres opérations dépendent seulement de n_T et pas de n_T et n_R.

Quant à V-SQR, il réalise une décomposition QR modifiée qui est plus complexe que la décomposition typique, parce qu'on doit permuter les colonnes de $\mathbf{Q}$ et $\mathbf{R}$ plusieurs fois. Aussi, les symboles estimés doivent être permutés pour les restituer l'ordre correct.

L'effet de la longueur du bloc L sur les algorithmes est étudié. Comme V-LS et V-SQR ont une structure identique à celle de V-BLAST, on espère que l'équation (9.17) reste toujours valide.

Du point de vue de la complexité, un aspect important de V-LS et V-SQR est que seulement deux parmi les opérations réalisées dépendent de n_R. Ces opérations sont la décomposition QR mince initiale de $\mathbf{H}$ et le calcul de $\mathbf{x} = \mathbf{Q}^\dagger\mathbf{r}$. Pour cette raison, il est intéressant d'étudier comment leur complexité dépend de n_R pour de n_T constant. Aussi, les taux d'erreur par bloc de ces deux algorithmes sont comparés.

9.5.5 Taux d'erreurs par bloc

Les figures 5.1 à 5.4 montrent le taux d'erreur *BLER* de V-LS et V-SQR.

On peut conclure que, pour des valeurs de n_R plus grandes que n_T, le taux d'erreur par bloc de V-SQR est similaire à celui de V-BLAST. Pour n_T proche à n_R, pourtant, la différence devient significative.

Complexité comme fonction de L

Le comportement de O_b en fonction de L a été étudié. La contribution de la phase de préparation du bloc à la complexité totale est réduite si L augmente, alors que

celui de la phase d'estimation des symboles augmente. Les résultats sont montrés dans les figures 5.5 à 5.7.

Comme conclusion, l'avantage de complexité présenté par V-SQR se voit réduit si L augmente, avec un effet plus prononcé si n_R est grand. Les raisons de cela sont la complexité ajouté par la mise en ordre des symboles dans la phase d'estimation (opération qui n'existe pas en V-LS), et l'importance réduite de la phase de préparation si L est grand.

Complexité en fonction de n_R

Pour n_T constant, on a déterminé le comportement de O_b en fonction de n_R. Les résultats sont montrés dans la figure 5.8.

C'est évident que V-LS et V-SQR ont une complexité qui est significativement plus petite que celle de V-BLAST. Cependant, quand n_R est suffisamment grand, V-LS est moins complexe que V-SQR.

La différence dans la pente de O_b entre V-BLAST et les deux autres techniques qui sont présentées dans la figure 5.8 est remarquable ; la raison de cette différence est l'influence moindre de n_R sur la plupart des opérations réalisées par ces deux algorithmes.

Chaque algorithme, V-LS et V-SQR, a des avantages et désavantages. Le tableau 5.1 présente un résumé des circonstances où chaque algorithme est une meilleur choix que l'autre.

9.5.6 Résultats expérimentaux

Comme il a été décrit avant, V-LS a été implanté dans un DSP TMS320C6711 pour différents valeurs de L, avec $n_T = 2$ et n_R 3, 6 et 23. Les résultats confirment les prédictions : la complexité diminue avec L et V-LS est substantiellement plus rapide que V-BLAST pour des n_R grands. Ces résultats son montrés dans les figures 5.9 à 5.11.

9.6 Décodage de réseau de points dans le contexte des codes espace-temps verticaux

9.6.1 Introduction

Les algorithmes basées sur V-BLAST ont montré être très intéressants, performants et leur complexité et taux d'erreur ont été étudiés. Cependant, V-BLAST fait un compromis entre performance et complexité. Pour mieux comprendre ce que les codes verticaux sont capables de fournir, il faut étudier de récepteurs capables de mieux exploiter la diversité.

On profite de cet étude pour répondre aux questions suivantes : combien on gagne en taux d'erreur si on utilise des récepteurs plus performants ? Et, combien on gagne en complexité par rapport à V-BLAST ?

Ces questions seront répondues pour des constellations de signaux qui peuvent être représentées comme une structure connue comme un réseau de points.

D'un point de vue géométrique, un réseau de points est un ensemble de points à dimension n qui est régulier, périodique, et infini. La figure 6.1 montre un réseau de points en deux dimensions, un point $\mathbf{r} \in \mathbb{R}^2$ est le point du réseau le plus proche à $\mathbf{r}$.

D'une façon plus formelle, soit $\mathbf{G}$ une matrice réelle, avec n files et m colonnes, ou les files sont linéairement indépendantes et $n \leq m$. Le réseau de points généré par $\mathbf{G}$ est défini comme l'ensemble de vecteurs :

$$\Lambda(\mathbf{G}) = \{\mathbf{uG} \colon \mathbf{u} \in \mathbb{Z}^n\}. \tag{9.20}$$

La matrice $\mathbf{G}$ est appelée la matrice génératrice de Λ. Dans un réseau Λ, soit un vecteur $\mathbf{x} \in \mathbb{R}^m$, le problème de trouver un vecteur $\hat{\mathbf{c}} \in \Lambda$ tel que :

$$\| \mathbf{x} - \hat{\mathbf{c}} \| \quad \leq \quad \| \mathbf{x} - \mathbf{c} \| \qquad \forall \mathbf{c} \in \Lambda,$$

ou $\| \cdot \|$ est la norme ℓ^2, est appelé le *problème du point le plus proche*. Dans le contexte du codage de canal, ce problème est semblable à celui du décodage à maximum de vraisemblance. Quand les mots de code peuvent être représentés comme des points d'un réseau de points, les décodeurs de réseau sont plus rapides que les décodeurs ML généraux.

Comme ils ont une importance fondamentale dans plusieurs applications, les décodeurs de réseau sont bien connus. Des algorithmes très efficaces et performants ont été développés. On peut profiter de ces connaissances quand on utilise ce type d'algorithmes pour les codes verticaux.

L'utilisation des décodeurs de réseau dans ce contexte n'est pas directe. On propose de modifications pour rendre ces décodeurs utilisables pour des constellations QAM. Une limitation de ce type d'algorithmes quand ils sont utilisés pour décoder de constellations finies est étudié et résolu d'une façon nouvelle.

9.6.2 L'utilisation des décodeurs de réseau dans le contexte V-BLAST

Pour utiliser un décodeur de réseau de points avec un système MIMO, les problèmes suivants doivent être résolus :

- comme S_M n'est pas, en général, un réseau de points, une représentation en réseau convenable doit être trouvé.
- un *mapping* entre la représentation en réseau de S_M et S doit être trouvé, parce que finalement le récepteur doit produire des estimations de $\mathbf{a}_m \in S'$.

Comment représenter S_M en réseau de points ?

Avant tout, une représentation réelle du système doit être trouvé. Un système MIMO peut être représenté par un modèle purement réel ; dans l'équation (2.2), le système est modélisé par :

$$\tilde{\mathbf{r}} = \tilde{\mathbf{H}}\tilde{\mathbf{a}}_{\mathbf{m}} + \tilde{\mathbf{n}},$$

où tous les termes sont réels.

Une fois que le système a été réduit à un modèle purement réél, le récepteur doit construire une matrice génératrice $\mathbf{G}$ à partir de $\mathbf{H}$, et un vecteur $\mathbf{x}$ à partir de $\mathbf{r}$, tels que :

$$\mathbf{G} = 2e_1 \begin{bmatrix} \Re(\mathbf{H}) & \Im(\mathbf{H}) \\ -\Im(\mathbf{H}) & \Re(\mathbf{H}) \end{bmatrix}^T,$$

$$\mathbf{x} = \tilde{\mathbf{r}} + (2n-1)e_1\mathbf{G}.$$

Maintenant le décodeur de réseau peut opérer sur $\mathbf{G}$ et $\mathbf{x}$, pour trouver un vecteur $\hat{\mathbf{u}}$ tel que $\hat{\mathbf{u}}\mathbf{G}$ soit le point du réseau le plus proche à $\mathbf{x}$.

Le *mapping* de $\hat{\mathbf{u}}$ vers S

Une conséquence importante d'utiliser un décodeur de réseau de points et non pas un algorithme ML général est que le vecteur $\hat{\mathbf{u}}$ est un élément du réseau Λ_S, mais pas nécessairement de la constellation S_T. Spécialement pour des faibles rapports SNR, il y a une grande probabilité que le point reçu se trouve en dehors du code, et dans ce cas l'estimation faite par l'algorithme n'appartiendra pas au code.

Il y a quelques méthodes «classiques» pour résoudre cette situation. L'option la plus simple consiste à déclarer un effacement, ce qui a le coût d'augmenter le taux d'erreur. Une autre action possible est d'identifier quand cette situation apparaît, et alors obliger l'algorithme à chercher dans une région plus grande jusqu'à ce qu'un point de la constellation soit trouvé. Cette méthode a, bien évidemment, le très gros désavantage d'augmenter de façon très significative la complexité de l'algorithme.

Une troisième méthode consiste à projeter le vecteur reçu sur la frontière de la constellation avant de chercher le point le plus proche. En outre de n'être pas optimal, cette procédure a, elle aussi, le désavantage d'être beaucoup plus complexe.

On a proposé et étudié une nouvelle méthode pour résoudre ce problème. Même si elle est sous-optimale, elle a l'avantage d'avoir un effet négligeable sur la complexité du récepteur, et au même temps un taux d'erreur moins élevé que celui de la méthode d'effacements.

La méthode proposé consiste à quantifier chaque élément de $\hat{\mathbf{u}}$ qui se trouve dehors du code au point de la constellation le plus proche.

9.6.3 Un algorithme de décodage de réseau de points adapté à V-BLAST

L'algorithme de décodage de réseau de points utilisé est basé sur l'algorithme nommé *Closest Point* [34]. Ce décodeur est le plus rapide qui soit connu. Il a été conçu pour des réseaux sans structure particulière. Au coeur de *Closest Point* se trouve l'algorithme *Decode*, qui trouve le point le plus proche à $\mathbf{x_3}$ dans $\Lambda(\mathbf{H_3}^{-1})$.

La complexité de la procédure *Decode* dépend de la longueur des lignes de $\mathbf{G}$. Pour accélérer son opération, on peut réaliser une réduction de la base du réseau. Les réductions les plus efficaces et les plus répandues son LLL et Korkine-Zolotareff ; elles peuvent réduire le temps d'exécution de *Decode* d'une façon significative, même si elles sont très complexes.

On a modifié l'algorithme pour décoder des codes espace-temps verticaux ; le nouvel algorithme est appelé *V-CP*.

9.7 Considérations pratiques sur l'utilisation d'un décodeur de réseau de points dans un système MIMO

9.7.1 Introduction

On présente des résultats qui permettent de comparer la complexité et la performance de trois algorithmes de réception : V-LS, V-CP sans réduction de base du réseau, et V-CP avec réduction LLL.

Les taux d'erreur par bloc et par bit de V-LS et V-CP sont comparés pour déterminer la sous-optimalité de V-LS. Les systèmes avec $n_R \gg n_T$ sont particulièrement intéressants, parce que V-BLAST a démontré être capable d'exploiter d'une façon efficace la diversité spatiale dans cette situation.

Il est important de remarquer que, dans [34], les résultats sur la complexité ne tiennent pas compte de la complexité ajoutée par la réduction de la base du réseau. Ce fait surprenant, veut dire que, même s'il a été prouvé que *Decode* est plus rapide quand la matrice génératrice est réduite, il n'a été prouvé que la complexité totale (réduction plus recherche des points) soit en fait réduite. On présente les résultats sur cette question.

En outre, la méthode utilisée dans V-CP pour résoudre le problème des points qui se trouvent en dehors du code est comparée avec décodage basé sur le critère de maximum de vraisemblance. Il est aussi comparé avec la méthode des effacements.

9.7.2 Comparaison des taux d'erreur

On a comparé les taux d'erreur de V-LS et V-CP. Les figures 7.1 à 7.4 montrent une comparaison entre les taux d'erreur par bloc, et les figures 7.5 à 7.8 comparent les taux d'erreur par bit.

Quelques observations intéressantes peuvent être dérivées de ces figures. La première est que pour $n_R = 2n_T$ et $BLER = 10^{-2}$, la différence entre V-LS et V-CP est comprise entre un demi décibel et moins d'un décibel. Cela justifie le fait que V-LS est vraiment très bon pour exploiter la diversité spatiale.

Cependant, quand $n_T \approx n_R$ la situation est différente. Le décodage du réseau de points fournit un taux d'erreur évidemment supérieure à celui de V-LS.

V-CP réalise une estimation conjointe de tous les symboles. Dans un certain sens, cela veut dire que la diversité du système est la même pour tous les éléments du vecteur reçu. Il suffit de comparer la pente des courbes de *BLER* pour les systèmes (n,n). Quand on utilise V-LS, la courbe indique que le récepteur ne peut pas exploiter la diversité. Une courbe de ce type est presque une ligne droite. Par contre, la pente des courbes de V-CP augmente avec le SNR moyen, ce qui indique que le récepteur est capable d'exploiter la diversité.

Des résultats pour *BER* sont présentés dans les figures 7.5 à 7.8. Le même comportement est observé pour le *BLER*.

9.7.3 Comparaisons de complexité

Les figures 7.9 à 7.12 montrent une comparaison entre la complexité de V-LS est celle de V-CP, sans avoir réalisé une réduction de la base du réseau.

O_b est présenté en fonction du SNR moyen, parce que, à différence de ce qui se passe avec V-BLAST, la complexité de V-CP n'est pas constante mais change avec la puissance du bruit. La raison est que pour des puissances de bruit importantes, il y a une probabilité plus grande que le vecteur reçu se trouve loin d'un point quelconque du code. Dans ce cas, l'algorithme doit chercher un ensemble plus important de points.

Il y a plusieurs conclusions très intéressantes que l'on peut tirer à partir des résultats présentés dans ces figures.

La première, et peut être la plus importante, est que la complexité de V-CP pour des systèmes plus petits peut être significativement plus grande que pour des systèmes plus grands. Les courbes de complexité montrent que les systèmes les plus grands n'ont une complexité plus grande qu'à partir d'une certaine valeur du SNR moyen.

Seulement une hypothèse est aventurée ici sur la cause possible d'un tel comportement. La taille de l'ensemble de points à chercher augmente quand la puissance du bruit augmente. Aussi, la taille diminue quand la dimension du réseau augmente. Cependant, une fois que la puissance du bruit a atteint une certaine valeur, la taille de l'ensemble est dominée par la dimension du réseau.

La raison pour laquelle la taille de l'ensemble augmente avec la puissance du bruit est que le point reçu est, avec une plus grande probabilité, localisé autour d'une grande quantité de points du réseau, et aucun d'entre eux est particulièrement proche de lui. Pour cette raison, l'algorithme doit examiner plusieurs points. Cet effet est opposé par la dimension du réseau ; quand la dimension augmente, le volume occupé par un nombre constant de points du réseau augmente aussi, ce qui en effet sépare les points, et fournit de l'immunité par rapport au bruit.

Quand la puissance du bruit est réduite, le point reçu est, avec grande probabilité, autour d'un point du réseau, qui sera trouvé rapidement si l'algorithme est efficace. Le bruit a un rôle moins important dans ce cas, et l'effet qui détermine la taille de l'ensemble de points est la dimension du réseau.

L'énorme complexité de V-CP pour des systèmes symétriques doit être remarquée aussi. La figure 7.12 est très illustratrice.

La figure 7.12 est un exemple d'un autre phénomène qui a été remarqué. Il y a un pic de complexité pour le système (8,8) à 21dB. Ces pics apparaissent de façon aléatoire ; une nouvelle simulation avec les mêmes conditions n'aura pas probablement de pics. L'explication la plus probable (à confirmer) est que certaines réalisations du canal forcent l' algorithme V-CP à prendre beaucoup plus de temps que le normal.

La complexité de V-CP n'a pas, apparemment, de bornes claires. Ceci pourrait être un gros problème pour son implantation, parce que la plus part des applications ont besoin d'un débit constant, et en conséquence, une complexité fixe ou, au moins, bornée. Il y a deux possibles solutions à ce problème. La première est de forcer V-CP à terminer après avoir cherché pendant un certain temps. Ceci aura un impact sur le taux d'erreur.

Une deuxième possibilité est d'utiliser la réduction LLL. Les pics de complexité n'ont pas été observés quand la matrice génératrice du réseau a été réduite. D'un coté, ceci donne du support à l'hypothèse que les pics sont provoquées par des matrices problématiques, parce que la réduction LLL donne de la stabilité à l'algorithme *Decode*. D' autre part, l'utilisation de la réduction n'a pas un effet positif sur la complexité du récepteur.

Dans un certain sens, ce comportement de la complexité veut dire qu'elle peut être modélisée comme une variable aléatoire. La complexité requise pour décoder un vecteur dépend de la matrice du canal et des valeurs du bruit. Les résultats présentés ici sont donc des valeurs moyennes. Pour mieux comprendre le comportement de V-CP il faudrait obtenir des estimateurs de second ordre de la complexité.

Il est aussi intéressant de constater que la complexité de V-CP a un comportement asymptotique quand le SNR moyen augmente.

Complexité de la réduction LLL

La complexité de V-CP avec et sans la réduction LLL de la matrice génératrice a été étudiée. Les résultats présentés sont ceux de la complexité totale du récepteur :

la réduction plus le traitement des vecteurs reçus. Les figures 7.14 à 7.17 montrent les résultats pour $L = 10$.

Les résultats sont clairs : pour la plupart de cas, l'augmentation de rapidité de *Decode* est tout à fait insuffisante pour compenser l'augmentation de la complexité provoqué par la réduction LLL.

La réduction LLL augmente la complexité de la phase de préparation de l'algorithme et elle diminue celle de la phase d'estimation. Pour cette raison, le plus grand bénéfice est obtenue quand une grande quantité de vecteurs sont traités par bloc. Les figures 7.18 à 7.21 sont similaires à celles présentées plus haut sauf que le largeur de bloc a été augmenté à 100.

Dans la plupart de cas, la réduction LLL est si coûteuse qu'elle termine par augmenter la complexité totale du récepteur, même pour $L = 100$.

En bref, le seul bénéfice évident de la réduction LLL est la suppression des pics de complexité.

9.7.4 V-CP et maximum de vraisemblance comparés

L'algorithme V-CP est comparé avec un récepteur basé sur le critère à maximum de vraisemblance (ML) afin d'évaluer la perte en performance occasionnée par la méthode proposée d'adaptation des algorithmes décodeurs des réseau de points au problème des codes verticaux. V-CP est comparé aussi à un autre algorithme, nommé *V-Fixed*, qui tout simplement réalise une assignation d'un point prédéterminé de la constellation à chaque fois que le point estimé par *Decode* n'appartient pas à la constellation.

Les figures 7.22 à 7.27 montrent les taux d'erreur obtenus.

Il a été déterminé que, dans tous les cas, V-CP est supérieur à l'algorithme V-Fixed. Aussi, il a été déterminé que V-CP devient très proche de ML si n_R augmente par rapport à n_T.

9.7.5 Commentaires

L'affirmation qui dit que les algorithmes basés sur V-BLAST sont toujours plus rapides, pendant que les décodeurs de réseau ont de meilleures taux d'erreur, a été montré n'être pas complètement vraie.

L'élection d'un algorithme ou d'un autre dépend sur les détails spécifiques d'implantation. Si le taux d'erreur doit être minimisé à n'importe quel prix, alors V-CP est évidemment le meilleur choix. Dans des situations plus réalistes il y a des autres facteurs à considérer. Par exemple, on doit considérer les valeurs de SNR moyen auxquels le récepteur devra fonctionner. Un autre facteur est la variance de la complexité qui peut changer d'un vecteur reçu à un autre et d'un bloc à un autre.

Enfin, la méthode utilisée par V-CP pour réaliser le *mapping* des points en dehors de la constellation a été justifié.

9.8 Conclusions

Au début, l'objectif de cette thèse était d'explorer la performance et la complexité des systèmes de communications MIMO basées sur les codes espace-temps en couches. Après avoir analysé, implanté, simulé et comparé un grand nombre d'algorithmes, c'est le moment de tirer quelques conclusions.

Les analyses de complexité présentées sont importantes, ou au moins utiles, parce que elles fournissent l'information nécessaire pour réaliser d'implantations pratiques dans le futur.

Des comportements intéressants et pas évidents ont été observés ; ils ont été expliqués. Aussi, différentes situations on été identifiées et le meilleur algorithme a été choisi.

Des modifications, nouvelles idées, et améliorations des algorithmes connus ont été proposées. Des techniques de moindres carrés ont été utilisés pour développer V-LS, une variante de V-BLAST. Le décodage des réseaux de points a été adapté aux codes verticaux avec succès.

Appendix A

Example of a space-time block code

THE FOLLOWING example, taken from [38] (and originally proposed in [11]), clarifies the decoding process of space-time block codes ($STBC$). Consider a (2,1) system where the vector $\mathbf{a} = (a_1, a_2)$ is mapped to matrix $\mathbf{A}$ given by:

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ -a_2^* & a_1^* \end{bmatrix}$$

Note that $\mathbf{A}$ is an orthogonal matrix. The first row of $\mathbf{A}$ is transmitted at time t_1 and the second at time t_2. The received signals at times t_1 and t_2 are, respectively:

$$r_1 = h_1 \cdot a_1 + h_2 \cdot a_2 + n_1$$

and

$$r_2 = -h_1 \cdot a_2^* + h_2 \cdot a_1^* + n_2$$

where a^* is the conjugate of a, the channel matrix $\mathbf{H} = (h_1, h_2)^T$ and n_1, n_2 are noise samples as defined in section 1.1. In other words, the system model (2.1) is applied to the rows of matrix $\mathbf{A}$, and the block length $L = 2$. Let $\mathbf{r} = (r_1, r_2^*)^T$, $\mathbf{n} = (n_1, n_2)^T$, $\mathbf{a} = (a_1, a_2)^T$, and $\mathbf{H} = \left(\begin{smallmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{smallmatrix}\right)$; then this system can be described by the following equation:

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n}$$

Let $\mathcal{C}$ be the set of all possible realizations of vector $\mathbf{a}$; then, the maximum-likelihood decoding rule is

$$\hat{\mathbf{a}} = \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmin}} \|\mathbf{r} - \mathbf{H}\mathbf{c}\|^2 \tag{A.1}$$

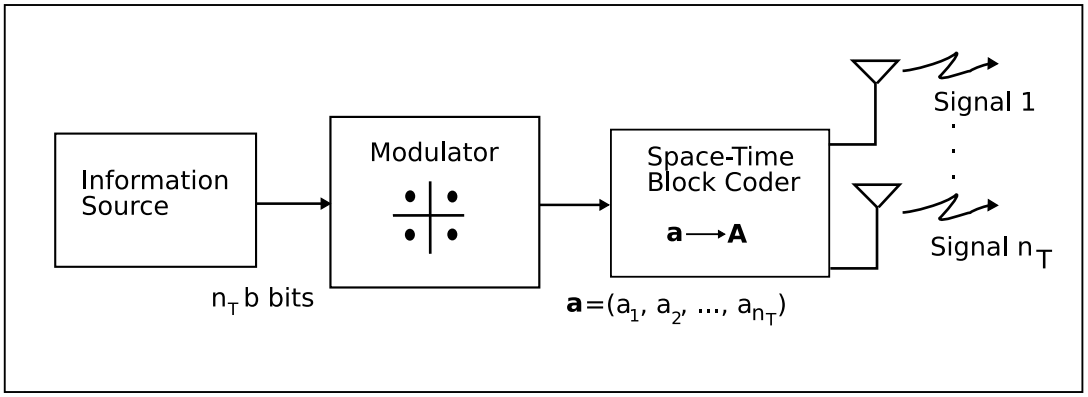

Figure A.1: Block diagram of a space-time block encoder. $n_T b$ information bits are modulated in groups of b bits into vector $\mathbf{a}$. This vector is used to construct codeword matrix $\mathbf{A}$, whose rows are then transmitted one at a time.

Taking advantage of the fact that $\mathbf{H}\mathbf{H}^\dagger = \rho\mathbf{I}$, the received signal can be modified to $\tilde{\mathbf{r}} = \mathbf{H}^\dagger\mathbf{r} = \mathbf{H}^\dagger\mathbf{H}\mathbf{a} + \mathbf{H}^\dagger\mathbf{n}$. Since $\mathbf{H}$ is orthogonal it does not change the properties of the noise in any relevant way, and the decoding rule A.1 can be simplified to

$$\hat{\mathbf{a}} = \underset{\mathbf{c}\in\mathcal{C}}{\operatorname{argmin}} \|\tilde{\mathbf{r}} - \rho\mathbf{c}\|^2 \tag{A.2}$$

This simplification is the principal advantage of space-time block codes. Unfortunately, constructing orthogonal matrices is not a trivial task and not many STBCs are known.

Appendix B

A simulator platform for MIMO systems

Computer simulation is an essential tool when studying the performance and complexity of the different receivers that have been proposed for use in MIMO telecommunication systems. Simulation also has an important role to play in architecture exploration, indicating which portions of the algorithms need to be optimized, and where the bottlenecks are found.

As part of the work of this thesis, a simulator platform, called MIMOSIM, has been written to obtain all the complexity and performance results presented in this thesis.

This simulator has been created from scratch in order to have the flexibility, granularity and speed that are difficult to obtain with generic simulators or with tools like MATLAB™.

MIMOSIM is divided into modules, which are called, in order, from a master controller module. First, configuration is carried out from a text file that contains the commands to be executed; then, the remaining modules are executed in a loop until a certain condition (typically, a number of bit errors) is met.

The modules present at this time are: source, coder, modulator, channel, channel estimator, receiver, demodulator, error counter, and a module that reports all the results. Each module's capabilities can be easily expanded.

At present, the modules are functional enough to create most typical simulation scenarios. MIMOSIM can cycle through a range of parameters in a single simulation; some of the main parameters that can be configured are:

- Selection of range of block sizes
- Selection of range of average SNR
- Type and average energy of constellation
- Number of errors to simulate

- Selection of range of receiver modules to simulate

MIMOSIM also produces extensive reports of all performance and complexity measurements. These reports are arranged in such a way that producing graphics from them, using tools such as gnuplot or a spreadsheet, is very easy.

Among the results provided are: BER, BLER, true average SNR, and detailed complexity measurements, which include counts of all arithmetic and memory operations performed. Extensive debug information can be produced on demand, including intermediate values of calculations.

Most of the debugging and complexity functions can be disabled, if simulation speed is a priority.

The MIMOSIM simulator can be found, along with its documentation, at:

http://www.comelec.enst.fr/~rodrigez/msim/index.html.

Appendix C

Papers published and submitted for publication

DURING the development of this thesis, four papers were written and submitted for publication. They are reproduced in this appendix. The first paper reproduced here is titled "Towards Real-Time V-BLAST" and was submitted to the IEEE Communications Letters. "A Simulator Platform for MIMO Systems" was submitted to the International Symposium on Personal, Indoor and Mobile Radio Communications. The paper titled "Least-Squares Techniques Applied to V-BLAST" was submitted to the World Conference on Networks and Communications. Finally, "Performance Study of Space-Time Communications Systems Based on the V-BLAST Algorithm" was submitted to the International Conference on Electronics, Communications, and Computers.

Towards Real-Time V-BLAST

L. M. Bazdresch and J. Rodríguez-Guisantes.

-

Abstract: V-BLAST is a sub-optimal algorithm designed for MIMO communications systems with quasi-static fading. It offers unprecedentedly high spectral efficiency and conceptual simplicity. It is burdened, however, by the need to calculate the Moore-Penrose pseudo-inverse of the channel matrix. In this letter it is confirmed, using simulation, that more than 95% of the number of operations performed by V-BLAST are spent calculating the pseudo-inverse. Then, two modifications to V-BLAST that reduce its complexity by a factor of approximately 1.5 times the block length are proposed. The trade-off is slightly larger memory requirements. There is no loss in bit-error performance.

I. Introduction

The Vertical Bell-Labs Layered Space-Time architecture (V-BLAST) [1], [2] is a communications method that exploits the multiple paths present in a rich-scattering channel to provide enormous capacity. M antennas are used to transmit and N to receive ($M \leq N$). The M antennas transmit at the same time and on the same frequency band, using the same signal constellation. The channel is represented by a matrix $\mathbf{H}^{N \times M}$, where each element h_{ij} is the transfer function from transmitter j to receiver i. The channel matrix is assumed to be perfectly estimated by the receiver, so that no distinction needs to be made between $\mathbf{H}$ and its estimate. The channel variation is assumed to be negligible during the transmission of LM symbols; afterwards, a new channel estimation is necessary.

V-BLAST doesn't use any kind of channel coding or time diversity; rather, it attempts to create M independent channels between transmitter and receiver. To accomplish this, V-BLAST needs the Moore-Penrose pseudo-inverse (MPPI) [3] of the channel matrix $\mathbf{H}$, and also of modified versions of $\mathbf{H}$ where some columns have been replaced by zeroes. This operation is very complex, and can prevent the use of V-BLAST at high bit rates.

In this letter results on the complexity of the pseudo-inverse operation, for various sizes of M and N, are presented. These results were obtained by computer simulation. Then, two modifications to the V-BLAST algorithm are proposed to reduce the number of operations required by the pseudo-inverse by a factor of approximately $1.5L$, without affecting the bit-error performance of the system. The cost of these modifications is a small increase in the memory requirements.

II. Quantifying V-BLAST Complexity

In order to evaluate how well suited V-BLAST is to operate in a high-speed, modern communications system, a first step is to count how many operations it requires per received information bit. This was done by computer simulation, with instruction counters in the appropriate places. The number of operations counted was then divided by the number of bits processed, which was between 160,000 and 240,000 in all cases.

Two types of operations were measured: those that involve an arithmetic or logic operation on a floating-point number, and those that involve a memory access (read or write). It was found that the MPPI accounts for more than 95% of the operations performed by V-BLAST, as shown in Table I for three antenna configurations.

From these numbers, several observations can be made:

- The complexity of V-BLAST needs to be reduced if it is to be used in a high-data-rate application. Even a configuration with a relatively small number of antennas requires thousands of operations per bit, which become tens of billions of operations per second in a $10Mb/s$ wireless network application, for example.
- Since the MPPI accounts for such a large percentage of the total complexity, the best approach for reducing it involves reducing the number of pseudo-inverses that need to be calculated.
- The number of memory accesses needed is also very large, suggesting that in a practical implementation of V-BLAST the memory architecture will be at least as important as the arithmetic unit architecture.

III. Reducing the number of MPPIs: Method I

V-BLAST requires the calculation of the MPPI of a matrix with zero or more columns replaced by zeroes. Let

$$B = pinv(A), \qquad i = 1, ..., N, j = 1, ..., M,$$

where **A** and **B** are two real matrices, with column k of **A** equal to zero; function $pinv(\cdot)$ denotes the pseudo-inverse operation. It can be observed that row k of **B** is zero also. Let $\mathbf{A}'$ equal to **A** except for column k, which is eliminated:

$$a'_{ij} = a_{ij}, \qquad j = 1, ..., k-1$$

$$a'_{ij} = a_{i,j+1}, \qquad j = k, ..., M-1$$

where a_{ij} denotes the element of **A** in row i and column j. Then, if $\mathbf{B}' = pinv(\mathbf{A}')$,

$$b'_{ji} = bji, \qquad j = 1, ..., k-1$$

$$b'_{ji} = bj+1, i, \qquad j = k, ..., M-1.$$

That is, the elements of $\mathbf{B}'$ outside of row k are the same as those of **B**. This suggests that the MPPI of **A** can be more efficiently calculated if it is performed on $\mathbf{A}'$ instead, and then a row of zeroes is inserted at row k in $\mathbf{B}'$ to find $\mathbf{B}$.

A can be thought of as the coefficient matrix of a system of linear equations, with the coefficient of variable a_k set to zero in all equations. It is clear that a_k doesn't contribute to the solution of the system and thus can be safely removed.

V-BLAST calculates the MPPI of the channel matrix and then replaces one of its columns by zeroes. This operation is performed a total of M times, zeroing a different column each time and keeping those already zeroed. Without optimization, V-BLAST will perform M MPPIs on matrices of dimension $N \times M$. If the above procedure is performed before each MPPI calculation, then V-BLAST will perform one $N \times M$ MPPI, one $N \times (M-1)$ MPPI, and so on until the last $N \times 1$ MPPI. The average number of columns operated on is $M(M+1)/2$ instead of M^2, which means the number of operations performed is reduced by a factor of $1/2 + 1/(2M)$, or approximately $1/2$.

IV. Reducing the number of MPPIs: Method II

V-BLAST is usually presented as an algorithm that operates on the vector of symbols received by the N antennas in the system. At first glance, this could seem to imply that M MPPIs will need to be calculated for each received vector. However, during each coherence interval, L received vectors are affected equally by the channel. This means that the processing order of the symbols will be the same for all vectors in the interval. This, in turn, means that the same M MPPIs are needed for processing each vector in the interval.

It is possible, then, to first calculate the M MPPIs that will be needed throughout the coherence interval, and then instruct V-BLAST to use the stored MPPIs instead of calculating each one anew. If the M MPPIs require P operations, then the application of V-BLAST to each vector will require PL operations during each coherence interval. If the M MPPIs are calculated first and kept in storage, then only P operations are needed for the whole interval. The number of operations required is reduced in this manner by a factor of L.

V. Memory requirements

Method I requires an additional MN complex words to store the channel matrix without the zeroed columns. Depending on the implementation of the MPPI, another MN words might be needed to store the result, because some algorithms destroy the matrix being operated on.

For method II, $M(MN)$ words are needed to store the M MPPIs. If this amount of memory is not available, a compromise can be reached by storing those MPPIs for which there is space, and calculating the rest once for each vector.

In the implementation used to generate the present results, the memory needed before optimization is $4N + 7M + 12MN$. The optimizations increased the memory usage by 35% in the 4×6 configuration, 52% in the 6×8 case and 67% in the 8×12 case.

VI. Simulation Results

Table II shows the number of operations, measured for the same antenna configurations as Table I, after concurrently using the two methods described above. The factor column represents the division of the number of operations required before the methods proposed were applied by the number of operations required afterwards. The coherence interval L selected was 10. It can be seen that the factor of reduction is close to $1.5L$.

As expected, the bit-error rate of V-BLAST was not affected by the modifications made to it.

VII. Conclusions

Two methods have been proposed to significantly reduce the complexity of V-BLAST, making it more attractive for high-speed applications, at the cost of some extra memory requirements. These methods do not reduce the bit-error rate performance of V-BLAST. The actual number of operations reported in this letter can vary among different implementations of V-BLAST; however, since the methods proposed reduce the number of pseudo-inverses that need to be calculated, and their size, the optimization obtained does not depend on implementation details.

REFERENCES

[1] G. D. Golden, C. J. Foschini, R. A. Valenzuela and P. W. Wolniansky, "Detection algorithm and initial laboratory results using v-blast space-time communication architecture," *Electronics Letters*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[2] P. W. Wolniansky, G. J. Foschini, G. D. Golden and R. A. Valenzuela, "V-blast: An architecture for realizing very high data rates over the rich-scattering wireless channel," presented at the Proc. Int. Symposium on Advanced Radio Technologies, Sept. 10 1998.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, 1996, pp. 257–258 and 448-456.

TABLE I

COMPARISON BETWEEN THE NUMBER OF OPERATIONS PER BIT REQUIRED BY THE MOORE-PENROSE PSEUDO-INVERSE OPERATION AND THE REST OF THE V-BLAST ALGORITHM

	memory		**arithmetic**	
$\boldsymbol{M \times N}$	**MPPI**	**rest of V-BLAST**	**MPPI**	**rest of V-BLAST**
4×6	2383.3	112.3	2285.2	46.5
6×8	7247.5	187.3	7100.2	74
8×12	18964.1	335.4	18315.9	129

.

TABLE II

NUMBER OF ARITHMETIC AND MEMORY OPERATIONS REQUIRED BY THE MOORE-PENROSE PSEUDO-INVERSE CALCULATIONS IN V-BLAST BEFORE AND AFTER THE OPTIMIZATION METHODS PROPOSED ARE APPLIED

	memory			**arithmetic**		
$\boldsymbol{M} \times \boldsymbol{N}$	**before**	**after**	**factor**	**before**	**after**	**factor**
4×6	2383.3	145	16.4	2285.2	151.3	15.1
6×8	7247.5	398.1	18.2	7100.2	429.3	16.5
8×12	18964.1	981.1	19.3	18315.9	1046.5	17.5

.

A SIMULATOR PLATFORM FOR MIMO SYSTEMS

Miguel Bazdresch, Georges Rodríguez-Guisantes

Ecole Nationale Supérieure des Télécommunications, 46, rue Barrault, 75634 Paris Cedex 13, France
miguelb@ieee.org

Abstract - A computer simulator package for MIMO systems is presented. This simulator is highly configurable, easily extensible, and can generate exhaustive performance and complexity measurements. Examples of use and capabilities are shown. The simulator's source code has been released to the research community under the MIT open-source license.

Keywords - V-BLAST, MIMO, algorithmic complexity, performance measurements, simulation.

I. Introduction

Computer simulation is an essential tool when studying the performance and complexity of the different receivers that have been proposed for use in multiple-input, multiple-output (MIMO) telecommunication systems. Simulation also has an important role to play in architecture exploration, indicating which portions of the algorithms need to be optimized, and where the bottlenecks are found.

In this paper, we present MSIM, a powerful simulator platform that has been successfully used to estimate the performance and complexity of several MIMO algorithms, and which is designed to be extensible, making it very easy to add new algorithms or new functionalities.

This simulator is presented to the community under MIT's open source license (which can be found at http://opensource.org/licenses/mit-license.php) with the aim of attracting improvements and contributions from other researchers.

We present our reasons for writing a MIMO simulator from scratch, instead of using existing tools, in Section II. In Section III we make a brief recount of the simulator's capabilities. Some examples of what can be done with MSIM are presented in Section IV. Finally, we state our conclusions in Section V.

II. Justification

There are many proven computational tools, commercial and open-source, that can be used to simulate a MIMO communications system. We have found, however, that it can be very difficult, or even impossible, to use these tools for detailed architecture exploration and complexity estimations, because they are intended to solve more general problems.

For example, one might want to use the LAPACK package of mathematical functions to build a simulator. One must then consider how LAPACK performs the calculations needed by the simulator. For instance, there is the problem of finding the Givens rotation of a matrix, needed for calculating the QR decomposition.

A Givens rotation is defined by a pair of functions, $c(f,g) = f/\sqrt{f^2+g^2}$ and $s(f,g) = g/\sqrt{f^2+g^2}$, where f and g are complex numbers [2]. This apparently straightforward calculation needs to be done very carefully in some limit cases, such as when f or g are very large, or when one of them is zero [4]. The LAPACK code for the Givens rotation takes all these limit cases into account, providing extremely efficient and reliable code as a result.

In MIMO communication systems, however, it is known that these limit cases will almost never appear. MSIM needs to be able to find the QR decomposition of a random matrix whose (complex) elements have a Gaussian distribution with zero mean and variance 0.5 per dimension [1]. Since, in this case, f and g will almost invariably be small and different from zero, the general solution implemented in LAPACK results in considerable inefficiencies.

Furthermore, modifying LAPACK to eliminate the unwanted functionality can prove to be harder than writing the code from scratch. Writing all the mathematical functions we need from scratch allows us to tailor the code precisely to our needs, and facilitates the task of estimating its complexity.

For this reason, we decided to create MSIM from scratch; we wish to contribute this simulator to the research community in order to help others facing the same questions that we did.

III. Simulator Capabilities

A. Organization

MSIM is divided into modules, which are called, in order, from a master controller module. First, configuration is carried out from a text file that contains the commands to be executed; then, the remaining modules are executed in a loop until a certain condition (typically, a number of bit errors) is met.

The modules present at this time are: source, coder, modulator, channel, channel estimator, receiver, demodulator, error counter, and a module that reports all the results. Each module's capabilities can be easily expanded.

The receiver module presently includes a V-BLAST receiver with four different methods of calculating the pseu-

doinverse of the channel matrix, and a receiver based on maximum-likelihood decoding of a lattice.

At present, the modules are functional enough to create most typical simulation scenarios. Some of the main parameters that can be configured are:

- Selection of range of block sizes
- Selection of range of average SNRs
- Type of constellation for each Tx antenna
- Power of the constellation
- Number of errors to simulate

B. Reporting

When it is finished, the simulator produces extensive reports of all performance and complexity measurements. These reports are arranged in such a way that producing graphics from them, using tools such as gnuplot or a spreadsheet, is very easy.

Among the results provided are: BER, $BLER$, true average SNR, and detailed complexity measurements, which include counts of all arithmetic and memory operations performed. Extensive debug information can be produced on demand, including intermediate values of calculations.

Most of the debugging and complexity functions can be disabled, if simulation speed is a priority.

C. Performance

On an Intel Pentium 3 at 800MHz computer MSIM can simulate a V-BLAST receiver at around 750,000 bits per second for a system with 4 transmit and 4 receive antennas, and 10 vectors per block.

IV. Experimental results

A. Performance estimation

As an example of performance simulation with MSIM, consider a MIMO system with 4 transmit and 6 receive antennas, where each transmitter uses a QAM-16 signal constellation of average energy equal to 1, and with a block size equal to 10 vectors (so that a block comprises 160 bits). Fig. 1 shows a comparison of two receivers, V-BLAST [1] and a receiver based on lattice decoding [3]. The performance measure being used is the block error rate ($BLER$), that is, the probability of at least one bit per block being in error.

The V-BLAST estimates for $BLER$ produced by MSIM coincide with other results reported elsewhere, for example in [7]

B. Complexity measurements

Figures 2 and 3 show how V-BLAST compares with the lattice-decoder receiver, both in total number of arithmetic operations (sums, multiplications, and square roots) and memory accesses. MSIM can also produce detailed statistics for each type of operation. The numbers shown are averages per bit of information transmitted.

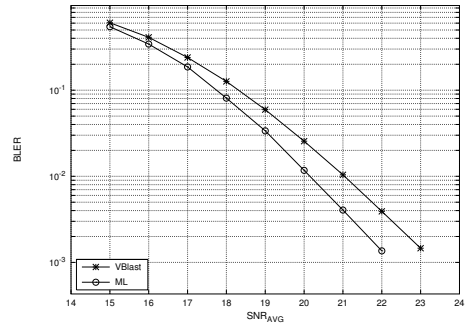

Fig. 1
Comparison between V-BLAST and a lattice-decoder algorithm.

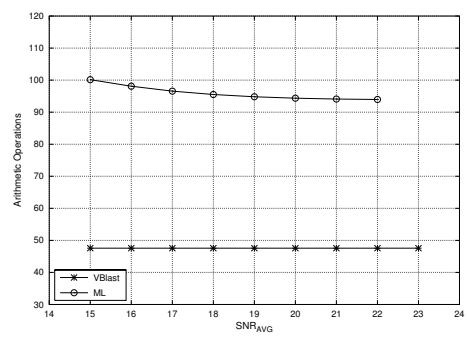

Fig. 2
Comparison of number of arithmetic operations required for V-BLAST and a lattice-decoder algorithm.

C. Architecture exploration

To illustrate how architecture exploration can be done with MSIM, consider the following problem. The lattice-decoder receiver can potentially benefit if the channel matrix is reduced according to the LLL criteria [5], [6]. The LLL reduction is very expensive, but it only needs to be done once per block, and the decoding of each block is faster when the channel matrix is reduced. The architectural question arises: in which cases is it worth the investment to carry out the LLL reduction?

Fig. 4 can help answer this question. As can be seen, for block sizes of around 10, there is a substantial increase in complexity when the LLL reduction is carried out. For larger block sizes, there is almost no advantage. The conclusion is that for this particular MIMO system, the LLL reduction has no tangible benefits.

As another example of the kind of architectural exploration possible with MSIM, it was found that, for the V-

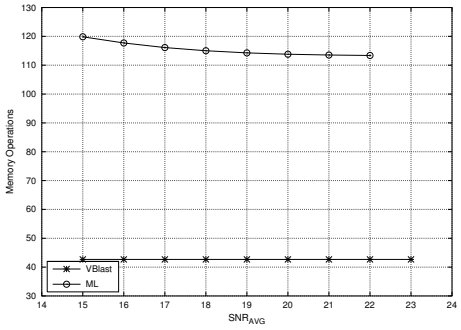

Fig. 3
Comparison of number of memory accesses required for V-BLAST and a lattice-decoder algorithm.

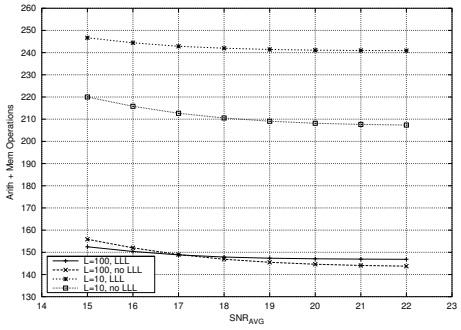

Fig. 4
Effect of block size L on the complexity of the lattice-decoder algorithm

BLAST receiver, 92 multiplications are needed per received bit. However, 24% of these multiplications are done during matrix multiplications. This allows an estimate of the maximum gain in performance that can be obtained by investing architectural resources in a fast matrix multiplier.

V. Conclusions

In this paper, we have presented a simulator for MIMO communications systems with the capacity to estimate algorithmic complexity and error performance, and to carry out architecture exploration. The source code of this simulator has been released, with the intention of attracting improvements from the research community. Some examples of what can be done with current capabilities have been shown.

The MSIM simulator can be found at: http://www.comelec.enst.fr/ rodrigez/msim.html.

References

[1] G. D. Golden, C. J. Foschini, R. A. Valenzuela and P. W. Wolniansky, "Detection algorithm and initial laboratory results using v-blast space-time communication architecture," *Electronics Letters*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[2] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, 1996.

[3] E. Agrell, T. Eriksson, A. Vardy and K. Zeger, "Closest Point Search in Lattices", *IEEE Trans. Inf. Theory*, vol. 48, pp. 2201-2214, August 2002.

[4] D. Bindel, J. Demmel, W. Kahan and O. Marques, "On computing Givens rotations reliably and efficiently", Technical Report CS-00-448, University of Tennessee, October 2000. [Online] Available as LAPACK Working Note 148, http:// www.netlib.org/lapack/lawns/lawn148.ps

[5] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, 1995.

[6] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovsz, "Factoring polynomials with rational coefficients", Math. Annalen, vol. 261, pp. 515534, 1982.

[7] N. Boubaker, K.B. Letaief, and R.D. Murch, "Performance of BLAST over frequency-selective wireless communication channels", *IEEE Tran. Comm.*, vol. 50, pp. 196-199, February 2002.

Least-Squares Techniques Applied to V-BLAST Receivers

M. Bazdresch, *Student Member, IEEE*, J. Rodríguez-Guisantes

***Abstract*— In this paper, we propose and compare three algorithms based on V-BLAST and examine their block error rate and their algorithmic complexity for different combinations of block length and number of antennas. Of special interest are two algorithms that use techniques from least-squares, specifically the QR decomposition, to substantially lower the complexity of the original V-BLAST. We present the algorithms in detail as well as some simulation results. We show that the algorithm of choice depends on the number of antennas in the system, the block length L, and on whether the application at hand requires low-error rates or low-complexity.**

I. Introduction

V-BLAST is a receiver for multiple-input, multiple-output communications systems [1]. Although suboptimal, it is attractive because it has substantially lower complexity than optimal receivers. In addition, it can effectively exploit space diversity, nearing optimal performance when the number of receiver antennas in the system increases in relation to the number of transmitters.

Several implementations of V-BLAST have been presented in the literature since the original idea was published; some examples are those implementations given in [2]–[4]. These usually trade bit-error rate (BER) for complexity, or vice-versa. Since V-BLAST's BER is suboptimal, and it is too complex to allow a feasible hardware implementation for a relatively small number of antennas, it seems useful to find algorithms that have nearly the same or better BER than V-BLAST, while decreasing its complexity.

In this paper, we focus on three different implementations of V-BLAST and compare them in terms of their BER and their algorithmic complexity. We measured their complexity empirically, by running each one in a computer and counting each operation performed (both arithmetic operations and memory accesses). This measurement does not take into account potential gains to be obtained by techniques such as parallelism; however, it does offer a common ground for comparison and eases the identification of each algorithm's bottlenecks.

The first implementation we consider, which we simply denote V-BLAST, is taken directly from [1]. As explained in Section III, we rearrange it to diminish the required number of matrix pseudo-inverses. This is our "base-line" implementation. The second version of the algorithm is based on an idea first proposed in [5]: to consider V-BLAST as if it were a least-squares algorithm to solve linear systems of equations. The QR decomposition is used to reduce the size of the problem, significantly lowering the complexity in some cases. We call our version of this algorithm LS-BLAST. Finally, we compare these algorithms to the one presented in [6], which we denote V-SQR, that trades some BER performance for a large reduction in complexity.

We compare these algorithms with different combinations of block sizes and number of transmit and receive antennas to find which one is better suited for each particular case.

The simulation results we present were corroborated by running these algorithms in a digital signal processor.

II. Models and Definitions

Consider a communications system where a data stream is demultiplexed into M streams, and each of these sub-streams is fed to a transmit antenna. All M antennas are symbol-synchronized, use the same frequency band, and use the same signal constellation S. The receiver consists of N antennas ($N \geq M$). The channel is modelled as an $N \times M$ complex matrix $\mathbf{H}$; each component h_{ij} of H is taken from a Gaussian distibution with zero mean and variance 0.5. Component h_{ij} of $\mathbf{H}$ is the channel model between transmitter j and receiver i. $\mathbf{H}$ is assumed to be full rank.

In this model, the received signal is given by

$$\mathbf{r} = \mathbf{H}\mathbf{a} + \mathbf{n} \tag{1}$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_M)$ denotes the vector of transmitted symbols, and $\mathbf{n} = (n_1, n_2, \ldots, n_N)$ is a noise vector whose elements are complex Gaussian independent random variables of mean zero and variance n_0.

The channel is assumed to remain constant during the transmission of L symbol vectors. Such a channel

is frequently referred to as a *block-fading* channel [7]. Each set of L symbol vectors is known as a *block*.

Finally, the receiver is assumed to know H perfectly.

III. The V-BLAST algorithm

We re-write V-BLAST as follows:

Algorithm V-BLAST

Input: an $N \times M$ matrix $\mathbf{H}$, a set of L $N \times 1$ vectors $\mathbf{r}_i$, $i = 1, \ldots, L$, and a signal constellation S.

Output: a set of L $M \times 1$ vectors $\hat{\mathbf{a}}_i$. Each vector $\hat{\mathbf{a}}_i$ is such that its elements are in S and $\mathbf{H}\hat{\mathbf{a}}_i = \mathbf{r}_i + \mathbf{v}_i$, where $\mathbf{v}_i$ is an error vector.

Step 1: Let $i = 1$, $\mathbf{A} = \mathbf{H}$

Step 2: Repeat M times:

- **2.i** Find $\mathbf{G}_i = \mathbf{A}^+$
- **2.ii** Let $k_i = \operatorname{argmin}_j \|(\mathbf{G}_i)_j\|$
- **2.iii** Let o_i equal to corresponding column in $\mathbf{H}$
- **2.iv** Remove column k_i of $\mathbf{A}$
- **2.v** Let $i = i + 1$

Step 3: Repeat L times:

- **3.i** Let $i = 1$
- **3.ii** Repeat M times:
  - **3.a** Let $\mathbf{w}$ equal to row k_i of $\mathbf{G}_i$
  - **3.b** Let $y = \mathbf{w}^H \mathbf{r}_i$
  - **3.c** Let $\hat{a}_{i,o_i} = f_q(y)$
  - **3.d** Let $\mathbf{z}$ equal to column o_i of $\mathbf{H}$
  - **3.e** Let $\mathbf{r}_i = \mathbf{r}_i - \hat{a}_{i,o_i}\mathbf{z}$
  - **3.f** Let $i = i + 1$

where $(\mathbf{G})_j$ is row j of $\mathbf{G}$, $\mathbf{H}^+$ is the Moore-Penrose pseudo-inverse of $\mathbf{H}$ [8], $\mathbf{w}^H$ is the conjugate transpose of vector $\mathbf{w}$, and $f_q(\cdot)$ is the appropriate quantizing operation for the constellation in use.

V-BLAST iterates over each symbol vector, estimating each symbol in turn. It starts by choosing the symbol with best SNR, and then subtracts the estimated symbol from the remaining ones. The rewritten algorithm makes evident that only M pseudo-inverses are needed per block.

Furthermore, at each step one whole column of matrix $\mathbf{A}$ is removed; thus, the pseudo-inverses are calculated on matrices of diminishing size. Step **2.iii** is needed to preserve the symbol ordering relative to the original channel matrix $\mathbf{H}$ and not to matrix $\mathbf{A}$.

When V-BLAST is presented as above, it becomes apparent that its complexity can be divided in two parts. One part, which we call the *block setup phase*, is computationally expensive, but is performed only once per block. Its complexity depends on M and N. The other part, called the *symbol estimation phase*, is computationally simpler but is performed many times. Its complexity depends not only on M and N but also on L.

IV. QR Decomposition in V-BLAST

The estimation problem can be seen as one of solving a system of linear equations perturbed by noise, with the added constraint that the solution must be an element of a constellation S. The QR decomposition can be used to solve this problem; the basic idea is to convert Eq. 1 to:

$$\begin{aligned} \mathbf{r} &= \mathbf{QRa} + \mathbf{n} \\ \mathbf{Q}^H\mathbf{r} &= \mathbf{Ra} + \mathbf{Q}^H\mathbf{n} \\ \mathbf{x} &= \mathbf{Ra} + \mathbf{v} \end{aligned} \tag{2}$$

where $\mathbf{Q}$ is an $N \times M$ matrix with orthonormal columns and $\mathbf{R}$ is an $M \times M$ upper-triangular matrix.

The complexity of V-BLAST depends on the size of $\mathbf{H}$; at the cost of a matrix decomposition and a matrix product, the complexity now depends on the size of $\mathbf{R}$.

Exploiting this idea, V-BLAST may be written as follows. It has the same inputs and outputs as V-BLAST.

Algorithm LS-BLAST

Step 1: Compute an $N \times M$ orthonormal matrix $\mathbf{Q}$ and an $M \times M$ upper-triangular matrix $\mathbf{R}$, such that $\mathbf{H} = \mathbf{QR}$

Step 2: let $i = 1$, $\mathbf{A} = \mathbf{R}$

Step 3: Repeat M times:

- **3.i** Find $\mathbf{G}_i = \mathbf{A}^+$
- **3.ii** Let $k_i = \operatorname{argmin}_j \|(\mathbf{G_i})_j\|$
- **3.iii** Let o_i equal to corresponding column in $\mathbf{H}$
- **3.iv** Remove column k_i of $\mathbf{A}$
- **3.v** Let $i = i + 1$

Step 4: Let $i = 1$

Step 5: Repeat L times:

- **5.i** Let $\mathbf{x} = \mathbf{Q}^H \cdot \mathbf{r}_i$
- **5.ii** Repeat M times:
  - **5.a** Let $\mathbf{w}$ equal to row k_i of $\mathbf{G}_i$
  - **5.b** Let $y = \mathbf{w}^H\mathbf{x}$
  - **5.c** Let $\hat{a}_{i,o_i} = f_q(y)$
  - **5.d** Let $\mathbf{z}$ equal to column o_i of $\mathbf{H}$
  - **5.e** Let $\mathbf{x} = \mathbf{x} - \hat{a}_{i,o_i}\mathbf{z}$
  - **5.f** Let $i = i + 1$

In comparison with V-BLAST, the block setup phase needs an extra QR decomposition; in exchange, the pseudo-inverses are performed on an $M \times M$ matrix. Likewise, the symbol estimation phase has an extra vector-matrix multiplication, but the size of all other operations depends exclusively on M instead of both M and N.

We now prove that the algorithm LS-VBLAST and V-BLAST as originally proposed are equivalent; that is, they produce exactly the same output.

The key point is the calculation of y (in step $3.b$ of V-BLAST and step $5.b$ of LS-BLAST). If both algorithms compute the same y then they will produce the same output.

Proof: Let y_V be defined as in step $3.b$ of V-BLAST.

$$\begin{aligned} \mathbf{w} &= (\mathbf{G})_{k_i} \\ y_V &= \mathbf{w}^H \mathbf{r}_i \\ &= \mathbf{w}^H \cdot (\mathbf{H}\mathbf{a} + \mathbf{n}) \\ &= a_{k_i} + \mathbf{w}^H \mathbf{n} \end{aligned} \tag{3}$$

Let y_L be defined as in step $5.b$ of the LS-BLAST algorithm. To avoid confusion we denote the pseudo-inverse of $\mathbf{A}$ by $\mathbf{T}$.

$$\begin{aligned} \mathbf{z} &= (\mathbf{T})_{k_i} \\ y_L &= \mathbf{z}^H \mathbf{x} \\ &= \mathbf{z}^H \mathbf{Q}^H \mathbf{r}_i \\ &= \mathbf{z}^H (\mathbf{R}\mathbf{a} + \mathbf{Q}^H \mathbf{n}) \\ &= a_{k_i} + \mathbf{z}^H \mathbf{Q}^H \mathbf{n} \\ &= a_{k_i} + \mathbf{z}^H \mathbf{R} \mathbf{H}^+ \mathbf{n} \\ &= a_{k_i} + \mathbf{e} \mathbf{H}^+ \mathbf{n} \\ &= a_{k_i} + \mathbf{w}^H \mathbf{n} \end{aligned} \tag{4}$$

where $\mathbf{e}$ is a vector whose k_i-th element is equal to 1, and all others are equal to 0.

Since y_V and y_L are equal, then the estimates calculated by V-BLAST and LS-BLAST are equal too.

■

We verified in simulation that these algorithms produce identical results in terms of bit-error rate.

V. The Sorted QR algorithm

This variant of V-BLAST was first proposed in [6]. Instead of calculating the channel matrix pseudo-inverses, it relies on a heuristic that estimates the length of the rows of $\mathbf{H}^+$ using the length of the columns of $\mathbf{H}$ (orthogonal to the vector space of the columns corresponding to as yet unestimated symbols).

This algorithm may be written as follows. Its inputs and outputs are identical to those of V-BLAST.

Algorithm V-SQR

Step 1: Find the sorted-QR decomposition of $\mathbf{H}$, store it in $\mathbf{Q}$ and $\mathbf{R}$.

Step 2: Let $i = 1$

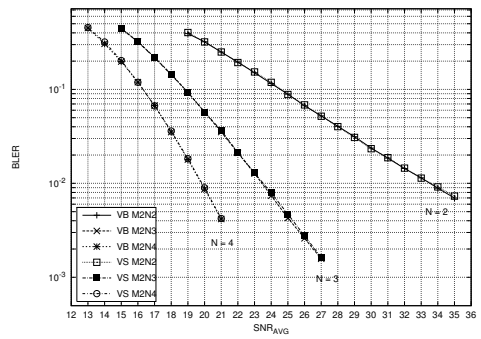

Fig. 1. $BLER$ as a function of SNR_{avg} for M = 2; N = 2, 3, 4; L = 10. VB denotes V-BLAST; VS denotes V-SQR.

Step 3: Repeat L times:

3.i Let $\mathbf{x} = \mathbf{Q}^H \cdot \mathbf{r}_i$

3.ii Find $\hat{\mathbf{a}}_i$ that solves $\mathbf{R}\hat{\mathbf{a}} = \mathbf{x}$

3.iii Re-arrange $\hat{\mathbf{a}}_i$ to correspond to original ordering of columns of $\mathbf{H}$

3.iv Let $i = i + 1$

The sorted-QR decomposition algorithm is given in the original paper. Step $3.iii$ is necessary since the ordering of the rows of matrix $\mathbf{R}$ does not correspond to the original ordering of the symbols in each vector.

VI. Simulation Results

We define the block-error rate ($BLER$) as the proportion of blocks that have at least one error. The number of operations required by each algorithm for estimating one bit of information is called O_b; this number includes all arithmetic as well as memory operations. For simplification, in this paper all operations are given equal weight; more detailed results are available upon request, or can be reproduced with our simulator, which is publicly available [9].

All simulations were run until 2000 block errors occurred. A $16 - QAM$ constellation was used in all cases. The noise measure used was SNR_{avg}, as defined in [1].

A. $BLER$ performance

Figs. 1 to 3 present the $BLER$ performance under several combinations of M and N. Results for LS-BLAST are not shown, since they are identical to those of V-BLAST.

We conclude that for values of N larger than M, the $BLER$ performance of V-SQR is similar to that of V-BLAST. As M and N get closer, however, the difference begins to be significant; for $M = N = 8$, it is more than $2dB$.

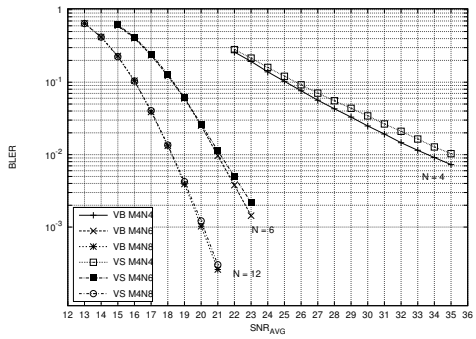

Fig. 2. $BLER$ as a function of SNR_{avg} for M = 4; N = 4, 6, 8; L = 10. VB denotes V-BLAST; VS denotes V-SQR.

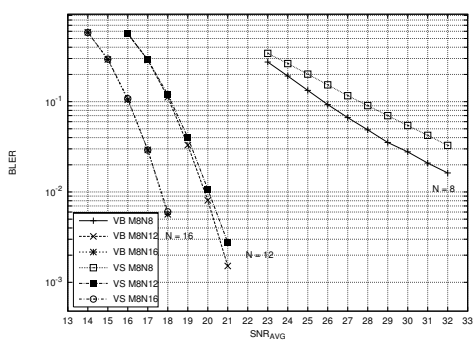

Fig. 3. $BLER$ as a function of SNR_{avg} for M = 8; N = 8, 12, 16; L = 10. VB denotes V-BLAST; VS denotes V-SQR.

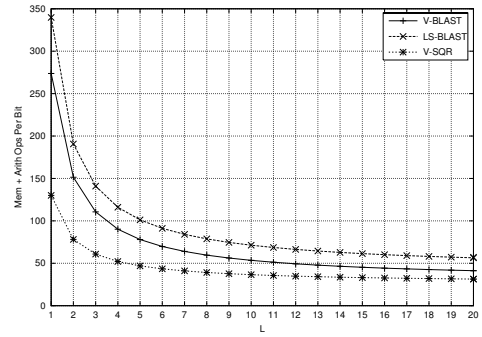

Fig. 4. O_b as a function of L for M = 4; N = 4;

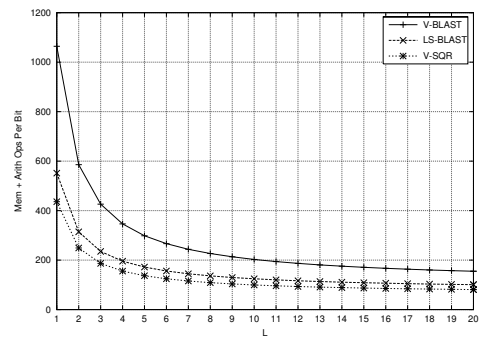

Fig. 5. O_b as a function of L for M = 4; N = 16;

B. Complexity as a function of L

We set $M = 4$ and determine how O_b changes as a function of L. The contribution to the block setup phase complexity diminishes as L grows, while that of the symbol estimation phase grows. Our results are presented in Fig. 4 through 6.

The conclusion is that the complexity advantage of V-SQR decreases as L increases, with a more pronounced effect for large N. The reason for this is the slight overhead in the symbol decoding phase of V-SQR as compared to LS-BLAST.

C. Complexity as a function of N

Again we set $M = 4$ and determine how O_b changes with increases in N, for $L = 15$. Our results are presented in Fig. 7.

The benefits of both LS-BLAST and V-SQR are clear; furthermore, for sufficiently large N, LS-BLAST has lower complexity.

The difference in slope between the traditional V-BLAST and the other two algorithms is worth noting; it is caused by the lessened influence of N upon the symbol estimation phase of the algorithm.

VII. Experimental Results

We ran these algorithms on a Texas Instruments 6711 digital signal processor (DSP) for different sizes of L, with $M = 2$ and N equal to 3, 6 and 23. The results confirm our predictions that the complexity per bit diminishes with L and that LS-BLAST is substantially faster than V-BLAST as N increases. For $N = 3$ there is little advantage in using LS-BLAST; for $N = 6$, however, the complexity is already reduced around 33%, and for $N = 23$ the reduction is around 45%.

VIII. Conclusions

Three MIMO receiver algorithms based on V-BLAST have been presented. Their performance in terms of block-error rate and complexity has been analysed. Results on the behavior of each algorithm have been set forth, showing that the choice of one algorithm over another is not straightforward; it will depend on the number of antennas in the system, the block length L,

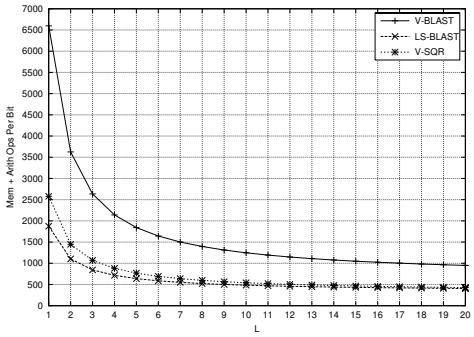

Fig. 6. O_b as a function of L for M = 4; N = 100;

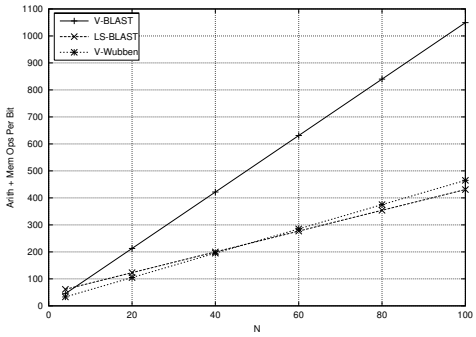

Fig. 7. O_b as a function of N; M = 4; L = 15

and on whether the application at hand requires low-error rates or low-complexity. Experiments on a digital signal processor validate these results.

References

[1] G. Golden, C. Foschini, R. Valenzuela, and P. Wolniansky, "Detection algorithm and initial laboratory results using v-blast space-time communication architecture," *Electronic Letters*, vol. 35, no. 1, 7th January 1999.

[2] S. Bäro, G. Bauch, A. Pavlic, and A. Semmler, "Improving blast performance using space-time block codes and turbo decoding," in *Proc. IEEE Globecom*, 2000, pp. 1067–1071.

[3] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a blast system," *IEEE Trans. Signal Processing*, vol. 51, no. 7, pp. 1722–1730, 2003.

[4] A. Bhargave, R. J. de Figueiredo, and T. Eltoft, "A detection algorithm for the v-blast system," in *Proc. GLOBECOM*, 2001, pp. 25–29.

[5] D. Shiu and J. M. Kahn, "Layered space-time codes for wireless communications using multiple transmit antennas," pp. 436–440, 1999.

[6] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEE Electronic Letters*, vol. 37, no. 22, pp. 1348–1350, 2001.

[7] E. Biglieri, J. Proakis, and S. Shamai, "Fading channels: Information-theoretic and communications aspects," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2619–2691, Oct. 1998.

[8] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996, pp. 257–258.

[9] M. Bazdresch and J. Rodríguez-Guisantes, "The msim simulator." [Online]. Available: http://www.comelec.enst.fr/˜rodrigez/msim/index.html

Performance Study of Space-Time Communications Systems Based on the VBLAST Algorithm

Karina Sosa*, Luis Fernando González Pérez, Assistant Professor*,
Miguel Bazdresch, Student Member IEEE**,
Jorgue Rodríguez-Guisantes, Associate Professor**,
*Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México
**Ecole Nationale Supérieure des Télécommunications, Paris, FRANCE
ksosa@itesm.mx, gonzalez.luis@itesm.mx,miguelb@ieee.org, rodrigez@enst.fr

Abstract

Capacity in wireless channels has become the prime aspect in mobile communications. Significant improvements are possible with MIMO channels -the use of multiple antennas at both the transmitter and receiver-. Reception in this type of channels is achieved with space-time coding by jointly designing channel coding, modulation and equalization, allowing an important increase in the throughput of band-limited wireless channels. One of the most important space-time coding systems is the Bell-Labs Layered Space-Time (BLAST) coding technique. Different variants of this algorithm exist focusing on computational complexity reductions. One of these variants is the Vertical BLAST algorithm which has different realizations depending on the way the signal processing at the receiver is performed. In this paper, an analysis of these realizations is presented from a performance-complexity standpoint. These variants are the Singular-Value Decomposition, the Sorted-QR Decomposition and the Least-Square algorithm. It is shown that the latter presents the best performance-complexity trade-off.

1. Introduction

Demand for capacity in wireless communications has been rapidly increasing worldwide. Nevertheless, the available spectrum is limited and capacity needs cannot be met without a significant increase in spectral efficiency. Advances in channel coding make it feasible to approach the Shannon capacity limit in systems with single antenna links (SISO - Single Input Single Output systems) [1]. However, even more significant improvements can be achieved with MIMO (Multiple Input - Multiple Output) systems by increasing the number of antennas at both the transmitter and the receiver [2]. It has been demonstrated that SISO systems can achieve spectral efficiencies between 1-2 bits/sec/Hz in cellular systems whereas MIMO systems with 8 antennas at each side can obtain 42 bits/sec/Hz, at an SNR of 20 dB [3]. Such spectral efficiencies highlight the potential advantages of MIMO systems. In general, capacity grows linearly with the number of transmit antennas, M, as long as the number of receive antennas, N, satisfies the condition $N \geq M$. As a result, MIMO systems are excellent candidates for high-data-rate future mobile systems, such as 3G and beyond [2].

MIMO systems were possible when space-time coding algorithms appeared. The principal idea of this approach is to jointly design modulation, coding and equalization. Foschini proposed a layered space-time architecture that approaches Shannon's theoretical limits. This algorithm is called DBLAST (Diagonal Bell Labs Layered Space-Time) [3].

DBLAST architectures use multiantenna arrays in both transmitter and receiver, and a diagonal layered coding structure in which blocks of information are dispersed across diagonals in space–time. Nevertheless, this algorithm suffers from computational complexity, making it inappropriate for hardware implementations. Therefore, simplified versions of this algorithm were soon proposed presenting good performance-complexity trade-offs. An excellent simplified version of the DBLAST algorithm is the Vertical-BLAST (V-BLAST) technique [11]. This approach can reach tens of bits/s/Hz with multiple antennas and reduced complexity.

In a V-BLAST system, the uncoded data stream is demultiplexed into M substreams, each being

transmitted simultaneously by one transmit antenna. At the receiver, received signals from N received antennas are detected by a decision feedback algorithm. At present time, multiple realizations of V-BLAST exist which aim to reduce its computational complexity. The aim of this paper is to analyze these realizations from a performance-complexity standpoint. The variants considered are the Singular Value Decomposition, Sorted QR Decomposition and the Least-Square algorithm [7]. In addition, these variants are compared to the near-optimum Maximum Likelihood (ML) algorithm proposed in [5]. The remainder of the paper is organized as follows. Section 2 gives a brief explanation of the overall system. Section 3, 4, 5 and 6 describe the near-optimal ML detection algorithm and the three V-BLAST variants, respectively. Performance results and a comparison in terms of computational complexity and performance between these four algorithms are given in section 7. Finally, conclusions and future work are discussed in section 8.

2. System Description

The V-BLAST system is shown in Figure 1. It consists of M 16-QAM transmitters operating at the same frequency band at a symbol rate of 1/T symbols/s with synchronized symbol timing. For simplicity, we assume that the transmission is organized in bursts of L symbols. Power in each transmitter is proportional to 1/M so that the total power is constant and independent of M.

The receiver consists of N conventional QAM receivers. They receive the signals from all M transmit antennas. Flat fading is assumed, and the matrix channel transfer function is H_{MxN}, where $h_{i,j}$ is the complex channel transfer function from transmit antenna i to receive antenna j (N≥M). The channel is Gaussian-distributed with zero mean and variance 0.5.

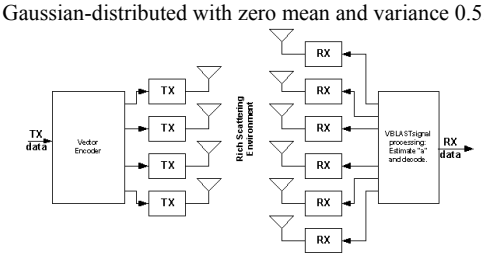

Figure 1: VBLAST architecture.

In the model, it is assumed that the detection process is symbol-synchronous. Letting $a = (a_1, a_2, ..., a_M)^T$ denote the vector of transmitted symbols, the corresponding received N-vector, $\boldsymbol{r}$ is

$$r = Ha + v \qquad (1)$$

where v is a white-noise Gaussian vector. In addition, the channel is quasi-stationary implying negligible variations over the L symbol periods, and it is estimated accurately by means of a training sequence embedded in each burst. Hence, we will not make distinction between **H** and its estimate. This assumption is referred to as block-fading channel [4]. Each set of L symbol vectors is known as a block. Next, the analysis of the different decoding approaches is discussed. A near-optimal ML detection algorithm is described, followed by the V-BLAST variants.

3. Agrell Algorithm

This algorithm is an ML decoder approximation that computes the closest lattice point to the received vector, achieving excellent performance. However, the speed with which it decodes the transmitted blocks varies considerably depending on the channel matrix size. The algorithm is described in figure (2) and explained next [5].

Algorithm ClosestPoint(G,x)

Input: An NxM generator matrix G, and an M-element vector $x \in \Re^M$ to decode in A(G).

Output: a lattice point $\hat{x} \in A(G)$ that is closest to x.

a. Let G_2:=WG, where W is an NxN matrix with integer entries and determinant ±1.
b. Compute an NxM orthonormal matrix Q such that G_2=G_3Q, where G_3 is an NxN lower-triangular matrix with positive diagonal elements.
c. Let H_3:=G_3^{-1}
d. Let x_3 = xQ^T
e. Let $\hat{u}_3$ = DECODE(H_3,x_3)
f. Return $\hat{x} = \hat{u}_3 G_2$ and convert to ML point.

Figure 2: Agrell Algorithm.

Assume that a generator matrix **G** and an input vector ***x*** are given. By linear integer row operations, matrix **G** is transformed into another matrix, say $\mathbf{G_2}$, which generates an identical lattice [12]. The purpose of this transformation is to speed up the DECODE algorithm which finds the closest lattice point to the received vector ***x***. $\mathbf{G_2}$ is rotated and reflected into a lower triangular form $\mathbf{G_3}$ so that all transformations give the same result.

It is essential to rotate and reflect the input vector $\boldsymbol{x}$ in the same way, so that the transformed input vector, $\boldsymbol{x_3}$, is in the same relation to the rotation of $\mathbf{G_3}$ as $\boldsymbol{x}$ is to the rotation of **G**. All this can be regarded as a change of the coordinate system. Now, the search problem has a form that is suitable for the DECODE procedure which finds the closest lattice point $\mathbf{x'_3}$ in this coordinate system. Reversing the operations of rotation and reflection produces $\boldsymbol{x'}$, the lattice point to $\boldsymbol{x}$ in rotation of **G**. Following these steps, the closest lattice point is found.

The DECODE procedure is the heart of the lattice search algorithm. It divides matrix $\mathbf{H_3}$ (which is the inverse of $\mathbf{G_3}$) (see Fig. 2) into N sublayers of dimension N and searches in each sublayer for the closest lattice point to vector $\mathbf{x'_3}$. At each searching step, the dimension of the sublayer can be reduced, reducing in turn the complexity of the searching procedure. There can be other cases where dimensionality must be increased if no smaller distance is encountered. The procedure continues until the algorithm has successfully moved down to the zero-dimensional layer, i.e. a lattice point. This point is stored as the potential output point, the lowest distance is updated and the algorithm moves up to continue searching for other potential lattice points without restarting the entire algorithm.

The computational complexity of this algorithm can be estimated as

$$Mem = 7 * S * M^2 * N * L$$
$$Add = 2.5 * S * M^2 * N * L^2$$
$$Mult = 0.2 * S * M^2 * N * L^2$$
$$Div = 0.02 * S * M^2 * N * L^2$$
$$Sqrt = 1.25x10^{-4} * S * M^2 * N * L^2$$

where M and N are the number of transmit and receive antennas, respectively, L is the number of symbols per block and S is the number of bits per symbol (in our case each symbols carries four information bits, 16-QAM). *Mem* refers to the storage requirements needed and *Add, Mult, Div* and *Sqrt* refers to computational complexity in terms of the arithmetic operations performed for each decoded bit.

4. SVD Based V-BLAST Algorithm

The Singular Value Decomposition approach constructs an upper bidiagonal matrix **A** where the V-BLAST algorithm works on. The V-BLAST algorithm iterates over each symbol vector, estimating each symbol in turn. It starts by choosing the symbol with best SNR, and then subtracts the estimated symbol from the remaining ones. Furthermore, at each step a whole column of matrix **A** is removed; thus, the pseudo-inverses are calculated on matrices of reduced size. These pseudo-inverses are calculated only once requiring thus a much lower computational complexity than that of the ML algorithm. This algorithm is described in figure 3 [6].

The storage requirements and computational complexity of this algorithm are

$$Mem = 11 * S * M^2 * N * L \qquad (2)$$
$$Add = 0.65 * S * M^2 * N * L^2$$
$$Mult = 0.68 * S * M^2 * N * L^2$$
$$Div = 0.023 * S * M^2 * N * L^2$$
$$Srt = 6.1x10^{-3} * S * M^2 * N * L^2$$

Algorithm V-SVD

a. Let i = 1
b. Repeat M times.
   - Identify w[j] that are too small, and invert others. $H_3 = G_3^{-1}$.
   - Let $V = W^{-1}$.
   - Let T = V*U'
   - Apply VBLAST algorithm to all columns
   - Let i = i + 1
c. Quantizes a number to its closest coordinate.

Figure 3: V-SVD algorithm.

5. Least-Square Algorithm

In comparison to SVD-V-BLAST, this algorithm needs an extra QR decomposition (Orthogonal Matrix Triangulation); nevertheless, pseudo-inverses are performed on an MxM matrix. The symbol estimation has an extra vector-matrix multiplication, but the size of all other operations depends exclusively on M instead of M and N.

The estimation problem can be seen as solving a system of linear equations perturbed by noise, with the added constraint that the solution must be an element of the modulation constellation. To do this, the algorithm modifies the conventional QR decomposition to solve equation (1) as follows

$$r = QRa + v$$
$$Q^H r = Ra + Q^H v$$
$$x = Ra + v \qquad (4)$$

where **Q** is an NxM matrix with orthonormal columns, **R** is an MxM upper-triangular matrix and v is a Gaussian-noise vector. It is important to note that some multiplications, divisions and square root operations

are transformed into additions, reducing the overall complexity in a considerable manner as compared to the other algorithms [7]. The Least Square algorithm is described in figure 4.

Algorithm V-LS.

a. Compute an NxM orthonormal matrix Q and an MxM upper-triangular matrix R, such that H = QR.
b. Let i = 1, A = R.
c. Repeat M time:
   i. Find $G_i = A^+$
   ii. Let
   $$k_i = \arg\min_j \left\| (G_i)_j \right\|$$
   iii. Let o_i equal to corresponding column in H.
   iv. Remove column k_i of A
   v. Let i = i + 1
d. Let i = 1
e. Repeat L times:
   i. Let $x = Q^H \cdot r_i$
   ii. Repeat M times:
      - Let W equal to row k_i of G_i
      - Let $y = W^H x$
      - Let $\ddot{o}_{i,o} = f_q(y)$
      - Let z equal to column o_i of H
      - Let $x = x - \ddot{o}_{i,o} z$
      - Let i = i + 1

Figure 4: V-Least-Square algorithm.

The computational complexity of this algorithm is

$$Mem = 1.12 * S * M^2 * N * L \quad (3)$$
$$Add = 0.08 * S * M^2 * N * L^2$$
$$Mult = 0.075 * S * M^2 * N * L^2$$
$$Div = 4.3x10^{-3} * S * M^2 * N * L^2$$
$$Srt = 1.45x10^{-4} * S * M^2 * N * L^2$$

6. Sorted QR Algorithm

In this V-BLAST variant, instead of calculating the channel matrix pseudo-inverses, it estimates the length of the rows of a matrix $\mathbf{H}^+$, using the length of columns of matrix **H** ($\mathbf{H}^+$ is orthogonal to the vector space **H**). This algorithm is an extension of the modified Gram-Schmidt algorithm by ordering the columns of **H** in each orthogonalization step [10]. The algorithm applies the Gram-Schmidt algorithm to compute matrix **R** line by line from top to bottom and matrix **Q** column by column from left to right. This is done by computing the elements of matrix **Q** so as to compute the elements of matrix **R** in a recursive manner. For a thorough explanation of this algorithm, the reader is referred to [10].

Algorithm V-SQR

b. Find the sorted-QR decomposition of H store it in Q and R.
c. Let i = 1.
d. Repeat L times:
   - Let $x = Q^H \cdot r_i$
   - Find $\hat{a}_i$ that solves $R\hat{a}=x$
   - Re-arrange $\hat{a}_i$ to correspond to original ordering of columns of H.
   - Let i = i + 1
e. Quantizes a number to its closest coordinate.
f. Reorder estimated symbols.

Figure 5: Sorted QR Algorithm.

The complexity of this algorithm can be estimated as

$$Totmem = 0.5 * S * M^2 * N * L$$
$$Add = 0.035 * S * M^2 * N * L^2$$
$$Mult = 0.033 * S * M^2 * N * L^2$$
$$Div = 1.3x10^{-3} * S * M^2 * N * L^2$$
$$Srt = 3.2x10^{-5} * S * M^2 * N * L^2$$

7. Results

In this section, performance results of the four algorithms are presented. First, performance analysis is presented for Block Error Rate (BLER) as a function of the average signal to noise ratio (SNR_{avg}). A block consists of 10 symbols and is defined as a single transmission [8]. Next, a complexity comparison in terms of arithmetic operations and storage requirements is presented. In this analysis, the number of operations required by each algorithm for estimating one bit of information is called O_b. Simulations were carried out with L = 10, and 16-QAM constellations and were run until 2000 block errors occurred.

7.1 BLER Performance

Table 1 and 2 present a comparison between 8x10 and 8x12 antenna arrays for SNR_{avg}=18 dB. As we can see from these tables, the V-SQR approach has the least computational complexity, but its BLER is considerably degraded. The best BLER is obtained by the ML algorithm, but its computational complexity higher. The algorithm that shows the best complexity-

performance trade-off is the Least-Square algorithm (V-LS).

8x10				
Algorithm	**Run Time**	**Simulated bits**	**BLER**	**Ob**
ML	1092931724	4256960	1.50E-01	3671
V-SVD	1092932090	1429120	4.48E-01	10916
V-LS	1092932880	1429120	4.48E-01	1247
V-SQR	1092932996	1389120	4.61E-01	486

Table 1: Performance analysis.

8x12				
Algorithm	**Run Time**	**Simulated bits**	**BLER**	**Ob**
ML	1092933343	19507840	3.28E-02	3709
V-SVD	1092939360	5627200	1.14E-01	12769
V-LS	1092940990	5627200	1.14E-01	1305
V-SQR	1092941284	5148800	1.24E-01	558

Table 2: Performance analysis.

Figure 6 presents the BLER performance of the V-LS and the ML algorithms for 2x4 and 2x6 MIMO systems. The V-SVD and V-SQR approaches are not shown in the figure since they have the same performance as the V-LS algorithm.

From the figure we can see that the performance of both algorithms is practically the same; nevertheless, the V-LS algorithm is less complex. On the other hand, by introducing another pair of receive antennas, the performance improves significantly since a gain of 4 dB is achieved, which means that we can save transmission power.

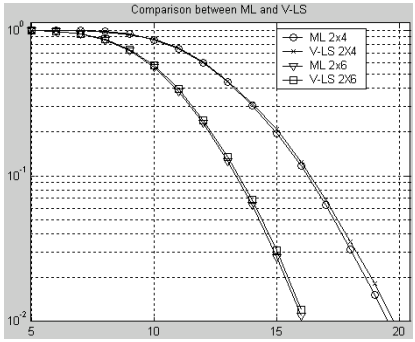

Figure 6: BLER vs SNR_{avg} M = 2; N = 4, 6.

Finally, figure 7 and 8 show simulation results for the ML, V-LS and V-SQR algorithms for 8x10 and 8x12 arrays. It is important to note that the ML algorithm presents a 1dB improvement; however, the other algorithms (V-LS and V-SQR) present a complexity reduction of a factor three and six respectively.

7.2 Computational Complexity

As complexity regards, tables 3 and 4 show the computational requirements of each algorithm. First column indicates the total storage requirements, and the following columns indicate the number of additions, multiplications, divisions and square root operations. All of these for each decoded bit. First row corresponds to the ML algorithm, second row to the V-SVD approach, third row to the V-LS algorithm and fourth row to the V-SQR decomposition.

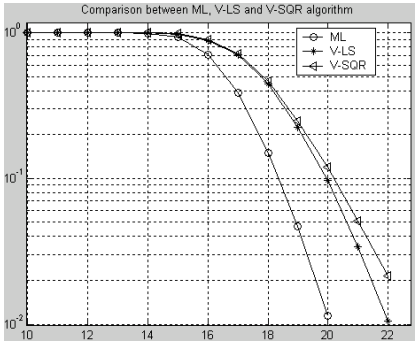

Figure 7: BLER vs SNR_{avg} M = 8; N = 10.

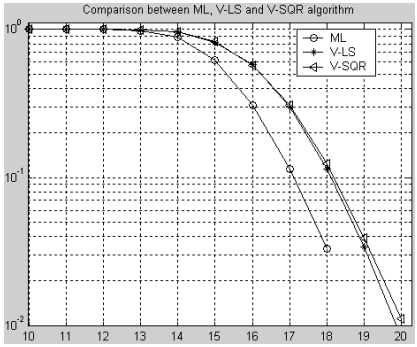

Figure 8: BLER vs SNR_{avg} M = 8; N = 12.

It is clear that the V-LS algorithm has the best trade-off. It is important to note that in this algorithm, multiplications, divisions and square root operations are reduced and replaced by addition operations.

Finally, figures 9 and 10 show the overall computational complexity, O_b, as a function of SNR_{avg} for 8x12 and 8x10 MIMO systems. As we can see, the V-SQR algorithm presents the lowest computational complexity. Nevertheless, its BLER behavior is not quite efficient. The second smallest computational complexity belongs to the V-LS algorithm which,

added to its BLER performance, becomes the best algorithm from a complexity-performance perspective.

8x10					
Algorithm	**Storage**	**Addition**	**Mult**	**Division**	**Sqrt**
ML	1326465318	530498286	427312046	30341945	425696
V-SVD	1183394500	579547260	757481046	25402415	6875846
V-LS	132908160	90606208	85247008	4876872	160776
V-SQR	48723384	38617536	35839296	1389120	34728

Table 3: Complete hardware complexity.

8x12					
Algorithm	**Storage**	**Addition**	**Mult**	**Division**	**Sqrt**
ML	6131084887	2.38E+09	2.042E+09	1.22E+08	1950784
V-SVD	5478406140	2.643E+09	3.447E+09	1.06E+08	2.7E+07
V-LS	546119760	376459680	355357680	19765540	633060
V-SQR	206853040	164504160	154206560	5663680	128720

Table 4: Complete hardware complexity.

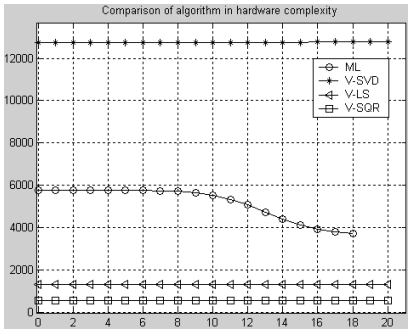

Figure 9: O_b vs. SNR_{avg} M = 8; N = 12.

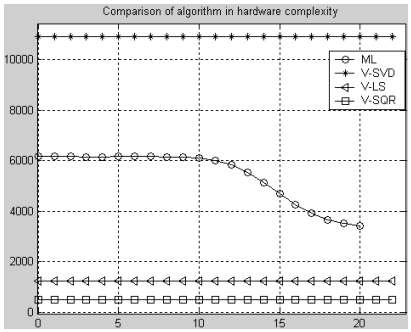

Figure 10: O_b vs. SNR_{avg} M = 8; N = 10.

8. Conclusions and Future Work

In this paper, three MIMO receiver procedures based on the V-BLAST algorithm and one based on the near-optimal ML algorithm have been presented. The performance and complexity of each algorithm have been analyzed and compared in order to determine the algorithm that is best suited for hardware implementation. It has been shown that MIMO systems can achieve much higher data rates than conventional SISO systems. This explains why space-time coding has been considered for standardization purposes in the third generation mobile communication systems (3G) where high rate multimedia services are required. It has also been shown that implementing an algorithm not only has to do with its overall performance but also with its computational burden.

Future work is focused on two aspects, the hardware implementation of the V-LS algorithm in DSP or Programmable Logic Device platforms, and the improvement of the overall system performance by introducing channel coding techniques, either trellis or block, to the MIMO system.

8. References

[1] J. G. Proakis. "Digital Communications", 4th Edition. Mc. Graw Hill, 2001.

[2] B. Vucetic, J. Yuan. "Space-Time Coding", WILEY, 2003.

[3] G. J. Foschini: "Layered space-time architecture for wireless communication in environment when using multiple antennas". Bell Lab. Tech. J., 1996. I. (2), pp. 41-59.

[4] E. Biglieri, J. Proakis, and S. Shamai, I: "Fading channels: Information-theoretic and communications aspects,î *IEEE Trans.Theory*, vol. 44, no. 6, pp. 2619ñ2691, Oct. 1998.

[5] E. Agrell, T. Eriksson, A. Vardy: "Closest point search in Lattices". IEEE Transactions on Information Theory. Vol. 48. No. 8. August 2002, pp. 2201 – 2214.

[6] G. H. Golub, C. F. Van Load: "Matriz Computations". Second Edition. The Johns Hopkins University Press. 1989, pp.427 – 435.

[7] M. Bazdresch, R. Guisantes: "Least-Squares Techniques Applied to VBLAST Receivers". Submitted for Publication.

[8] N. Boubaker, K. B. Letaief and R. D. Murch: "Transmit and Receive Spatial Multiplexing for Broadband Wireless Communitations". IEEE. 2001, pp. 224 – 229.

[9] M. Bazdresch and J. Rodriguez-Guisantes: "The msim simulator". [Online], Available: http://www.comelec.enst.fr/~rodriguez/msim/index.html

[10] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, K. D. Kammeyer: "Efficient algorithm for decoding layered space-Time Codes". Electronics Letters. 25 October 2001. Vol. 37. No. 22

[11] P.W. Wolniansky, G.J. Foschini, G.D. Golden, R.A. Valenzuela; "V-BLAST: An Architecture for Realizing Very High Data Rates Over the Rich-Scattering Wireless Channel". In Proc. Int. Symposium on Avanced Radio Technologies, Boulder, CO. Sept. 10 1998.

[12] E. Viterbo, E. Biglieri, "A universal lattice decoder", in GRETSI 14 ème Colloque, Juan-les-Pins, France, Sept. 1993

Bibliography

[1] E. Biglieri, J. Proakis, and S. Shamai, "Fading channels: Information-theoretic and communications aspects," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2619–2691, Oct. 1998.

[2] P. Smith and M. Shafi, "An approximate capacity distribution for MIMO systems," *IEEE Trans. Commun.*, vol. 52, no. 6, pp. 887–890, June 2004.

[3] I. E. Telatar, "Capacity of multiple-antenna Gaussian channels," *European Trans. Telecomm.*, vol. 10, no. 6, pp. 585–595, 1999.

[4] T. L. Marzetta and B. M. Hochwald, "Capacity of a mobile multiple-antenna communication link in Rayleigh flat fading," *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 139–157, Jan. 1999.

[5] 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, pp. 311–335, Mar. 1998.

[6] I. C. Abou-Faycal, M. D. Trott, and S. Shamai, "The capacity of discrete-time memoryless Rayleigh-fading channels," *IEEE Trans. Inform. Theory*, vol. 47, no. 4, pp. 1290–1301, May 2001.

[7] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[8] J. G. Proakis, *Digital Communications*, 3rd ed. McGraw-Hill, 1995.

[9] V. Tarokh, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communication: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 744–765, Mar. 1988.

[10] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1456–1467, July 1999.

[11] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[12] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block coding for wireless communications: Performance results," *IEEE J. Select. Areas Commun.*, vol. 17, no. 3, pp. 451–460, Mar. 1999.

[13] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Lab. Tech. J.*, vol. 1, no. 2, pp. 41–59, 1996.

[14] 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 J. Select. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[15] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[16] G. Golden, C. Foschini, R. Valenzuela, and P. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," *Electronic Letters*, vol. 35, no. 1, 7th January 1999.

[17] P. Wolniansky, G. Foschini, G. Golden, and R. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering channel," in *Signals, Systems, and Electronics, International Symposium on*, Oct. 1998, pp. 295–300.

[18] S. Loyka and F. Gagnon, "Performance analysis of the V-BLAST algorithm: An analytical approach," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1326–1337, July 2004.

[19] S. Bäro, G. Bauch, A. Pavlic, and A. Semmler, "Improving BLAST performance using space-time block codes and turbo decoding," in *Proc. IEEE Globecom*, 2000, pp. 1067–1071.

[20] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Trans. Signal Processing*, vol. 51, no. 7, pp. 1722–1730, July 2003.

[21] A. Bhargave, R. J. de Figueiredo, and T. Eltoft, "A detection algorithm for the V-BLAST system," in *Proc. GLOBECOM*, 2001, pp. 25–29.

[22] G. Ginis and J. M. Cioffi, "On the relation between V-BLAST and the GDFE," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 364–366, Sept. 2001.

[23] T. Guess, H. Zhang, and T. V. Kotchiev, "The outage capacity of BLAST for MIMO channels," in *Communications, IEEE International Conference on*, vol. 4, 2003, pp. 2628–2632.

[24] S. Loyka, "V-BLAST outage probability: Analytical analysis," in *Vehicular Technology Conference, Proceedings of the 56th*, vol. 4, 2002, pp. 1997–2001.

[25] S. Loyka and F. Gagnon, "Analytical framework for outage and BER analysis of the V-BLAST algorithm," in *Communications, International Zurich Seminar on*, 2004, pp. 120–123.

[26] N. Boubaker, K. B. Letaief, and R. D. Murch, "Transmit and receive spatial multiplexing for broadband wireless communications," in *Proc. 6th IEEE Symposium on Computers and Communications*, 2001, pp. 224–229.

[27] (2004) TMS320C6711 - Floating Point DSP. Texas Instruments. [Online]. Available: http://focus.ti.com/docs/prod/folders/print/tms320c6711.html

[28] W.-J. Choi, R. Negi, and J. M. Cioffi, "Combined ml and dfe decoding for the V-BLAST system," in *Proc. IEEE International Conference on Communications*, 2000, pp. 1243–1248.

[29] R. Fischer and C. Windpassinger, "Real versus complex-valued equalisation in V-BLAST systems," *Electronic Letters*, vol. 39, no. 5, pp. 470–471, Mar. 2003.

[30] X. Jing, H. Wang, C. Ming1, and S. Cheng, "A novel blast detection algorithm based instantaneous error ordering," in *Communications, 2003. IEEE International Conference on,*, 2003, pp. 3056–3060.

[31] D. Shiu and J. M. Kahn, "Layered space-time codes for wireless communications using multiple transmit antennas," in *Proc. IEEE International Conference on Communications*, 1999, pp. 436–440.

[32] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEE Electronic Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[33] R. Böhnke, D. Wübben, V. Kühn, and K.-D. Kammeyer, "Reduced complexity mmse detection for BLAST architectures," in *Proc. GLOBECOM*, 2003, pp. 2258–2262.

[34] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[35] A. K. Lenstra, H. W. L. Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[36] A. Korkine and G. Zolotareff, "Sur les formes quadratiques," *Mathematische Annalen*, vol. 6, pp. 366–389, 1873.

[37] H. Cohen, *A course in computational algebraic number theory*, 1st ed. Springer-Verlag, 1993.

[38] A. Naguib, N. Seshadri, and A. Calderbank, "Increasing data rate over wireless channels," *IEEE Signal Processing Mag.*, pp. 76–92, May 2000.