Signal Detection Strategies and Algorithms for Multiple-Input Multiple-Output Channels

A Thesis
Presented to
The Academic Faculty

by

Deric W. Waters

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2005

Signal Detection Strategies and Algorithms for Multiple-Input Multiple-Output Channels

Date Approved: 15 November 2005

To Mom and Dad.

ACKNOWLEDGEMENTS

I would like to thank my advisor John Barry. His feedback has always been excellent, and my work was best when trying to answer his questions and correct the flaws he pointed out. I was fortunate to have an advisor who gave me complete freedom to work at my own pace on problems that interested me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

BER	Bit-Error Rate
BLAST	Bell Labs LAyered Space-Time
BODF	BLAST-Ordered Decision-Feedback
	interchangeable with optimally-ordered decision-feedback.
CDMA	Code-Division Multiple Access
DF	Decision-Feedback
DOS	DOuble-Sorted lattice reduction
LA-DF	Lattice-Aided Decision-Feedback
LLL	Lenstra-Lenstra-Lov<sz
MIMO	Multiple-Input Multiple-Output
ML	Maximum-Likelihood
MMSE	Minimum Mean-Squared Error
MSE	Mean-Squared Error
NP	Noise Predictive
PDF	Partial Decision-Feedback
QAM	Quadrature Amplitude Modulation
RM	Real Multiplications
SNR	Signal-to-Noise Ratio
ZF	Zero-Forcing

SUMMARY

In today's society, a growing number of users are demanding more sophisticated services from wireless communication devices. In order to meet these rising demands, it has been proposed to increase the capacity of the wireless channel by using more than one antenna at the transmitter and receiver, thereby creating multiple-input multiple-output (MIMO) channels. Using MIMO communication techniques is a promising way to improve wireless communication technology because in a rich-scattering environment the capacity increases linearly with the number of antennas. However, increasing the number of transmit antennas also increases the complexity of detection at an exponential rate. So while MIMO channels have the potential to greatly increase the capacity of wireless communication systems, they also force a greater computational burden on the receiver.

Even suboptimal MIMO detectors that have relatively low complexity, have been shown to achieve unprecedented high spectral efficiency. However, their performance is far inferior to the optimal MIMO detector, meaning they require more transmit power. The fact that the optimal MIMO detector is an impractical solution due to its prohibitive complexity, leaves a performance gap between detectors that require reasonable complexity and the optimal detector. The objective of this research is to bridge this gap and provide new solutions for managing the inherent performance-complexity trade-off in MIMO detection.

The optimally-ordered decision-feedback (BODF) detector is a standard low-complexity detector. The contributions of this thesis can be regarded as ways to either improve its performance or reduce its complexity – or both.

- We propose a novel algorithm to implement the BODF detector based on noise-prediction. This algorithm is more computationally efficient than previously reported implementations of the BODF detector. Another benefit of this algorithm is that it can be used to easily upgrade an existing linear detector into a BODF detector.
- We propose the partial decision-feedback detector as a strategy to achieve nearly the same performance as the BODF detector, while requiring nearly the same complexity as the linear detector.
- We propose the family of Chase detectors that allow the receiver to trade performance for reduced complexity. By adapting some simple parameters, a Chase detector may achieve near-ML performance or have near-minimal complexity. We also propose two new detection strategies that belong to the family of Chase detectors called the B-Chase and S-Chase detectors. Both of these detectors can achieve near-optimal performance with less complexity than existing detectors.
- Finally, we propose the double-sorted lattice-reduction algorithm that achieves near-optimal performance with near-BODF complexity when combined with the decision-feedback detector.

CHAPTER 1

Problem Introduction and Motivation

The means by which people communicate have been revolutionized with the advent of wireless communication technology. As more and more people begin to use different wireless communication technologies, service providers need to improve the reliability and throughput of their systems. For example, most local area networks (LANs) are built on wired infrastructure, but wireless LANs can provide a degree of mobility and freedom that make them an attractive alternative. However, before their large-scale adoption, wireless LAN technologies must improve their reliability and throughput.

The reliability and throughput of wireless communication systems can be improved by adding multiple antennas at the transmitter and receiver to create multiple-input multiple-output (MIMO) channels. Narrowband MIMO channels with rich scattering have greater *potential* throughput than conventional single-input single-output channels because the capacity of the MIMO channel increases linearly as the number of transmit and receive antennas increases [21]. MIMO channels also provide greater reliability because the probability of all the subchannels between the transmitter and receiver fading at the same time decreases exponentially as antennas are added.

The problem of data throughput comes down to a problem of spectral efficiency. If unlimited bandwidth were available, then wireless systems would have no problem accommodating any number of users demanding high quality service. However, the availability of frequency spectrum has physical and legal restrictions. This means that wireless communication systems need to use the radio spectrum more efficiently in order to increase their throughput. By employing multiple antennas at the transmitter and receiver, MIMO systems can greatly increase spectral efficiency while decreasing the total transmit power.

Of course, the benefits of using multiple antennas at the transmitter and receiver do not come without costs. One fundamental obstacle for MIMO systems is the increased complexity of recovering the transmitted information. As the capacity increases linearly with the number of antennas, the complexity of the detection problem increases exponentially with the number of transmit antennas. As a result, the maximum-likelihood (ML) detector, which finds the best symbol vector from among an exponential number of possibilities, is prohibitively complex even for small numbers of channel inputs. Suboptimal detectors can achieve the same spectral efficiency as the ML detector, but they need more transmit power to do so. In fact, the performance of MIMO detectors is measured by the amount of transmit power, or signal-to-noise ratio (SNR), they require to recover the transmitted data. The ML detector has optimal performance, but requires exponential complexity in return. Some suboptimal detectors require only linear complexity, but they cannot achieve optimal performance. This gives rise to an inherent trade-off between performance and complexity in MIMO detection.

The objective of this research is to investigate new detection techniques which make the realization of MIMO systems more practical by improving the performance-complexity trade-off of MIMO detectors. The Bell Labs Layered Space-Time (BLAST) MIMO system [21] has demonstrated that MIMO systems are feasible that can dramatically increase spectral efficiency with reasonable complexity. However, the performance and complexity gaps between the optimal detector and the suboptimal detector used in the BLAST system are enormous, which leaves substantial room for improvement.

Left with the choice between the ML and low-complexity detectors, a MIMO system designer will quickly discover that the detector is either the predominant source of complexity in the system, or else the predominant source of performance loss. This thesis shrinks the performance gap between low-complexity detectors and the ML detector.

In this chapter, we describe the MIMO channel model and introduce two standard ways to perform detection. Specifically, we define optimal MIMO detection as well as a low-complexity detector called the linear detector. The huge performance and complexity gaps between these detectors will be shown to illustrate the fundamental trade-off between performance and complexity in MIMO detection. This introduction also provides background for later chapters, which describe better ways to perform MIMO detection.

1.1. Channel Model

In this thesis we consider the problem of communicating over narrowband memoryless MIMO channels with rich-scattering. Two primary applications of this channel are wireless point-to-point [21] and code-division multiple-access (CDMA) [16] communications. In point-to-point systems the antennas at both the transmitter and receiver are located nearby each other, but separated by at least half a wavelength to insure independent fading. In CDMA systems, users are separated geographically, but since their signals have a common destination they are grouped together to form a single transmitter that has multiple antennas.

Figure 1-1 illustrates a MIMO channel. Mathematically, the memory-less channel with N inputs $\boldsymbol{a} = [a_1, \dots a_N]^T$ and M outputs $\boldsymbol{r} = [r_1, \dots r_M]^T$ can be written as:

$$\boldsymbol{r} = \mathbf{H}\boldsymbol{a} + \boldsymbol{w}, \tag{1-1}$$

where $\mathbf{H} = [\boldsymbol{h}_1, \dots \boldsymbol{h}_N]$ is a complex $M \times N$ channel matrix whose i-th column is $\boldsymbol{h}_i$, and where $\boldsymbol{w} = [w_1, \dots w_M]^T$ is noise. We assume that the columns of $\mathbf{H}$ are linearly independent, which implies $M \geq N$. We assume that the noise is uncorrelated such that $\mathrm{E}[\boldsymbol{w}\boldsymbol{w}^*] = \sigma^2\mathbf{I}$, where $\boldsymbol{w}^*$ denotes the conjugate transpose of $\boldsymbol{w}$, $\mathbf{I}$ is the $N \times N$ identity matrix, and σ^2 is the variance of the noise. Further, we assume that the inputs are uncorrelated and chosen from the same unit-energy alphabet $\mathcal{A}$, so that $\mathrm{E}[\boldsymbol{a}\boldsymbol{a}^*] = \mathbf{I}$.

The memoryless MIMO channel is often simulated using Rayleigh fading. In other words, the real and imaginary parts of each element of the channel matrix $\mathbf{H}$ are independently and identically distributed Gaussian random variables with a variance of one half. Rayleigh fading accurately describes a rich scattering channel with antenna elements that are separated by at least half a wavelength [39]. In the numerical simulations throughout this thesis, the MIMO channel is simulated as a Rayleigh-fading channel.

In some cases it is beneficial to represent the MIMO detection problem in terms of a real channel, with real-valued inputs and outputs. The channel model (1-1) can be converted into a real-valued expression as follows [20]:

$$\begin{bmatrix} \Re\boldsymbol{r} \\ \Im\boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \Re\mathbf{H} & -\Im\mathbf{H} \\ \Im\mathbf{H} & \Re\mathbf{H} \end{bmatrix} \begin{bmatrix} \Re\boldsymbol{a} \\ \Im\boldsymbol{a} \end{bmatrix} + \begin{bmatrix} \Re\boldsymbol{w} \\ \Im\boldsymbol{w} \end{bmatrix}, \tag{1-2}$$

where $\Re$ and $\Im$ represent the real and imaginary parts, respectively. This real-valued channel model is denoted as:

$$\boldsymbol{r}_R = \mathbf{H}_R\boldsymbol{a}_R + \boldsymbol{w}_R. \tag{1-3}$$

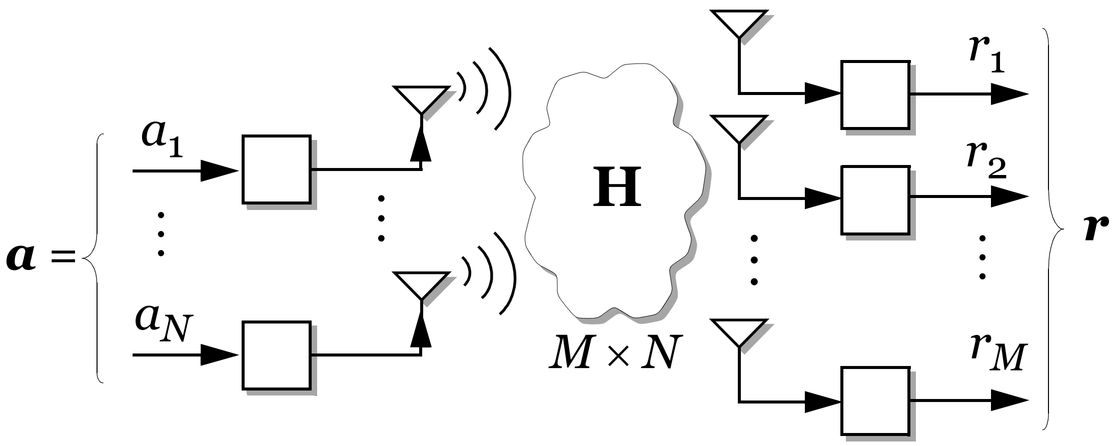

Figure 1-1. Illustration of the MIMO channel.

1.2. Optimal Detection

The ML detector is defined from the likelihood of observing $\boldsymbol{r}$ given that the vector $\hat{\boldsymbol{a}}$ was transmitted. As the name implies the maximum-likelihood detector finds the decision vector $\hat{\boldsymbol{a}}$ which maximizes this likelihood function. If the channel is known to the receiver, and the noise is zero-mean Gaussian the likelihood function is defined as:

$$p(\boldsymbol{r}|\hat{\boldsymbol{a}}) = \frac{1}{(\pi\sigma^2)^N}\exp\left(-\frac{\|\boldsymbol{r} - \mathbf{H}\hat{\boldsymbol{a}}\|^2}{\sigma^2}\right). \tag{1-4}$$

The search for the ML decision vector can be formalized succinctly as:

$$\hat{\boldsymbol{a}} = \underset{\tilde{\boldsymbol{a}} \in \mathcal{A}^N}{\operatorname{argmin}} \; \| \boldsymbol{r} - \mathbf{H}\tilde{\boldsymbol{a}} \|^2, \tag{1-5}$$

where $\mathcal{A}^N$ is the set of all possible transmit vectors. The ML detector is straightforward, but a brute-force implementation of (1-5) quickly becomes prohibitive as N or $|\mathcal{A}|$ increases, where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$.

1.3. Linear Detection

The simplest MIMO detector is the zero-forcing linear detector [41], which simply inverts the channel matrix. For the case when the inverse of the channel does not exist, the pseudoinverse of the channel matrix is used. The linear detector begins by multiplying the channel output by the channel matrix pseudoinverse:

$$\begin{aligned} \boldsymbol{y} &= (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\boldsymbol{r} \\ &= \boldsymbol{a} + (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\boldsymbol{w}\,. \end{aligned} \tag{1-6}$$

A slicer is used to make a decision regarding the k-th channel input. The slicer chooses the element from the symbol alphabet nearest y_k:

$$\begin{aligned} \hat{a}_k &= \underset{a\,\in\,\mathcal{A}}{\text{argmin}} \ \| y_k - a \|^2 \\ &= \text{dec}\{y_k\}. \end{aligned} \tag{1-7}$$

The linear detector performs poorly when the channel matrix is close to being singular because it amplifies the noise. On the other hand, when the channel matrix is orthogonal the linear detector does not amplify the noise, and is equivalent to the ML detector.

1.4. The Performance-Complexity Trade-Off

Performance of MIMO detectors is measured in decibels (dB) of SNR. The SNR of a system is directly proportional to the transmit power, which is directly related to the cost of transmission in the communication system. With enough transmit power, any MIMO detector can achieve a small probability of bit error, or bit-error rate (BER). However, transmit power is expensive, so the detector's goal is to minimize the amount of SNR required to reach the necessary BER. In order to quantify the amount of signal power needed to effectively communicate each bit across the channel, we measure the SNR as $E[\|\mathbf{H}\boldsymbol{a}\|^2] \,/\, (\, E[\|\boldsymbol{w}\|^2] \log_2|\mathcal{A}|)$.

Ideally, the complexity metric for MIMO detection should provide a universal measure of how fast a particular MIMO detector can operate, as well as how expensive it is to implement. Unfortunately, such a universal complexity metric is impossible to define because the speed and cost of a given detector depends upon how it is actually implemented. For example, the cost and speed of implementing an algorithm on a digital signal processor is different from the cost of implementing it in hardware. However, simply counting the number of multiplications required by a given detector gives a reasonable indication of how costly it would be to implement.

The total complexity of a MIMO detector is divided into *preprocessing* and *core-processing* complexity. The preprocessing complexity includes those computations which are performed only once for a given channel matrix. Once the channel estimation is updated or changed, the preprocessing computations need to be recalculated. The core-processing complexity includes only those computations that are necessary for every symbol period. The faster the channel changes, the more important it becomes to reduce preprocessing complexity. On the other hand, if the channel changes slowly then the preprocessing contributes relatively little to the total complexity, and reducing the core-processing complexity is most important.

There is a fundamental trade-off in MIMO detection systems between performance and complexity. Although the brute-force ML detector (1-5) is conceptually simple and achieves optimal performance, it is impractical due to its high core-processing complexity.

On the other hand, the linear detector (1-7) has low core-processing complexity, but its performance is far from optimal. The enormous gap in performance between the ML and linear detectors is illustrated in Figure 1-2, where the ML detector requires about 17 dB less SNR to reach a BER of 10^{-3} than the linear detector. At the same time, the brute-force ML detector may require as many as $N|\mathcal{A}|^N$ multiplications, while the linear detector needs only $3MN$ multiplications. This means that to achieve the performances shown in Figure 1-2, the ML detector could need more than 5000 times as many multiplications as the linear detector.

The objective of this research is to investigate new detection techniques for MIMO channels whose performance is close to that of the ML detector, and whose complexity is near that of the linear detector.

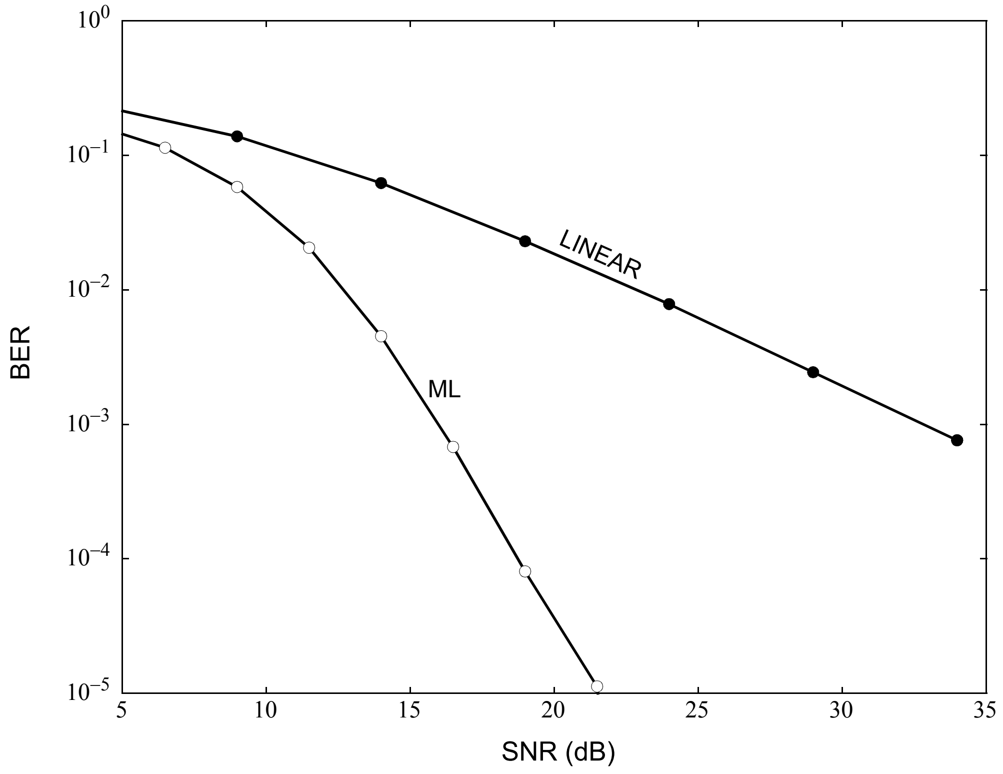

Figure 1-2. Performance of the ML and linear detectors as averaged over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs.

1.5. Thesis Outline

Chapter 2 gives an extensive review of existing MIMO detection techniques. In particular, one low-complexity that plays a central role in this thesis is the decision-feedback (DF) detector, which is introduced in this chapter.

The new contributions of this thesis are presented in detail in the next five chapters.

- Chapter 3 describes how implementing the optimally-ordered DF detector using noise prediction can reduce its complexity.
- Chapter 4 introduces the partial DF detector which is less complex than the optimally-ordered DF detector and has only a small performance penalty.
- Chapter 5 describes the Chase family of detectors that allow the receiver to trade performance for complexity by adjusting a single parameter. This chapter also uses the Chase detector framework to introduce new low-complexity detectors that achieve near-ML performance.
- Chapter 6 investigates lattice-aided DF detection, and describes the double-sorted (DOS) lattice-reduction algorithm. It is shown that the combination of DOS lattice reduction and DF creates a detector that achieves near-ML performance with low complexity.

Chapter 7 draws some final conclusions and summarizes the contributions of this research.

CHAPTER 2

State-of-the-Art MIMO Detection

The main topic of this research is to find detectors whose performance is as close to that of the maximum-likelihood (ML) detector as possible, and whose complexity is as low possible. This problem has been addressed extensively in the literature, and our goal is to improve performance and/or reduce the complexity of existing detectors. This chapter describes the state-of-the-art in MIMO detection, then subsequent chapters describe new detection techniques. First, Section 2.1 describes a simple way to improve upon linear detection called the decision-feedback (DF) detector. Then the next three sections describe ways to improve the performance of the DF detector. Specifically, Section 2.2 presents how to improve the DF detector by choosing the order in which the symbols are detected. Next, Section 2.3 describes another way to improve the performance of not only the DF detector, but also the linear detector by replacing the zero-forcing design criterion with its minimum mean-squared error (MMSE) counterpart. In many cases, the MMSE versions of the detectors outperform the zero-forcing versions significantly with only a small complexity increase. Section 2.3 introduces the MMSE channel model, and shows how it simplifies the implementation of MMSE detectors. Section 2.4 introduces a detection technique called lattice-aided detection, which can achieve near-ML performance with low complexity. Besides improving the performance of the DF detector, another approach to achieving a better performance-complexity trade-off is to reduce the complexity of the ML detector. Section 2.5 shows how to reduce the complexity of the ML detector using sphere detection. Finally, Section 2.6 discusses ways to sacrifice performance in order to reduce the complexity of the sphere detector.

2.1. Decision-Feedback Detection

A standard low-complexity detector first proposed in the context of multiuser detection for CDMA systems [16] is the decision-feedback (DF) detector, also known as the successive interference canceller. In short, the DF detector uses nonlinear feedback to reduce the noise enhancement suffered by the linear detector.

Figure 2-1 shows a block diagram of the DF detector of the *conventional zero-forcing (ZF) DF detector*. This detector is based on the QR decomposition [24] of the channel:

$$\mathbf{H} = \mathbf{QDM}, \tag{2-1}$$

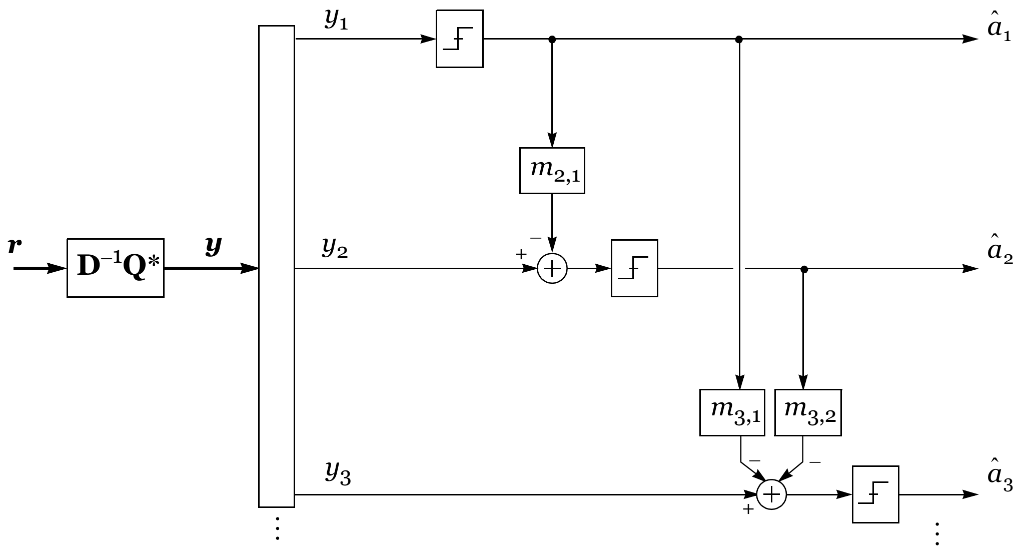

Figure 2-1. The conventional decision-feedback detector.

where $\mathbf{Q} = [\boldsymbol{q}_1, \dots \boldsymbol{q}_M]$ is an $M \times N$ matrix with orthonormal columns, where $\mathbf{D}$ is a $N \times N$ diagonal matrix with diagonal elements that are positive and real, and where $\mathbf{M}$ is a lower triangular matrix with ones along the diagonal.

The DF detector first applies a forward filter $\mathbf{D}^{-1}\mathbf{Q}^*$ to the received vector $\boldsymbol{y} = \mathbf{D}^{-1}\mathbf{Q}^*\boldsymbol{r}$, yielding:

$$\boldsymbol{y} = \mathbf{M}\boldsymbol{a} + \mathbf{D}^{-1}\mathbf{Q}^*\boldsymbol{w}. \tag{2-2}$$

The i-th element of $\boldsymbol{y}$ is thus:

$$y_i = a_i + \sum_{j<i} m_{i,j} a_j + \boldsymbol{q}_i^* \boldsymbol{w} \,/\, d_{i,i}, \tag{2-3}$$

where $m_{i,j}$ is the element from the i-th row and j-th column of the matrix $\mathbf{M}$. Since $\mathbf{M}$ is lower triangular, y_1 is free of interference. As a result, the decision $\hat{a}_1$ can be found directly by quantizing y_1 to the nearest element in the symbol alphabet $\mathcal{A}$. Using this decision, the interfering term can be subtracted from y_2. Proceeding iteratively, the ZF-DF detector is succinctly defined by the following recursion:

$$\hat{a}_k = dec\left\{ y_k - \sum_{j<k} m_{k,j} \hat{a}_j \right\}. \tag{2-4}$$

The performance of the DF detector is best understood by comparing the SNR of each symbol at the input to the slicer. From (2-3) and (2-4), the input to the slicer is written as:

$$y_i = a_i + \sum_{j<i} m_{i,j}(a_j - \hat{a}_j) + \boldsymbol{q}_i^* \boldsymbol{w} \,/\, d_{i,i}. \tag{2-5}$$

If the interference is cancelled perfectly, then the SNR of the i-th symbol is $d_{i,i}^2/\sigma^2$. For Rayleigh-fading channels, the first symbol almost always has the weakest SNR since $d_{i,i}^2$ is a Chi-squared random variable with $2i$ degrees of freedom. This means that the first symbol is the most likely to be detected incorrectly, making it the performance bottleneck for the DF detector.

Figure 2-2 shows a recursive implementation for the DF detector which is functionally equivalent to the conventional DF detector already discussed. This approach is preferable in some cases because it does not require knowledge of the matrix $\mathbf{M}$. The forward filter is denoted as $\mathbf{F} = \mathbf{D}^{-1}\mathbf{Q}^*$, and its k-th row is $\boldsymbol{f}_k$. The *recursive-DF* detector applies the forward filter one row at a time. Specifically, the receiver first computes $z_1 = \boldsymbol{f}_1\boldsymbol{r}$:

$$z_1 = a_1 + \boldsymbol{q}_1{}^*\boldsymbol{w} \,/\, d_{1,1}. \tag{2-6}$$

The decision regarding a_1 is computed by passing y_1 through a symbol slicer, $\hat{a}_1 = dec\{z_1\}$. Next, the receiver recreates the channel interference in order to cancel it:

$$\boldsymbol{r}_1 = \boldsymbol{r} - \boldsymbol{h}_1\hat{a}_1\,. \tag{2-7}$$

After this interference cancellation, the receiver repeats the same process. Specifically, to detect the second symbol the receiver computes $z_2 = \boldsymbol{f}_2\boldsymbol{r}_1$, if $a_1 = \hat{a}_1$ then z_2 reduces to:

$$z_2 = a_2 + \boldsymbol{q}_2{}^*\boldsymbol{w} \,/\, d_{2,2}, \tag{2-8}$$

and the decision regarding the second symbol is $\hat{a}_2 = dec\{z_2\}$. This procedure continues until the receiver has detected all the symbols.

2.2. Controlling Symbol-Detection Order to Improve the DF Detector

One way to improve the performance of the DF detector is to control the order in which the symbols are detected. Since all the symbols arrive at the receiver simultaneously the receiver may detect them in any order. Since the first symbol limits the performance of the DF detector, it is easy to see that the DF detector performance will

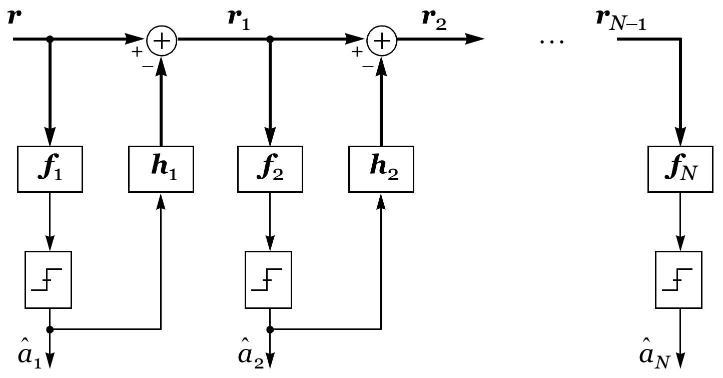

Figure 2-2. The recursive decision-feedback detector.

improve if we detect the symbol with the strongest SNR first. Most proposed detection orderings depend only on the channel matrix **H**, but performance can be improved by choosing a detection ordering that also depends on the channel output [37].

Like the DF detector, the ordered-DF detector is based on the QR decomposition of the channel matrix (2-1). The difference is that controlling the detection order is equivalent to permuting the columns of the channel matrix with a permutation matrix Π. This column permutation creates a new channel model:

$$\begin{aligned} \boldsymbol{r} &= \mathbf{H}\Pi\,\Pi^{*}\boldsymbol{b} + \boldsymbol{w} \\ &= \tilde{\mathbf{H}}\boldsymbol{b} + \boldsymbol{w}, \end{aligned} \tag{2-9}$$

where $\boldsymbol{b} = \Pi^{*}\boldsymbol{a}$ is the new channel input, and $\tilde{\mathbf{H}} = \mathbf{H}\Pi$ is the new channel matrix. The ordered-DF detector is implemented in basically the same way as the DF detector except that it uses this new channel model. The QR decomposition has the same form as for the DF detector (2-1):

$$\mathbf{H}\Pi = \mathbf{QDM}. \tag{2-10}$$

After this decomposition, the ordered-DF detector first makes decisions regarding $\boldsymbol{b}$ in the same way the DF detector made decisions regarding $\boldsymbol{a}$ (see (2-2)–(2-4)). It must then transform those decisions regarding $\boldsymbol{b}$ into decisions regarding $\boldsymbol{a}$ by reversing the channel permutation, $\hat{\boldsymbol{a}} = \Pi\hat{\boldsymbol{b}}$. Obviously the DF detector is a special case of the ordered-DF detector where the permutation matrix is simply the identity matrix $\Pi = \mathbf{I}$.

The ordered-DF detector introduces the new problem of finding the best permutation matrix Π. This could be a difficult problem since there are $N!$ possible permutations, but it was shown in [22] that the greedy algorithm which recursively chooses the symbol with the largest SNR is optimal. This so-called BLAST ordering [22] computes the permutation matrix in an optimal way because it maximizes the minimum SNR of the symbols. The BLAST ordering effectively strengthens the weakest link in the system by only increasing the preprocessing complexity. The original BLAST-ordering algorithm required $\mathcal{O}(N^4)$ multiplications [23], but it can also be computed with only $\mathcal{O}(N^3)$ multiplications [3][25][50][52]. Appendix A gives a reduced-complexity version of the original BLAST-ordering algorithm.

The impact of ordering on the performance of the DF detector is illustrated by Figure 2-3, where using the BLAST ordering instead of natural ordering leads to about a 4 dB improvement in the performance of the zero-forcing DF detector.

Some suboptimal symbol orderings have been proposed to reduce the complexity of computing the permutation matrix [43][47]. The sorted-QR decomposition [47] is an example of a low-complexity way to compute a detection ordering that performs worse than the BLAST ordering but still much better than the natural ordering. The main advantage of using the sorted-QR decomposition is that it has almost the same preprocessing complexity as the conventional QR decomposition, or about half as much preprocessing complexity as computing the BLAST ordering. Appendix B gives pseudocode for a lower-triangular and an upper-triangular sorted-QR decomposition.

2.3. The MMSE Channel Model

An easy way to improve the performance of low-complexity detectors without increasing core complexity is to design them using the minimum mean-squared error (MMSE) criterion. At the receiver, the intersymbol interference is detrimental to performance. The zero-forcing (ZF) detector solves this problem by completely cancelling out all interference. However, in doing so the ZF detector throws away some useful signal energy. In contrast, the MMSE detector will leave some low-power interference if it can capture more signal energy in doing so. By balancing the trade-off between cancelling interference and maximizing signal energy, the MMSE detector outperforms the ZF detector.

We illustrate the difference between ZF and MMSE detectors using the linear detector as an example. ZF and MMSE linear detectors use different criteria to minimize the mean-squared error (MSE). The MSE is defined as:

$$MSE = \|\mathbf{C}\boldsymbol{r} - \boldsymbol{a}\|^2, \tag{2-11}$$

where $\mathbf{C}$ is an $M \times N$ linear filter. The MMSE detector chooses $\mathbf{C}$ to minimize the MSE without any constraint [41]:

$$\mathbf{C} = (\mathbf{H}^*\mathbf{H} + \sigma^2\mathbf{I})^{-1}\mathbf{H}^*. \tag{2-12}$$

The ZF detector has less freedom to choose $\mathbf{C}$ because it requires that $\mathbf{CH} = \mathbf{I}$ in order to completely cancel interference:

$$\mathbf{C} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^* . \tag{2-13}$$

Unless there no noise ($\sigma^2 = 0$) the MMSE linear detector has smaller MSE than the ZF linear detector, and this translates into a performance improvement.

As the SNR tends to infinity, the MMSE and ZF versions of a given detector converge. A practical implication of this is that MMSE detectors perform better for small quadrature-amplitude modulation (QAM) alphabets. For example, for 64-QAM inputs the MMSE and ZF decision-feedback detectors have almost the same performance, but for 4-QAM inputs the MMSE DF detector achieves a significant performance improvement.

MMSE detectors can be explained using a simple modification to the channel model (1-1). The MMSE channel model is based upon the extended channel matrix [6][25]:

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \hat{\sigma}\mathbf{I}_{NxN} \end{bmatrix}, \tag{2-14}$$

where $\hat{\sigma}^2$ is the receiver's estimate of the noise variance σ^2. The output of this new channel model is $\bar{\boldsymbol{r}} = [\boldsymbol{r}^T, 0_{1\times N}]^T$:

$$\bar{\boldsymbol{r}} = \overline{\mathbf{H}}\boldsymbol{a} + \overline{\boldsymbol{w}} . \tag{2-15}$$

The MMSE versions of the linear, DF, and ordered-DF detectors are defined by applying the corresponding zero-forcing detectors to this new channel model. For example, the linear detector (1-6) is obtained by multiplying $\bar{\boldsymbol{r}}$ by the pseudoinverse of $\overline{\mathbf{H}}$, then quantizing the result to the nearest symbol in the alphabet:

$$\hat{\boldsymbol{a}} = \text{dec}\{(\overline{\mathbf{H}}^*\overline{\mathbf{H}})^{-1}\overline{\mathbf{H}}^*\bar{\boldsymbol{r}}\}, \tag{2-16}$$

where dec$\{\boldsymbol{x}\}$ quantizes each of the elements in $\boldsymbol{x} = [x_1, \ldots x_N]^T$ to the nearest symbol in $\mathcal{A}$ in the Euclidean distance sense. The ordered-DF detector is implemented as already described in (2-2)–(2-4), except that the QR decomposition and permutation matrix are computed based on $\overline{\mathbf{H}}$ instead of $\mathbf{H}$:

$$\overline{\mathbf{H}}\Pi = \mathbf{QDM}. \tag{2-17}$$

Computing this QR decomposition, including the permutation matrix, requires more computations than (2-10) because $\overline{\mathbf{H}}$ has larger dimensions than $\mathbf{H}$. But the remainder of the MMSE version of the ordered-DF detector requires exactly the same complexity as the ZF ordered-DF detector. In fact, a ZF detector is just a special case of its MMSE counterpart, because if the receiver estimates the noise variance as $\hat{\sigma}^2 = 0$ then the MMSE QR decomposition (2-17) reduces to the ZF QR decomposition (2-10).

Figure 2-3 shows the performance improvement achieved by the MMSE versions of the BODF and DF detectors over 4-input 4-output Rayleigh-fading channels with 16-QAM inputs. The MMSE version of the BODF detector outperforms its ZF version by about 4 dB. This is a significant performance improvement that is achieved with a small increase in preprocessing complexity, but with no additional core-processing complexity.

2.4. Lattice-Aided Detection

A new approach to solving the detection problem is created by viewing the channel output as a point in the lattice generated by the channel matrix. This approach helps the detector because the matrix that generates this lattice is not unique, and the receiver can find "better" matrices that generate the same lattice. Lattice-aided detectors achieve near-ML performance by using a lattice-reduction algorithm (such as the LLL algorithm [31][34][44][46]) to create a more orthogonal *effective* channel. However, finding the best lattice-reduction is in general an NP-complete problem, and the viability of lattice-aided

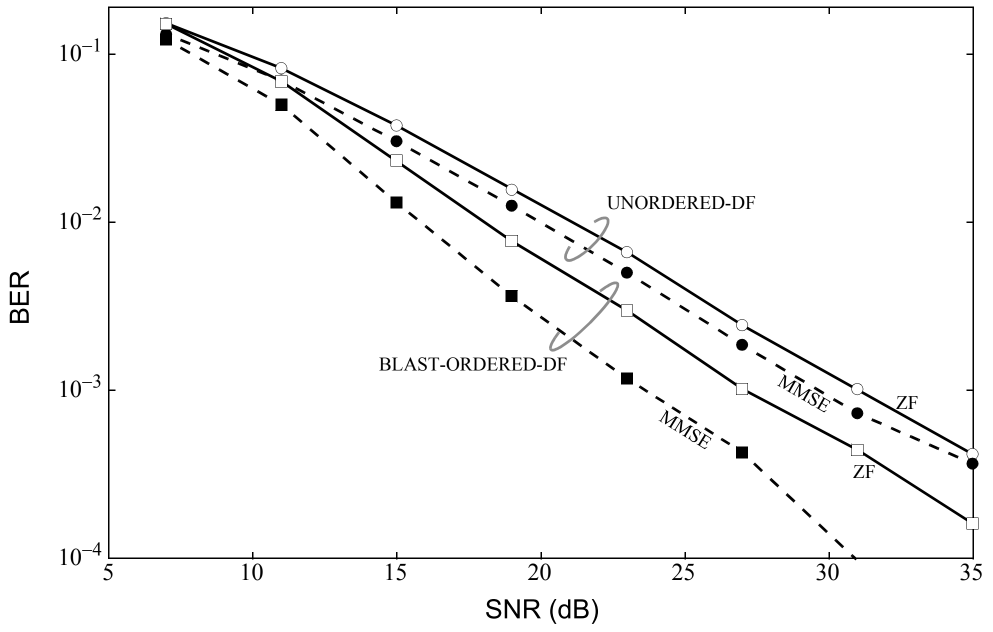

Figure 2-3. Performance of the MMSE and ZF versions of the DF detector with and without ordering. Results averaged over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs.

detection is limited in practice by the high complexity of lattice-reduction algorithms. Particularly on wireless channels that vary rapidly with time, the high overhead of lattice reduction can waste much of the computational savings.

In this subsection we will introduce a general framework to describe lattice-aided detection. The standard approach for implementing lattice-aided detector has been to use the real channel model (1-3), because then the detection problem becomes a search for the nearest point in an integer lattice. However, lattice-aided MIMO detection can just as easily be defined in terms of the complex channel, often resulting in less complex detectors [34]. We will describe lattice-aided detection for complex channels, but our discussion is also valid for real channels.

A *complex integer* is defined as a complex number whose real and imaginary parts are both integers. A *complex lattice* is defined as the set of all linear combinations of a set of linearly independent basis vectors $\{\boldsymbol{b}_1, \dots \boldsymbol{b}_N\}$ with complex integer coefficients, where N is the lattice *dimension*. In terms of the matrix $\mathbf{B} = [\boldsymbol{b}_1, \dots \boldsymbol{b}_N]$, the lattice points can be written as $\mathbf{B}\boldsymbol{x}$ where $\boldsymbol{x}$ is a vector of complex integers.

The basis for a lattice is not unique. If $\mathbf{B}$ is a basis, the product $\mathbf{BT}$ will also be a basis whenever $\mathbf{T}$ is an $N \times N$ unimodular matrix; i.e., whenever $\mathbf{T}$ and $\mathbf{T}^{-1}$ have complex integer entries. Trivial examples of unimodular matrices include the identity matrix and permutation matrices. Lattice reduction is a technique for finding a unimodular $\mathbf{T}$ matrix that transforms one basis into another, usually with the goal of making the new basis as orthogonal as possible.

One constraint of lattice-aided detection is that the symbol alphabet must contain only complex integers. This rules out the use of some phase-shift keying (PSK) alphabets. For lattice-aided detection, we assume that the inputs are chosen from the same QAM alphabet $\mathcal{A} = \{\pm c, \pm 3c, \dots \pm(\sqrt{q}-1)c\} + \sqrt{-1}\{\pm c, \pm 3c, \dots \pm(\sqrt{q}-1)c\}$, where $c = \sqrt{1.5/(q-1)}$ such that $\mathrm{E}[\boldsymbol{a}\boldsymbol{a}^*] = \mathbf{I}$. Since this alphabet has elements that are not integers, the output of the channel must be scaled and shifted such that the effective alphabet lies on a subset of the complex lattice. The result of this scaling and shifting is denoted as $\tilde{\boldsymbol{r}}$, and is called the effective channel output:

$$\tilde{\boldsymbol{r}} = \bar{\boldsymbol{r}}/(2c) - \overline{\mathbf{H}}\boldsymbol{s}, \tag{2-18}$$

where $\bar{\boldsymbol{r}}$ is the output of the MMSE channel model (2-15), and where $\boldsymbol{s} = 0.5(1+\sqrt{-1})[1, \ldots, 1]^T$. The effective channel output further reduces to:

$$\tilde{\boldsymbol{r}} = \overline{\mathbf{H}}\boldsymbol{b} + \tilde{\boldsymbol{w}}, \tag{2-19}$$

where the input vector has been transformed $\boldsymbol{b} = \boldsymbol{a}/(2c) - \boldsymbol{s}$, and $\tilde{\boldsymbol{w}}$ is the noise of the effective channel. The benefit of operating on $\tilde{\boldsymbol{r}}$ instead of $\bar{\boldsymbol{r}}$ is that the real and imaginary parts of $\boldsymbol{b}$ belong to the set of integers $\{-\sqrt{q}/2, -\sqrt{q}/2+1, \ldots \sqrt{q}/2-1\}$. Therefore, recovering $\boldsymbol{b}$ can be seen as a closest point lattice search since $\overline{\mathbf{H}}\boldsymbol{b}$ is a point in the N dimensional complex lattice generated by the columns of $\overline{\mathbf{H}}$.

The underlying principle behind lattice-aided detection is the creation of an effective channel matrix $\tilde{\mathbf{H}}$, whose columns are more orthogonal than $\overline{\mathbf{H}}$. For any unimodular matrix **T**, the effective channel model becomes:

$$\begin{aligned}\tilde{\boldsymbol{r}} &= \overline{\mathbf{H}}\mathbf{T}\mathbf{T}^{-1}\boldsymbol{b} + \tilde{\boldsymbol{w}} \\ &= \tilde{\mathbf{H}}\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{w}},\end{aligned} \tag{2-20}$$

where the effective channel matrix is $\tilde{\mathbf{H}} = \overline{\mathbf{H}}\mathbf{T}$, and the effective channel input is $\tilde{\boldsymbol{b}} = \mathbf{T}^{-1}\boldsymbol{b}$. Since $\mathbf{T}^{-1}$ and $\boldsymbol{b}$ contain only complex integers, the elements of $\tilde{\boldsymbol{b}}$ are also complex integers.

When DF detection is applied to this new effective channel model, it is called the lattice-aided decision-feedback (LA-DF) detector. The LA-DF detector can be implemented following the conventional DF process (2-2)–(2-4). First, the receiver computes the QR decomposition of the effective channel matrix:

$$\tilde{\mathbf{H}} = \mathbf{QDM}, \tag{2-21}$$

where $\mathbf{Q} = [\boldsymbol{q}_1, \dots \boldsymbol{q}_M]$ is an $M \times N$ matrix with orthonormal columns, where $\mathbf{D}$ is a $N \times N$ diagonal matrix with diagonal elements that are positive and real, and where $\mathbf{M}$ is a lower triangular matrix with ones along the diagonal.

Following the QR decomposition, the receiver multiplies the effective channel output by a front-end filter $\boldsymbol{y} = \mathbf{D}^{-1}\mathbf{Q}^* \tilde{\boldsymbol{r}}$, which reduces to:

$$\boldsymbol{y} = \mathbf{M}\tilde{\boldsymbol{b}} + \boldsymbol{n}. \tag{2-22}$$

Although $\boldsymbol{n}$ contains residual intersymbol interference, it is treated as noise. After this front-end filter, the decision regarding $\tilde{b}_k$ is made after removing the interference due to $\tilde{b}_1, \dots, \tilde{b}_{k-1}$ according to:

$$\hat{b}_k = slicer\left\{ y_k - \sum_{j=1}^{k-1} m_{k,j}\hat{b}_j \right\}, \tag{2-23}$$

where $slicer\{x\}$ is the element in the set $\mathcal{L}_k$ nearest x. The set $\mathcal{L}_k$ is defined as the subset of all possible vectors $\{\tilde{\boldsymbol{b}}\}$ whose first $k-1$ elements are equal to $\hat{b}_1, \dots, \hat{b}_{k-1}$, respectively. Implementing the slicer function used in (2-23) is difficult because the set $\mathcal{L}_k$ can be large, and it depends on the channel. Since the elements of $\tilde{\boldsymbol{b}}$ are known to be integers, a common simplification is to assume that the transmission alphabet is the set of complex integers. This assumption causes only a small degradation in performance, and the slicer function becomes a simple round:

$$\hat{b}_k = \left\lceil y_k - \sum_{j=1}^{k-1} m_{k,j}\hat{b}_j \right\rfloor, \tag{2-24}$$

where $\lceil y \rfloor = \lceil \mathrm{Re}\{y\} \rfloor + \sqrt{-1} \lceil \mathrm{Im}\{y\} \rfloor$ independently rounds each part of y to the nearest integer.

The final step of the LA-DF detector is to convert the decision about $\tilde{\boldsymbol{b}} = \mathbf{T}^{-1}\boldsymbol{b}$ into a decision about $\boldsymbol{a}$. To do so, first $\hat{\boldsymbol{b}}$ is multiplied by $\mathbf{T}$, then the scaling and shifting transformation is reversed. Since $\hat{b}_k$ could be any complex integer, this conversion may yield symbol decisions that do not belong to the alphabet $\mathcal{A}$. To deal with this possibility we append a conventional symbol slicer, yielding:

$$\hat{\boldsymbol{a}} = dec\{(\mathbf{T}\hat{\boldsymbol{b}} + \boldsymbol{s})2c\}, \tag{2-25}$$

where $dec\{\boldsymbol{x}\}$ returns the element of $\mathcal{A}$ nearest each element of $\boldsymbol{x}$.

So far we have described how to implement the LA-DF detector given the lattice-reduction matrix $\mathbf{T}$. But calculating $\mathbf{T}$ is a difficult problem in itself. In [49] an optimal lattice-reduction technique that applies only to channels with two inputs was proposed. The most popular lattice-reduction technique is the LLL algorithm [31][34][44][46]. A pseudocode implementation of the LLL algorithm is given in Appendix C. Figure 2-4 illustrates the performance improvement attained using lattice reduction. The BER curves for the linear and DF detectors combined with LLL lattice reduction are within 3 dB of optimal performance.

In terms of complexity, the LA-DF detector is very similar to the DF detector. The only significant difference is that the preprocessing complexity is greater for the LA-DF detector since it involves calculating the lattice-reduction matrix $\mathbf{T}$. The LA-DF detector is implemented following the same process as the DF detector except that the symbol slicer is replaced by a simple rounding operation. The only other differences are that the LA-DF detector must perform an initial scale and shift of the channel output (2-18), and it maps its decision vector back to the QAM alphabet (2-25).

2.5. Sphere Detection: Reducing the Complexity of the ML Detector

The brute-force ML detector (1-5) is impractical due to its exponential complexity. The sphere detector is a better way to implement the ML detector whose average complexity can have polynomial complexity [26]. The sphere detector was first applied specifically to the real channel model of MIMO detection in [42] and [14], but it is founded on earlier works [35][29][36]. It was applied to the complex MIMO channel model in [27]. Instead of computing the costs of every possible decision vector, the sphere

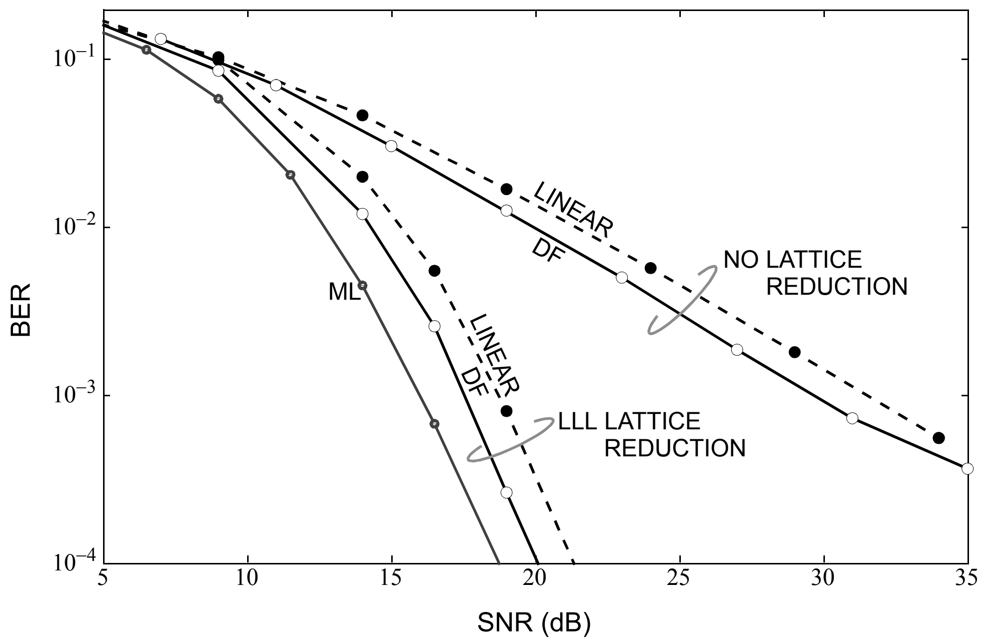

Figure 2-4. Performance of the linear and DF detectors with and without the help of LLL lattice reduction as averaged over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs.

detector only computes the costs of decision vectors that lie within a hypersphere centered on the channel output. The key to the complexity reduction achieved by the sphere detector is its management of the radius of this hypersphere.

The sphere detector begins in the same way as the DF detector, that is by computing the QR decomposition of the channel (2-1). Just like the DF detector, the sphere detector may be applied to an effective channel model where the columns of the channel matrix have been permuted (2-9) [1][13], or the lattice generated by the channel matrix has been reduced (2-20) [1]. However, if the sphere detector is applied to the MMSE channel model (2-15) [13], it only approximates the ML detector. For simplicity, we describe the sphere detector using the original channel model (1-1), with the understanding that this sphere detector can also be applied to the effective channel models (1-3), (2-9), (2-15), and (2-20).

The MIMO detection problem may be mapped onto a tree where each possible symbol vector, $\tilde{\boldsymbol{a}} \in \mathcal{A}^N$, defines a unique leaf node, and the tree has a level for each of the N symbols in the vector $\tilde{\boldsymbol{a}}$. The MIMO detector's job is to find the path from the root node of the tree to the "best" leaf node, which is the leaf node with minimum mean-squared error (MSE) $\| \boldsymbol{r} - \mathbf{H}\tilde{\boldsymbol{a}} \|^2$. Each branch leading from the root node towards a leaf node corresponds to choosing one symbol in the symbol decision vector $\tilde{\boldsymbol{a}}$.

The second step of the sphere detector is also the same as for the DF detector; it applies a front-end filter to the channel output to triangularize the channel (2-2):

$$\boldsymbol{y} = \mathbf{M}\boldsymbol{a} + \mathbf{D}^{-1}\mathbf{Q}^*\boldsymbol{w}. \tag{2-26}$$

After channel triangularization, the cost function of the leaf nodes, or candidate decision vectors, is written as:

$$\| \boldsymbol{r} - \mathbf{H}\tilde{\boldsymbol{a}} \|^2 = \mathbf{D}^2 \| \boldsymbol{y} - \mathbf{M}\tilde{\boldsymbol{a}} \|^2. \tag{2-27}$$

In order to implement classical tree-search algorithms, we must assign a cost to each branch in the tree. The triangular structure of $\mathbf{M}$ makes this possible. Specifically, let the symbols $\hat{a}_1, \ldots, \hat{a}_{k-1}$ define a path through the tree to a node at the k-th level. Using this notation, $\hat{a}_k$ specifies a branch connecting nodes on the k-th and $(k+1)$-th levels of the tree, whose cost can be expressed as:

$$\lambda(\hat{a}_k) = \sum_{j=1 \ldots k} | y_j - \sum_{m \le j} m_{j,m} \hat{a}_m |^2. \tag{2-28}$$

Although a breadth-first search is possible [18], the depth-first tree search is preferable due to its simplicity. Beginning from the root node it prunes (discards) branches and all leaf nodes descending from branches when their cost exceeds the current radius of the hypersphere R. Therefore, branches from the k-th level of the tree are pruned if $\lambda(\hat{a}_k) > R$. Each time a leaf node is reached whose cost is less than the current radius of the hypersphere, then R is set to this lower cost.

The choice of the initial radius of the hypersphere is critical to both the complexity and performance of the sphere detector. If the initial radius is too small then the hypersphere will not include the ML solution. On the other hand, the number of points that must be searched inside the hypersphere increases as its radius increases. In reality, the number of points inside the hypersphere is a random variable depending upon the channel and the additive noise, and the sphere detector cannot guarantee that it will not search all possible symbol vectors. Various methods of choosing the initial radius have been proposed in [26][53]. Another approach is to choose a small initial radius, and increase it if the hypersphere is empty. If the costs of the branches calculated for partial paths through the tree are stored, then no computations would need to be repeated after increasing the radius. However, storing the costs of partial paths of unsuccessful searches requires an exponential amount of memory. In [51], a compromise between reducing complexity by storing all information and reducing memory by storing the information corresponding to only the most promising paths was proposed.

If the initial radius is set to infinity, then the sphere detector is guaranteed to find the ML decision vector. The expected complexity of the sphere detector search beginning with an infinite initial radius depends on the distance of the first candidate vector found from that of the ML decision vector. For this reason, if one could improve the quality of the first candidate vector found, then the expected complexity of the sphere detector would be reduced since the hypersphere would contain fewer candidates. Therefore, to reduce complexity it is a good idea to explore the most likely branches first [9], in which case the first candidate decision vector found by the sphere detector is the same as the zero-forcing DF detector decision. From this viewpoint, it makes sense that in the same way the BODF detector outperforms the DF detector, the sphere detector has lower average complexity if its levels are sorted according to the BLAST ordering [13][53].

Recently it has been shown [38] that the channel permutation that minimizes the complexity of the sphere detector depends not only on the channel matrix, but also on the additive noise of the channel. Furthermore, given this optimal permutation matrix, the complexity of the sphere detector's tree search is dramatically reduced. Unfortunately, calculating this permutation matrix is relatively complex, and it must be recomputed for every channel output. Finding a low-complexity way to compute this permutation matrix is an important open problem.

Finally, the sphere detector is usually implemented using the real channel model (1-3) because that allows for the use of the Schnorr enumeration [36][13] which simplifies the implementation. However, recent research suggests that using the complex channel model will reduce complexity [7].

2.6. Approximating the ML detector

Besides the sphere detector, the tree-search view of the detection problem has also spawned many approximations of the ML detector that reduce complexity [5][28][19][12]. In [5], the detection problem is broken down into pieces to be solved by sphere detectors. The decision regarding the first symbol comes from the best decision vector at the T-th level of the tree. This interference due to the first symbol is cancelled out leaving a tree with only $N-1$ levels. The decision regarding the second symbol comes from the best decision vector at the T-th level of this new smaller tree. The process continues until all decisions are made. In [28], a breadth-first tree search is used, but the detector maintains only the T best paths. In [19], the ZF linear detector makes decisions about the first T symbols, and the ML detector is used to make the remaining decisions. Finally, [12] proposes a way to reduce the complexity when larger QAM alphabets are used. It first implements the sphere detector assuming that the 4-QAM alphabet was used. It then eliminates possible symbol decisions that are outside the quadrant of the symbols found using the 4-QAM alphabet, and searches for the best decision vector from the remaining candidates.

Since the sphere detector complexity is reduced when the first candidate decision vector found is nearer the ML decision vector, it has been proposed to apply the sphere detector to the BLAST-ordered MMSE channel model (2-17) because the first decision vector found in this case is that of the MMSE BODF detector [13]. This will not produce the ML decision vector because of the residual ISI term that is characteristic of MMSE detection. However, the performance loss relative to the ML detector is small [13].

2.6.1. Truncated Sphere-Detector

The sphere detector has low *average* complexity, but high *worst-case* complexity. A practical system must be prepared to implement the detector for all possible channels, so measuring the worst-case complexity of the sphere detector is more meaningful than measuring its expected complexity. Unfortunately this means that the sphere detector is prohibitively complex in many cases.

An intuitive way to limit the complexity of the depth-first sphere detector is to simply abort the tree-search once a complexity limit has been exceeded, a technique we refer to as *truncated-sphere detection*. Since the depth-first sphere detector continually updates a candidate decision vector, once the complexity limit has been reached it could simply return the best candidate it found – this is truncated-sphere detection. Since the depth-first tree search goes directly to a leaf node corresponding to the decision vector of the DF detector, even if the tree search is aborted it returns a reasonably reliable decision vector. By implementing the truncated-sphere detector with a range of complexity thresholds we can measure the performance-complexity trade-off of the sphere detector quantitatively.

The less the tree search is aborted, the closer the truncated-sphere detector performance will be to ML performance. In order to determine an appropriate complexity limit for the truncated-sphere detector, we built a histogram of the complexity of the sphere detector over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs. The most multiplications used by the sphere detector during any one symbol period was 4514. The number of multiplications used by the sphere detector exceeded 1642, 656, and 230, with probabilities 10^{-4}, 10^{-3}, and 10^{-2}, respectively. Figure 2-5 shows the BER curves of the truncated-sphere detector using these complexity limits as well as the sphere detector. These results demonstrate that even setting the complexity limit high enough that the truncated-sphere detector differs from the sphere detector only one in a thousand times, causes the amount of SNR required to reach BER = 10^{-3} to increase by about half a dB. Increasing the complexity limit further causes even bigger performance penalties.

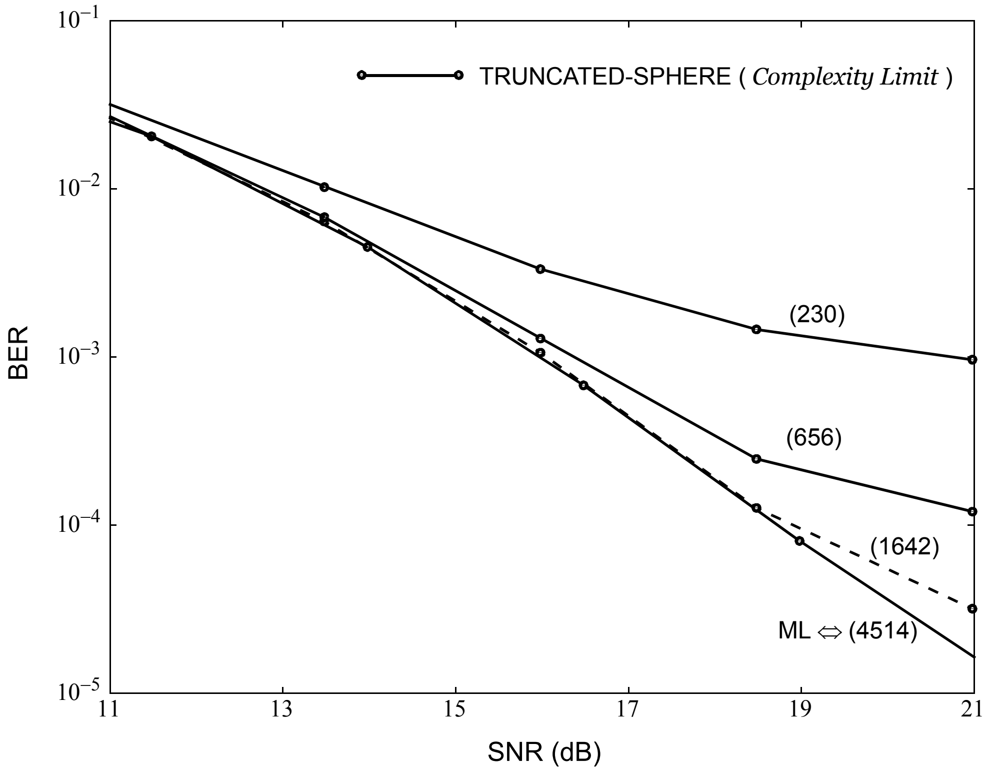

Figure 2-5. Performance of the truncated-sphere detector with various complexity limits as averaged over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs.

2.6.2. The ML-DF Detector

In order to achieve a compromise between the performance of the ML detector and the low complexity of the BDF detector, it has been proposed to combine the two detectors [11][30]. The ML-DF detector [11] uses the sphere detector to find the best path to the i-th level of the tree, then uses decision-feedback detection to make decisions on the remaining $N - i$ symbols. The performance of the ML-DF detector decays as SNR^{-i}, in other words it has a diversity order of i, and the complexity is exponential in i. When $i = 1$, the ML-DF detector reduces to the DF detector, and when $i = N$ it reduces to the sphere detector. Therefore, adjusting the parameter i allows the receiver to trade performance for complexity.

CHAPTER 3
Reducing Complexity of the Optimally-Ordered DF Detector

The performance of the decision-feedback (DF) detector is strongly impacted by the order in which the inputs are detected. Unfortunately, optimizing the detection order is a difficult problem that often dominates the overall receiver complexity. It is common and practical to define as optimal the detection order that maximizes the worst-case post-detection SNR. This ordering, known as the BLAST ordering, approximately minimizes the joint error probability of the DF detector. The DF detector that uses this BLAST ordering is known either as the optimally-ordered or BLAST-ordered decision-feedback (BODF) detector. The BLAST-ordering algorithm of [23] uses repeated computations of a matrix pseudoinverse to find this ordering with a complexity of $\mathcal{O}(N^4)$, where N is the number of channel inputs. Other algorithms that compute the BLAST ordering with complexity $\mathcal{O}(N^4)$ have also been proposed: the post-sorting algorithm of [48], the decorrelating algorithm of [50], the square-root algorithms of [25] and [52], and the recursive algorithm of [3].

In [16] an architecture for implementing the DF detector based on linear prediction of the noise was presented. The noise-predictive DF detector consists of a linear detector followed by a linear prediction mechanism that reduces the noise variance before making a decision. In this paper we propose a low-complexity technique for determining the BLAST symbol ordering that is facilitated by the noise-predictive DF detector. The resulting noise-predictive BLAST-ordered DF (NP-BODF) detector is mathematically equivalent to the BLAST-ordered DF (BODF) detector [23]. However, the NP-BODF detector is less complex than the lowest-complexity BODF detector previously reported [50]. In fact, if the linear detection filter is already known, the NP-BODF detector requires roughly half the preprocessing complexity required by other BODF detectors. A key advantage of the noise-predictive approach is that it allows *existing* systems that use linear detection to be transformed (upgraded) into BODF detectors with the addition of relatively simple processing.

In this chapter, we also derive the minimum-mean-squared-error (MMSE) version of the noise-predictive DF detector for MIMO channels. We show that our novel ordering algorithm is easily modified to find the MMSE BLAST ordering.

The outline of this chapter is as follows. Section 3.1 describes the zero-forcing noise-predictive DF detector of [16]. Section 3.2 describes a low-complexity implementation of the NP-BODF detector. Section 3.3 derives the MMSE noise-predictive DF detector and describes how to find the corresponding BLAST ordering. Section 3.4 describes practical implementation issues for the BODF detector. Finally, Section 3.5 compares the complexity of the NP-BODF detector with previously proposed implementations of the BODF detector.

3.1. ZF Noise-Predictive DF Detection

We now derive an alternative implementation of the ZF-DF detector based upon linear prediction of the noise, as first proposed in [16]. Figure 3-1 shows the block diagram of the *zero-forcing noise-predictive* DF (ZF-NP-DF) detector which employs this linear-prediction strategy; the filters $\boldsymbol{c}_i$ and $p_{i,j}$ will be defined shortly. The notion of ordering (the permutation block) is neglected momentarily by assuming an identity permutation.

The starting point for the ZF-NP-DF detector is the ZF *linear* detector [41], which essentially inverts the channel by computing $\boldsymbol{y} = \mathbf{C}\boldsymbol{r}$, where $\mathbf{C}$ is the channel pseudoinverse:

$$\mathbf{C} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*. \tag{3-1}$$

In Figure 3-1, $\boldsymbol{c}_i$ denotes the i-th *row* of $\mathbf{C}$. From (2-1), the output of this filter is free of interference:

$$\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{n}, \tag{3-2}$$

where the noise $\boldsymbol{n} = [n_1, \dots n_N]^T = \mathbf{C}\boldsymbol{w}$ is no longer white; its autocorrelation matrix is $\mathbf{R}_{\boldsymbol{nn}} = \mathrm{E}[\boldsymbol{nn}^*] = \sigma^2(\mathbf{H}^*\mathbf{H})^{-1}$.

The correlation of the noise can be exploited using linear prediction to reduce its variance. If the first $i-1$ elements of the noise vector were known, we could form an estimate $\hat{n}_i$ of the i-th element n_i and subtract this estimate from y_i to reduce its variance. Specifically, given $\{n_1, \dots n_{i-1}\}$, a linear predictor estimates n_i according to:

$$\hat{n}_i = \sum_{j<i} p_{i,j} n_j, \tag{3-3}$$

or equivalently $\hat{\boldsymbol{n}} = \mathbf{P}\boldsymbol{n}$, where $\mathbf{P}$ is a strictly lower triangular *prediction filter* whose element at the i-th row and j-th column is $p_{i,j}$. This process is complicated by the fact that the receiver does not have access to n_i directly, but rather to the sum $y_i = a_i + n_i$. However, as shown in Figure 3-1, the decision about a_i can be subtracted from y_i to yield n_i as long as the decision is correct.

Let us define the *total* MSE as $\mathrm{E}[\|\hat{\boldsymbol{n}} - \boldsymbol{n}\|^2] = \sum_i \mathrm{E}[|\hat{n}_i - n_i|^2]$, which measures the quality of the prediction. As shown in [8], this total MSE is minimized by the following prediction filter:

$$\mathbf{P} = \mathbf{I} - \mathbf{M}\,, \tag{3-4}$$

where $\mathbf{M}$ is defined by the QR decomposition of (2-1). Having thus defined the prediction coefficients, the ZF-NP-DF detector of Figure 3-1 can be summarized succinctly by the following recursion:

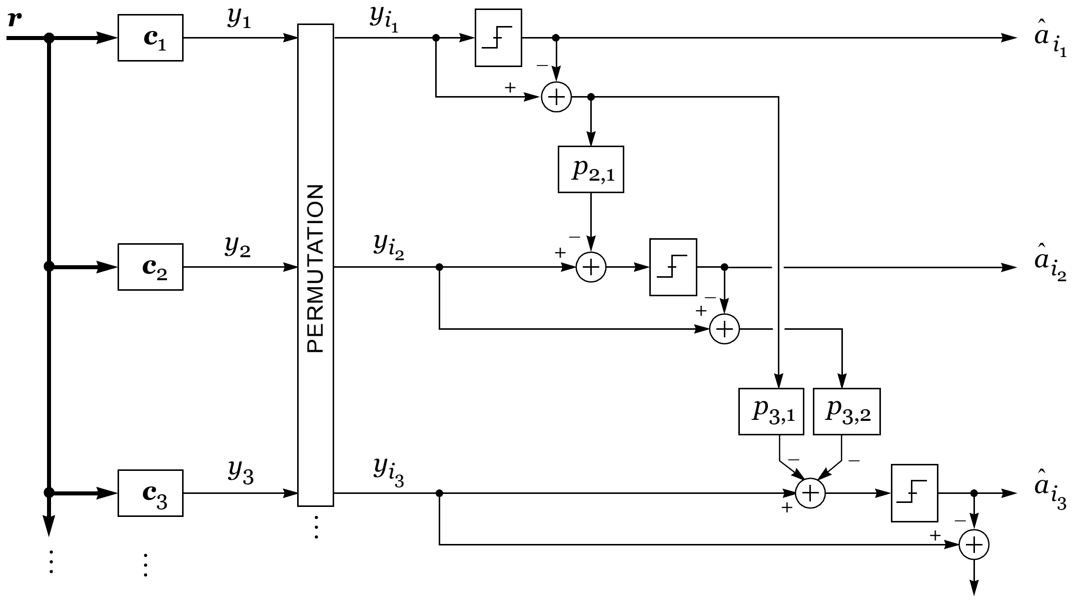

Figure 3-1. The noise-predictive DF detector.

$$\hat{a}_i = \text{dec}\left\{ y_i - \sum_{j<i} p_{i,j}(y_j - \hat{a}_j) \right\}. \tag{3-5}$$

We now show that the ZF-DF detector (2-4) and the ZF-NP-DF detector (3-5) are equivalent. Substituting (3-2) and (3-4) into (3-5) yields the following for the noise-predictive implementation:

$$\hat{a}_i = \text{dec}\left\{ a_i + \sum_{j \le i} m_{i,j}\mathbf{c}_j\boldsymbol{w} - \sum_{j<i} m_{i,j}(\hat{a}_j - a_j) \right\}, \tag{3-6}$$

where we exploited the fact that $m_{i,i} = 1$ and $p_{i,j} = -m_{i,j}$ when $j < i$, and where we substituted $n_j = \mathbf{c}_j\boldsymbol{w}$. On the other hand, for the conventional implementation, substituting (2-3) into (2-4) gives:

$$\hat{a}_i = \text{dec}\left\{ a_i + \boldsymbol{q}_i^*\boldsymbol{w}/d_{i,i} - \sum_{j<i} m_{i,j}(\hat{a}_j - a_j) \right\}. \tag{3-7}$$

The conventional and noise-predictive detectors are equivalent when equations (3-6) and (3-7) are identical, or when:

$$\sum_{j \le i} m_{i,j}\mathbf{c}_j = \boldsymbol{q}_i^*/d_{i,i}. \tag{3-8}$$

In matrix form, (3-8) simplifies to:

$$\mathbf{MC} = \mathbf{D}^{-1}\mathbf{Q}^*. \tag{3-9}$$

But since:

$$\begin{aligned} \mathbf{MC} &= \mathbf{M}(\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^* \\ &= \mathbf{M}(\mathbf{M}^{-1}\mathbf{D}^{-2}\mathbf{M}^{-*})\mathbf{H}^* \\ &= \mathbf{D}^{-1}\mathbf{Q}^*, \end{aligned} \tag{3-10}$$

we conclude that the conventional ZF-DF detector and the ZF-NP-DF detector are indeed equivalent.

3.2. Optimally-Ordered Noise-Predictive DF Detection

To implement the ordered ZF-NP-DF detector of Figure 3-1, the receiver must first calculate the channel pseudoinverse **C**, the symbol detection order, and the linear prediction filter **P**. In this section we show how to calculate both the optimal detection order and the prediction filter given knowledge of the channel pseudoinverse.

We first describe a low-complexity algorithm for finding the best (BLAST) detection order. As implied by Figure 3-1, this sorting algorithm occurs after $\boldsymbol{y} = \mathbf{C}\boldsymbol{r}$ has been calculated. The permutation in the block diagram of Figure 3-1 gives the detector the flexibility to use any symbol detection order, but in this paper we assume that the BLAST ordering is used. Let $\{i_1, i_2, \dots i_N\}$ denote the BLAST ordering, a permutation of the integers $\{1, 2, \dots N\}$ such that i_k denotes the index of the k-th symbol to be detected.

The noise-predictive view of the DF detector leads to a simple algorithm for finding the BLAST ordering. As proven in [22], the BLAST ordering can be found in a recursive fashion by choosing each i_k so as to maximize the post-detection SNR of the k-th symbol, or equivalently minimize its MSE. Specifically, because the MSE for the first detected symbol is $\sigma^2 \| \mathbf{c}_{i_1} \|^2$, we have:

$$i_1 = \underset{j \in \{1, \dots N\}}{\operatorname{argmin}} \| \mathbf{c}_j \|^2. \tag{3-11}$$

In other words, the channel pseudoinverse row with the smallest norm determines which symbol to detect first. Once i_1 is chosen, and assuming $\hat{a}_{i_1}$ is correct, the MSE for the second symbol is:

$$\begin{aligned} E[|n_{i_2} - \hat{n}_{i_2}|^2] &= E[|\mathbf{c}_{i_2}\boldsymbol{w} - p_{2,1}\mathbf{c}_{i_1}\boldsymbol{w}|^2] \\ &= \sigma^2 \| \mathbf{c}_{i_2} - p_{2,1}\mathbf{c}_{i_1} \|^2. \end{aligned} \tag{3-12}$$

When the prediction coefficient $p_{2,1}$ is chosen to minimize the above MSE, the term $p_{2,1}\mathbf{c}_{i_1}$ reduces to the projection of $\mathbf{c}_{i_2}$ onto the subspace spanned by $\mathbf{c}_{i_1}$, which we denote as $\hat{\mathbf{c}}_{i_2}$. Hence, the optimal i_2 satisfies:

$$i_2 = \underset{j \neq i_1}{\arg\min} \ \| \mathbf{c}_j - \hat{\mathbf{c}}_j \|^2. \tag{3-13}$$

Repeating the above procedure recursively leads to the following simple and succinct procedure for finding the BLAST ordering:

$$i_k = \underset{j \notin \{i_1, \dots i_{k-1}\}}{\arg\min} \ \| \mathbf{c}_j - \hat{\mathbf{c}}_j \|^2, \tag{3-14}$$

where $\hat{\mathbf{c}}_j$ denotes the projection of $\mathbf{c}_j$ onto the span of $\{\mathbf{c}_{i_1}, \dots \mathbf{c}_{i_{j-1}}\}$. This is a key result that is the basis of the noise-predictive implementation of the BLAST ordered DF detector. In words, *finding the BLAST ordering amounts to choosing the rows of the channel pseudoinverse, where the best choice for the k-th row is the unchosen row that is closest to the subspace spanned by the rows already chosen*.

A computationally efficient implementation of the sorting algorithm of (3-14) is given in Figure 3-2. It is based on an adaptation of the modified Gramm-Schmidt (MGS) QR decomposition [24]. The algorithm accepts the channel pseudoinverse $\mathbf{C}$ as an input, and it produces the optimal ordering $\{i_1, \dots i_N\}$. The MGS procedure of the sorting algorithm operates on the rows of $\mathbf{C}$, $\{\mathbf{c}_{i_1}, \dots \mathbf{c}_{i_N}\}$. During the first iteration ($j = 1$), Line 4 chooses the row nearest to the null space. Then, Line 10 removes the portions from the remaining rows of $\mathbf{C}$ that are parallel to $\mathbf{c}_{i_1}$. Therefore, in the next iteration ($j = 2$) each of the candidate rows of $\mathbf{C}$ is orthogonal to $\mathbf{c}_{i_1}$. Consequently, the remaining row closest to the subspace spanned by the previously chosen row is simply the row with minimum norm. As before, Line 10 ensures that the remaining rows of **C** are orthogonal to $\mathbf{c}_{i_2}$. The iterations continue the BLAST ordering is determined.

Close inspection of the algorithm in Figure 3-2 indicates that it is functionally equivalent to an upper triangular sorted-QR decomposition [48]. Figure 3-3 gives the pseudocode for the sorted-QR decomposition that implements the exact same sort. The sorted-QR decomposition computed in Figure 3-3 can be described as:

$$\mathbf{C}^*\Pi = \mathbf{QU}, \tag{3-15}$$

where **U** is assigned from the **R** output of the sorted-QR decomposition. Since the QR decomposition is unique, comparing (3-15) to (2-10) indicates that **U** is also equal to the

```
Function    NPsort.
                Input: $\mathbf{C}$,    Output: $\{i_1, i_2, \ldots i_N\}$

1.  $\mathcal{U} = \{1, 2, \ldots N\}$ = the set of unchosen rows.
2.  for $j$ = 1 to $N$, $e_j = \sum_{k=1\ldots j} |c_{j,k}|^2$ , end
3.  for $j$ = 1 to $N$,
4.      $i_j = \operatorname{argmin}_{k \in \mathcal{U}} e_k$
5.      $\mathcal{U} = \mathcal{U} - i$ ; remove chosen row from $\mathcal{U}$.
6.      $f_{i_j,j} = \sqrt{e_{i_j}}$
7.      $\mathbf{c}_{i_j} = \mathbf{c}_{i_j} / f_{i_j,j}$
8.      for $k \in \mathcal{U}$
9.          $f_{k,j} = \mathbf{c}_k \mathbf{c}_{i_j}^*$
10.         $\mathbf{c}_k = \mathbf{c}_k - f_{k,j}\mathbf{c}_{i_j}$
11.         $e_k = e_k - |f_{k,j}|^2$
12.     end
13. end
```

Figure 3-2. The noise-predictive sorting algorithm.

inverse of $\mathbf{L}^* = \mathbf{M}^*\mathbf{D}^*$, where $\mathbf{D}$ and $\mathbf{M}$ are defined from the QR decomposition of $\mathbf{H}$ (2-10).

The fact that the BLAST sort can be implemented as a sorted-QR decomposition leads to a conceptually simple way to compute the prediction matrix $\mathbf{P} = \mathbf{I} - \mathbf{M}$ (3-4). The matrix $\mathbf{D}^{-1}$ is actually a by-product of the sorted-QR decomposition of Figure 3-3 because the diagonal elements of $\mathbf{D}^{-1}$ are just the diagonal elements of $\mathbf{U}$. This means that $\mathbf{M} = \mathbf{D}^{-1}(\mathbf{U}^{-1})^*$ and the prediction matrix is computed as:

$$\mathbf{P} = \mathbf{I} - \mathbf{D}^{-1}(\mathbf{U}^{-1})^*. \tag{3-16}$$

Implementing the BLAST-sorting algorithm using the sorted-QR decomposition as in Figure 3-3 makes it easy to see that the same sorting algorithm can be realized using any

```
Function    SortedQR_UPPER
              Input: $\mathbf{Q} = \mathbf{C}^*$, Output: $\mathbf{Q}$, $\mathbf{R}$, $\Pi$, and $\mathbf{D}$

1.   $\mathbf{Q} = \mathbf{C}^*$,  $\Pi = \mathbf{I}$,  $\mathbf{R} = \mathbf{0}_{N\text{x}N}$
2.   for $j = 1$ to $N$, $e_j = \sum_{k=1\ldots j} |q_{k,j}|^2$ , end
3.   for $j = 1$ to $N$,
4.       $i = \arg\min_{k=j \text{ to } N} e_k$
5.       Swap $i$-th and $j$-th columns of $\mathbf{Q}$, $\mathbf{R}$, and $\Pi$
         Swap $i$-th and $j$-th elements of $\boldsymbol{e}$
6.       $r_{j,j} = \sqrt{e_j}$
7.       $\boldsymbol{q}_j = \boldsymbol{q}_j / r_{j,j}$
8.       for $k = j+1$ to $N$,
9.           $r_{j,k} = \boldsymbol{q}_j^* \boldsymbol{q}_k$
10.          $\boldsymbol{q}_k = \boldsymbol{q}_k - r_{j,k}\boldsymbol{q}_j$
11.          $e_k = e_k - |r_{j,k}|^2$
12.      end
13.  end
```

Figure 3-3. The noise-predictive sorting algorithm using the upper triangular sorted-QR decomposition.

implementation of the QR decomposition. The MGS implementation we have shown here is only one possibility. Two other well-known implementations of the QR decomposition include using Householder or Givens rotations [24].

In summary, given the channel pseudoinverse, the zero-forcing NP-BODF detector implementation has four steps. First, the channel pseudoinverse is applied to the received vector. Next, the BLAST symbol order is calculated using the sorted-QR decomposition of the conjugate transpose of the channel pseudoinverse. Then, the linear prediction filter is calculated from the output of the sorting algorithm. After these calculations the detector can be implemented using (3-5), as illustrated in Figure 3-1.

3.3. Noise-Predictive MMSE DF Detection

The ZF-DF detector cancels the interference completely without regard to noise amplification. The MMSE-DF detector improves on this strategy by finding the optimal balance between interference cancellation and noise reduction that minimizes the total MSE [2]. In this section we derive a noise-predictive implementation of the BLAST-ordered MMSE-DF detector.

Like the ZF-DF detector, the MMSE-DF detector can also be implemented as a cascade of a linear filter and a noise-predictive mechanism, so that the basic architecture of Figure 3-1 applies to both the zero-forcing and MMSE versions of the DF detector. But instead of the channel pseudoinverse, the noise-predictive MMSE DF detector begins with the MMSE linear detection filter [41]:

$$\mathbf{C} = \tilde{\mathbf{R}}^{-1}\mathbf{H}^*, \tag{3-17}$$

where:

$$\tilde{\mathbf{R}} = \mathbf{H}^*\mathbf{H} + \sigma^2\mathbf{I}\,. \tag{3-18}$$

This choice for **C** minimizes the total MSE $E[||\varepsilon||^2]$, where $\varepsilon = \mathbf{C}\boldsymbol{r} - \boldsymbol{a}$ is the vector of errors after the linear filter. Unlike the ZF case, this error vector contains residual intersymbol interference (ISI) as well as noise:

$$\begin{aligned}\varepsilon &= \tilde{\mathbf{R}}^{-1}(\mathbf{H}^*\mathbf{H} + \sigma^2\mathbf{I} - \sigma^2\mathbf{I})\boldsymbol{a} + \mathbf{C}\boldsymbol{w} - \boldsymbol{a} \\ &= -\sigma^2\tilde{\mathbf{R}}^{-1}\boldsymbol{a} + \mathbf{C}\boldsymbol{w}\,.\end{aligned} \tag{3-19}$$

In the following we continue to use our "noise"-predictive terminology, even though strictly speaking the "noise" being predicted is ε, which contains residual ISI as well as noise.

Let $\hat{\varepsilon} = \mathbf{P}\varepsilon$ denote an estimate of ε based on linear prediction. Let $\boldsymbol{e} = (\mathbf{I} - \mathbf{P})\varepsilon$ denote the error in this estimate. We now derive the strictly lower-triangular linear prediction filter **P** that minimizes the total MSE $E[||\boldsymbol{e}||^2]$. From (3-19), the autocorrelation matrix $\mathbf{R}_{\varepsilon\varepsilon} = E[\varepsilon\varepsilon^*]$ of ε can be written as:

$$\begin{aligned}\mathbf{R}_{\varepsilon\varepsilon} &= \sigma^4\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{R}}^{-1} + \sigma^2\mathbf{C}\mathbf{C}^* \\ &= \sigma^4\tilde{\mathbf{R}}^{-2} + \sigma^2\tilde{\mathbf{R}}^{-1}(\mathbf{H}^*\mathbf{H} + \sigma^2\mathbf{I} - \sigma^2\mathbf{I})\tilde{\mathbf{R}}^{-1} \\ &= \sigma^2\tilde{\mathbf{R}}^{-1}.\end{aligned} \tag{3-20}$$

Since $\tilde{\mathbf{R}}$ is Hermitian and positive definite, $\mathbf{R}_{\varepsilon\varepsilon}$ has the following Cholesky factorization:

$$\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2\tilde{\mathbf{M}}^{-1}\tilde{\mathbf{D}}^{-2}\tilde{\mathbf{M}}^{-*}, \tag{3-21}$$

where $\tilde{\mathbf{M}}^{-1}$ is a lower triangular matrix with diagonal elements of one, and where $\tilde{\mathbf{D}}^{-2}$ is a real diagonal matrix with positive diagonal elements. The total MSE after linear prediction is related to $\mathbf{R}_{\varepsilon\varepsilon}$ by:

$$E[\,\|\boldsymbol{e}\|^2] = \text{trace}\{(\mathbf{I}-\mathbf{P})\mathbf{R}_{\varepsilon\varepsilon}(\mathbf{I}-\mathbf{P}^*)\}. \tag{3-22}$$

It is easy to show [8] that the best choice for $(\mathbf{I}-\mathbf{P})$ cancels $\tilde{\mathbf{M}}^{-1}$:

$$\mathbf{I}-\mathbf{P} = \tilde{\mathbf{M}}. \tag{3-23}$$

Therefore, the effective front-end filter of the MMSE noise-predictive DF detector is given by:

$$\begin{aligned}(\mathbf{I}-\mathbf{P})\mathbf{C} &= \tilde{\mathbf{M}}\tilde{\mathbf{R}}^{-1}\mathbf{H}^* \\ &= \tilde{\mathbf{D}}^{-2}\tilde{\mathbf{M}}^{-*}\mathbf{H}^*.\end{aligned} \tag{3-24}$$

This forward filter is identical to the forward filter of the conventional MMSE-DF detector defined in [15]. With this forward filter, the corresponding feedback filter is $-\mathbf{P}$, which is identical to the feedback filter of the conventional MMSE-DF detector defined in [15]. Therefore, we conclude that the noise-predictive MMSE-DF detector is equivalent to the conventional MMSE-DF detector.

Just as for the ZF-DF detector, the performance of the MMSE-DF detector is improved if the detection order of the symbols is chosen to minimize the maximum MSE. However, the ordering problem for the noise-predictive MMSE-DF detector is complicated by the fact that the "noise" includes residual ISI. For convenience, we define an augmented matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} \mathbf{C} & \sigma\tilde{\mathbf{R}}^{-1} \end{bmatrix}, \tag{3-25}$$

so that $\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2\mathbf{B}\mathbf{B}^*$. Let $\boldsymbol{b}_j$ be the j-th row of the matrix $\mathbf{B}$. From (3-20), the MSE for the first detected symbol is equal to $[\mathbf{R}_{\varepsilon\varepsilon}]_{i_1,i_1} = \sigma^2\|\,\boldsymbol{b}_{i_1}\|^2$. Therefore, we choose the symbol with minimum MSE by:

$$i_1 = \underset{j \in \{1, \dots N\}}{\arg\min} \, \| \boldsymbol{b}_j \|^2. \tag{3-26}$$

After i_1 is chosen, and assuming $\hat{a}_{i_1}$ is correct, the MSE for the second symbol is:

$$E[|\varepsilon_{i_2} - \hat{\varepsilon}_{i_2}|^2] = E[|\varepsilon_{i_2} - p_{2,1}\varepsilon_{i_1}|^2] \tag{3-27}$$

Let $\tilde{\boldsymbol{r}}_j$ be the j-th row of the matrix $\tilde{\mathbf{R}}^{-1}$. Then by substituting from (3-19), the MSE for the second symbol becomes:

$$\begin{aligned} E[|\varepsilon_{i_2} - \hat{\varepsilon}_{i_2}|^2] &= E[|\boldsymbol{c}_{i_2}\boldsymbol{w} - \sigma^2\tilde{\boldsymbol{r}}_{i_2}\boldsymbol{a} - p_{2,1}\boldsymbol{c}_{i_1}\boldsymbol{w} + \sigma^2 p_{2,1}\tilde{\boldsymbol{r}}_{i_1}\boldsymbol{a}|^2] \\ &= \sigma^2\|\boldsymbol{c}_{i_2} - p_{2,1}\boldsymbol{c}_{i_1}\|^2 + \sigma^4\|\tilde{\boldsymbol{r}}_{i_2} - p_{2,1}\tilde{\boldsymbol{r}}_{i_1}\|^2 \\ &= \sigma^2\|\boldsymbol{b}_{i_2} - p_{2,1}\boldsymbol{b}_{i_1}\|^2, \end{aligned} \tag{3-28}$$

where the last equality in (3-28) follows from straightforward algebraic manipulation. When the prediction coefficient $p_{2,1}$ is chosen to minimize the MSE, the term $p_{2,1}\boldsymbol{b}_{i_1}$ reduces to the projection of $\boldsymbol{b}_{i_2}$ onto the subspace spanned by $\boldsymbol{b}_{i_1}$, which we denote as $\boldsymbol{b}_{i_2}$. Hence, the optimal i_2 satisfies:

$$i_2 = \underset{j \neq i_1}{\arg\min} \, \| \boldsymbol{b}_j - \hat{\boldsymbol{b}}_j \|^2. \tag{3-29}$$

The above procedure can be repeated recursively to determine the BLAST ordering. In a fashion reminiscent of the ZF sorting algorithm (3-14), the above procedure is succinctly described by the following recursive sorting algorithm:

$$i_k = \underset{j \notin \{i_1, \dots i_{k-1}\}}{\arg\min} \, \| \boldsymbol{b}_j - \hat{\boldsymbol{b}}_j \|^2, \tag{3-30}$$

where $\hat{\boldsymbol{b}}_j$ denotes the projection of $\boldsymbol{b}_j$ onto the span of $\{ \boldsymbol{b}_{i_1}, \dots \boldsymbol{b}_{i_{j-1}} \}$.

The MMSE sorting algorithm (3-30) just described is identical to the ZF sorting algorithm (3-14), except that $\mathbf{c}_j$ and $\hat{\mathbf{c}}_j$ have been replaced by $\boldsymbol{b}_j$ and $\hat{\boldsymbol{b}}_j$, respectively. As a result, we need not derive an implementation of the MMSE ordering algorithm from scratch; it is realized by the NPsort function of Figure 3-2 when the augmented matrix $\mathbf{B}$ is input instead of the channel pseudoinverse.

When $\sigma = 0$, $\mathbf{C}$ reduces to the channel pseudoinverse, and the $\sigma\tilde{\mathbf{R}}^{-1}$ matrix has no impact on the sorting algorithm. Therefore, as expected, the MMSE sorting algorithm reduces to the ZF sorting algorithm when the noise is zero.

3.4. Noise-Predictive BODF Detection Given the Channel Matrix

If the receiver is given the channel matrix $\mathbf{H}$ beforehand instead of the linear filter $\mathbf{C}$, the implementation of the NP-BODF detector can be simplified. The previous two sections have shown in detail that given the factorization of the autocorrellation of the noise $\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2\mathbf{BB}^*$, computing the upper-triangular sorted-QR decomposition of $\mathbf{B}^*$ yields the optimal ordering for the MMSE BODF detector. The simplification of the BLAST sort we discuss in this section is based on the fact that the factorization of $\mathbf{R}_{\varepsilon\varepsilon}$ is *not* unique. We describe the simplification only in terms of the MMSE BLAST sort because it includes the ZF BLAST sort as a special case.

The definition of $\mathbf{B}$ given in (3-25) is not the only factorization of $\mathbf{R}_{\varepsilon\varepsilon}$. Recall from (3-18) and (3-20), that $\mathbf{R}_{\varepsilon\varepsilon}$ is defined as:

$$\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2(\mathbf{H}^*\mathbf{H} + \sigma^2\mathbf{I})^{-1}. \tag{3-31}$$

This autocorrellation matrix can be written as a function of $\overline{\mathbf{H}}$ as follows:

$$\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2(\overline{\mathbf{H}}^*\overline{\mathbf{H}})^{-1}, \tag{3-32}$$

where $\overline{\mathbf{H}}$ is the extended channel matrix used to create the MMSE channel model (2-14):

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma\mathbf{I}_{NxN} \end{bmatrix}. \tag{3-33}$$

The QR decomposition of $\overline{\mathbf{H}}$ gives another valid factorization of $\mathbf{R}_{\varepsilon\varepsilon}$. The lower-triangular QR decomposition of $\overline{\mathbf{H}}$ is written as:

$$\overline{\mathbf{H}} = \mathbf{QDM}, \tag{3-34}$$

where $\mathbf{Q}$ is an $(M+N)\times N$ matrix with orthonormal columns, where $\mathbf{D}$ is an $N\times N$ diagonal matrix with diagonal elements that are positive and real, and where $\mathbf{M}$ is a lower triangular matrix with ones along the diagonal. $\mathbf{R}_{\varepsilon\varepsilon}$ can now be decomposed as follows:

$$\mathbf{R}_{\varepsilon\varepsilon} = \sigma^2(\mathbf{M}^*\mathbf{D}^*\mathbf{DM})^{-1}. \tag{3-35}$$

Given this new decomposition, $\mathbf{U} = (\mathbf{M}^*\mathbf{D}^*)^{-1}$ can replace $\mathbf{B}^*$ as the input to the NPsort function that calculates the MMSE BLAST ordering. This simplifies the implementation of the BLAST sort because $\mathbf{U}$ is easier to calculate than $\mathbf{B}$ since $\mathbf{U}$ has smaller dimensions than $\mathbf{B}$, and $\mathbf{U}$ is triangular. Figure 3-4 gives the pseudocode to implement the MMSE BLAST sort starting from $\overline{\mathbf{H}}$.

3.5. Comparing Different DF Implementations

Figure 3-4 gives the pseudocode for three different ways to implement the MMSE BLAST sort starting from $\overline{\mathbf{H}}$. After the BLAST sorting is done, the BODF detector can be implemented using either the noise-predictive approach (see Figure 3-1), the conventional approach (see Figure 2-1), or the recursive approach (see Figure 2-2). The noise-predictive decision-feedback (NP-DF) detector can be implemented as in Figure 3-1 after computing the prediction filter **P** and the forward filter $\mathbf{F} = \tilde{\mathbf{R}}^{-1}\mathbf{H}^*$. The conventional DF detector requires a different forward filter **F**, and the interference-cancellation matrix **M** (2-1), while the recursive DF detector needs only its forward filter **F**. Figure 3-4 gives the pseudocode to implement each of these three versions side-by-side. The three BODF detector implementations are identical until the fifth line of pseudocode. The modified-DDF detector [50] uses the conventional DF implementation, while the original BLAST-ordered-DF detector [23] uses the recursive-DF implementation albeit with a different BLAST-ordering algorithm.

Table 3-1 and Table 3-2 give the number of real multiplications required to implement the *preprocessing* complexity of the three BODF detector implementations. The matrix Θ from Line of 4 Figure 3-4 could have anywhere from zero to MN complex entries depending on the permutation calculated by the sorted-QR decomposition. This means that the preprocessing complexity of the BODF detector also depends on the permutation. We assume the worst-case scenario when Θ has no zero entries. Table 3-2 shows that the recursive implementation is the least complex of the three, requiring N^2 fewer multiplications than the noise-predictive implementation.

Table 3-1: Preprocessing complexity of the BLAST sorting algorithm in real multiplications (See Figure 3-4).

	MMSE	ZF
Line 1	$3MN^2 + N^3 + MN$	$3MN^2 + MN$
Line 3	none	$0.5N^3 + N^2 - 1.5N$
Line 4	$3N^3 + 0.5N^2 - 0.5N - 1$	$3N^3 + 0.5N^2 - 0.5N - 1$
Total (M=N)	$7N^3 + 1.5N^2 - 0.5N - 1$	$6.5N^3 + 2.5N^2 - 2N - 1$

MMSE BLAST-ordered DF

```
1. $[\,\mathbf{Q}, \mathbf{L}\,] = \mathrm{QR}_{\mathrm{lower}}(\overline{\mathbf{H}})$
2. $\tilde{\mathbf{Q}}$ = first $M$ rows of $\mathbf{Q}$
3. Compute $\mathbf{L}^{-1}$
4. $[\,\Theta, \mathbf{U}, \Pi\,] = \mathrm{sortedQR}_{\mathrm{upper}}(\,(\mathbf{L}^{-1})^*\,)$
5. $\mathbf{D}^{-1} = \mathrm{diag}(\,\mathbf{U}\,)$
```

Noise-Predictive DF

```
6. $\mathbf{P} = \mathbf{I} - (\mathbf{D}^{-1}\Theta^*)\mathbf{L}\Pi$
7. $\mathbf{F} = \mathbf{L}^{-1}\tilde{\mathbf{Q}}^*$
8. $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$
9. Implement DF
   as in Figure 3-1
```

Conventional DF

```
6. $\mathbf{M} = (\mathbf{D}^{-1}\Theta^*)\mathbf{L}\Pi$
7. $\mathbf{F} = (\mathbf{D}^{-1}\Theta^*)\tilde{\mathbf{Q}}^*$
8. $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$
9. Implement DF
   as in Figure 2-1
```

Recursive DF

```
6. skip to 7
7. $\mathbf{F} = (\mathbf{D}^{-1}\Theta^*)\tilde{\mathbf{Q}}^*$
8. skip to 9
9. Implement DF
   as in Figure 2-2
```

Figure 3-4. Three different implementations of the MMSE BLAST-ordered DF detector.

The noise-predictive implementation of the BODF detector is most beneficial when the receiver is given the forward filter before detection begins. For example, if an existing system is being upgraded and hardware has already been developed to compute the forward filter, or the receiver estimates the forward filter directly. In this case, the BLAST sorting algorithm can avoid the initial QR decomposition all together, but it must invert **U** (3-16) which requires $0.5N^3 + N^2 - 1.5N$ real multiplications. The sorted-QR

Table 3-2: Total complexity of the BLAST sorting algorithm in real multiplications (See Figure 3-4).

	Noise-Predictive DF	Conventional DF	Recursive DF
Lines 1-5	$7N^3 + 1.5N^2 - 0.5N - 1$	$7N^3 + 1.5N^2 - 0.5N - 1$	$7N^3 + 1.5N^2 - 0.5N - 1$
Line 6	$1.5N^3 + 2.5N^2$	$1.5N^3 + 2.5N^2$	none
Line 7	$1.5MN^2 + 0.5MN$	$3MN^2$	$3MN^2 + 2N^2$
Lines 8-9	not counted	not counted	not counted
Total (M=N)	$10N^3 + 4.5N^2 - 0.5N - 1$	$11.5N^3 + 2N^2 - 0.5N - 1$	$10N^3 + 3.5N^2 - 0.5N - 1$

decomposition is also slightly more complex at $3MN^2 + MN + N^2 - N$ real multiplications since its input is no longer lower triangular. In the end, if $M = N$ the receiver that is given the forward filter requires $6.5N^3 + 6N^2 - 2.5N$ real multiplications to compute the BLAST ordering.

Another example when the noise-predictive implementation allows for reduced complexity is in implementing the ZF version of the BLAST-ordering algorithm and $M = N$. In this case, the LU decomposition [24] can compute the inverse of the channel matrix with only $N^3 - N$ real multiplications. In this case, the total number of real multiplications needed to implement the BLAST-ordering algorithm is $7.5N^3 + 6N^2 - 3.5N$.

The original BLAST algorithm proposes an implementation of the zero-forcing BLAST sort that requires $13N^4/8 + 73N^3/12 + 23N^2/8 - 7N/12$ real multiplications when $M = N$ [23]. Appendix A gives the pseudocode for a modified version of this original BLAST algorithm that requires only $9N^4/8 + 53N^3/12 + 15N^2/8 - 5N/12$ real multiplications. We refer to this algorithm as the iterative-QR sorting algorithm since it iteratively implements N QR decompositions. The complexity of this algorithm grows as $\mathcal{O}(N^4)$, whereas the complexity of the algorithms in Figure 3-4 grow as $\mathcal{O}(N^3)$. However, for small values of N the iterative-QR algorithm has low complexity.

Figure 3-5 illustrates the number of real multiplications required to implement the preprocessing of the zero-forcing BODF detector using four approaches: the noise-predictive algorithm given the channel pseudoinverse, the noise-predictive algorithm using the LU decomposition to calculate the channel pseudoinverse, the recursive algorithm, and the iterative-QR algorithm. Recall that each of these approaches are simply different ways to implement the same detector. In other words, each approach achieves exactly the same performance. As a result the distinguishing criteria among these algorithms is the number of computations they require. None of the algorithms addresses estimation directly, so in this comparison the complexity of estimation is neglected. However, it is possible to estimate **C** directly [4][41], and the complexity of the noise-predictive algorithm that assumes this is shown.

Figure 3-5 demonstrates that the least complex implementation of the zero-forcing BODF detector that is given **H**, is the iterative-QR algorithm if $N \leq 3$, and the noise-predictive algorithm if $N > 3$. On the other hand, the noise-predictive algorithm given **C** is 12% less complex than when given **H** for the case $N = 6$.

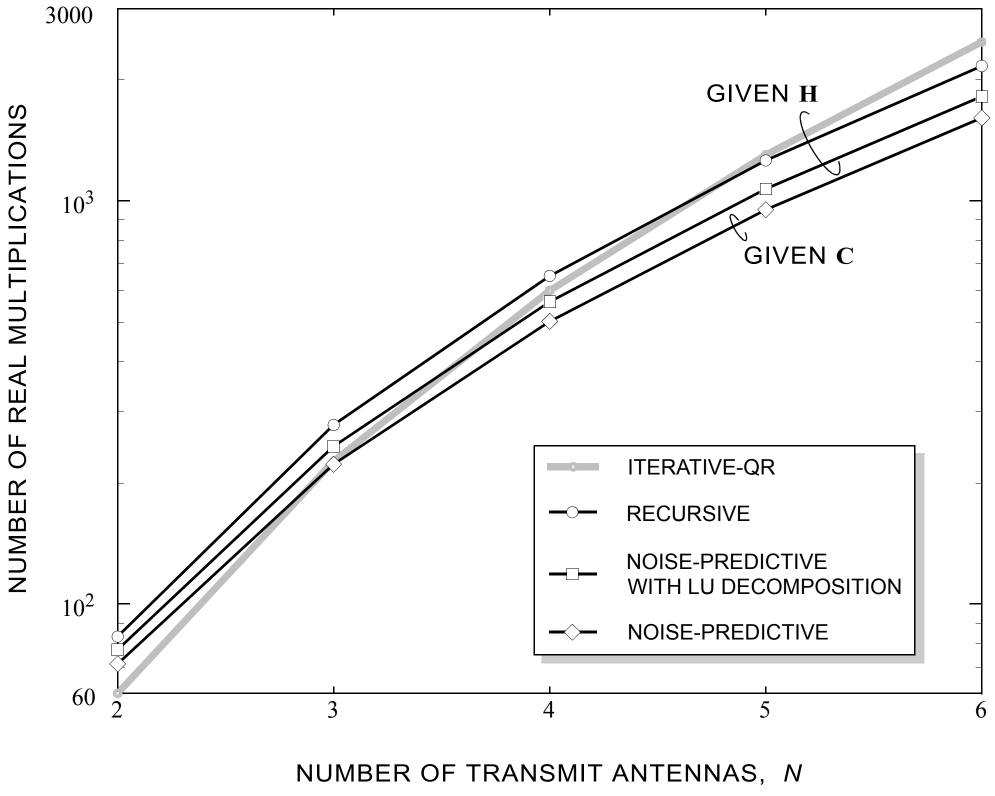

Figure 3-5. Complexity comparison for various BLAST-sorting algorithms for the zero-forcing BODF detector, with $M = N$.

3.6. Chapter Summary

The noise-predictive DF detector consists of a linear detector and a linear prediction mechanism that reduces noise variance. We showed that the noise-predictive view of the DF detector leads to a simple and computationally efficient way of calculating the BLAST detection ordering for both the MMSE and ZF versions of the DF detector. The noise-predictive implementation makes it easy to upgrade an existing linear detector by appending relatively simple additional processing. Furthermore, despite the fact that the linear detector and this add-on processing may have been designed independently, the overall complexity of the resulting noise-predictive BODF detector is lower than previously reported BODF detectors.

CHAPTER 4
IMPROVING THE PERFORMANCE OF THE LINEAR DETECTOR WITH LOW-COMPLEXITY

Many MIMO detectors have been proposed to close the performance and complexity gaps between the linear and ML detectors. For example, the BLAST-ordered decision-feedback (BODF) detector [23] can significantly outperform the linear detector. The BODF detector may be implemented in two stages, where the first stage is the linear detection filter [41][16], and the second stage is a noise-prediction mechanism that reduces noise variance (see Chapter 3). Another low-complexity detector is the group detector [30][40][11], which divides symbols into two groups, and then detects the first group using ML detection. After cancelling the interference due to the first group of symbols, the second group of symbols is detected using a suboptimal technique. Finally, in contrast to the decision-feedback detector, the partial feedback multiuser detector of [17] cancels the interference of only a subset of available decisions. It first divides the users into groups according to their signal energies. Then, the detection strategy for each user group is different, but a given user always uses every decision from stronger users for interference cancellation.

In this paper we propose a new method for improving upon the linear detector called the *partial decision-feedback* (PDF) detector. At very high data rates, where computations are at a premium, even the BODF detector may be too complex to implement. In such applications, the PDF detector offers a way to improve upon the linear detector with much less additional complexity. Like the BODF detector, the first stage of the PDF detector is also a linear detection filter. The second stage is a simplified noise-prediction mechanism that attains most of the performance improvement achieved by the BODF detector, while adding significantly fewer computations. In fact, in the limit of high signal-to-noise ratio (SNR), the word-error rate of the PDF detector converges to that of the BODF detector. The PDF detector can also be viewed as a variation of the group detector where the first and second groups are both detected using linear detection. In this chapter we focus specifically on the case where the first group contains only a single symbol. The PDF detector differs from the partial feedback multiuser detector not only in how it orders the users, but also because it removes the interference from only a subset of the stronger users.

The remainder of this chapter is organized as follows. In Section 4.1 we describe the PDF detector. In Section 4.2 we show that the word-error probability of the zero-forcing PDF detector approaches that of the BODF detector at high SNR. In Section 4.3 we describe the complexity of the PDF detector. In Section 4.4 we compare the performance and complexity of the PDF, BODF, and linear detectors. Finally, in Section 4.5 we make concluding remarks.

4.1. Partial Decision-Feedback Detection

The PDF detector is defined by five steps, which are applied to the memoryless MIMO channel (2-1) as outlined below:

Step 1. Apply a linear filter to the channel output $\boldsymbol{r}$.

Step 2. Identify the index i of the symbol to detect first.

Step 3. Detect the i-th symbol by slicing the i-th output of the linear filter.

Step 4. Cancel the interference due to the i-th symbol.

Step 5. Detect the remaining symbols linearly.

A straightforward implementation of the zero-forcing PDF detector would use the i-th row of the channel pseudoinverse to detect the i-th symbol, and the pseudoinverse of $\mathbf{H}^{(i)}$ to detect the remaining symbols, where $\mathbf{H}^{(i)}$ is the submatrix created by swapping the first and i-th columns of $\mathbf{H}$ then deleting the first column. However, in this section we propose a lower complexity implementation of the PDF detector based on noise prediction. We describe this noise-predictive implementation in a general way that applies to both its zero-forcing (ZF) and minimum-mean-squared error (MMSE) versions.

Step 1: The PDF detector begins by applying the linear filter [41] $\mathbf{C} = (\mathbf{H}^*\mathbf{H} + \hat{\sigma}^2)^{-1}\mathbf{H}^*$ to the channel output, where the parameter $\hat{\sigma} \in \{0, \sigma\}$ determines whether the ZF ($\hat{\sigma} = 0$) or MMSE ($\hat{\sigma} = \sigma$) version of the filter is implemented. The linear filter can be expressed as $\mathbf{C} = \mathbf{L}^{-1}\tilde{\mathbf{Q}}^*$, where the matrices $\mathbf{L}$ and $\tilde{\mathbf{Q}}$ are defined by the following QR decomposition of the extended channel matrix [6][25]:

$$\begin{bmatrix} \mathbf{H} \\ \hat{\sigma}\mathbf{I} \end{bmatrix} = \mathbf{QL}. \tag{4-1}$$

The $(M + N) \times N$ matrix $\mathbf{Q}$, which satisfies $\mathbf{Q}^*\mathbf{Q} = \mathbf{I}$, can be further decomposed according to [25]:

$$\mathbf{Q} = \begin{bmatrix} \tilde{\mathbf{Q}} \\ \hat{\sigma}\mathbf{L}^{-1} \end{bmatrix}. \tag{4-2}$$

so that $\tilde{\mathbf{Q}}$ is defined as the first M rows of $\mathbf{Q}$. The $N \times N$ matrix $\mathbf{L}$ is lower triangular with positive and real diagonal elements.

Applying the linear filter to the channel output yields $\boldsymbol{y} = \mathbf{C}\boldsymbol{r}$, which reduces to:

$$\boldsymbol{y} = \boldsymbol{a} - \hat{\sigma}^2\,\mathbf{U}^*\mathbf{U}\boldsymbol{a} + \mathbf{U}^*\tilde{\mathbf{Q}}^*\boldsymbol{w}, \tag{4-3}$$

where $\mathbf{U} = (\mathbf{L}^{-1})^*$, and where we used the fact that $\tilde{\mathbf{Q}}^*\mathbf{H} = \mathbf{L} - \hat{\sigma}^2\mathbf{U}$. This is the desired signal plus an *effective* noise:

$$\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{n}, \tag{4-4}$$

where the effective noise $\boldsymbol{n} = [n_1, \dots n_N]^T$ is no longer white; its autocorrelation matrix is $\mathbf{R}_{\boldsymbol{nn}} = \sigma^2\mathbf{U}^*\mathbf{U}$, since $\tilde{\mathbf{Q}}^*\tilde{\mathbf{Q}} = \mathbf{I} - \hat{\sigma}^2\mathbf{U}^*\mathbf{U}$. Although $\boldsymbol{n}$ includes a residual interference term in addition to the noise when $\hat{\sigma} \neq 0$, we continue to refer to it simply as *noise*.

Step 2: We propose choosing i as the symbol with the smallest noise variance, which corresponds to the first index of the BLAST ordering. The noise variance of the j-th symbol is proportional to the j-th diagonal element of the autocorrelation matrix $\mathbf{R}_{\boldsymbol{nn}}$, so the index of the first symbol is chosen according to:

$$i = \underset{j \in \{1, 2, \dots N\}}{\operatorname{argmin}} \| \boldsymbol{u}_j \|^2, \tag{4-5}$$

where $\boldsymbol{u}_j$ is the j-th column of $\mathbf{U}$.

Step 3: The i-th symbol is detected by quantizing the i-th output of the linear filter to the nearest symbol in the alphabet, $\hat{a}_i = \text{dec}\{y_i\}$, where $\text{dec}\{x\}$ rounds x to the nearest element of $\mathcal{A}$.

Step 4: Noise-predictive decision feedback is used to reduce the noise variance, and is described by the following equation:

$$\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{p}(y_i - \hat{a}_i), \tag{4-6}$$

where $\boldsymbol{p} = [p_1, \dots p_N]^T$ is a vector of prediction coefficients. The difference $y_i - \hat{a}_i$ reduces to the noise n_i whenever the decision is correct ($\hat{a}_i = a_i$). The term $p_k(y_i - \hat{a}_i)$ is a prediction of n_k that exploits the correlation between n_k and n_i. Since the i-th symbol is detected first, it cannot benefit from noise prediction, meaning $p_i = 0$.

The k-th prediction coefficient is chosen to minimize the mean-squared error (MSE) for the k-th symbol (3-4), which reduces to the following when $\hat{a}_i$ is correct:

$$E[|\, n_k - p_k n_i \,|^2] = E[|\, (\boldsymbol{u}_k^* - p_k \boldsymbol{u}_i^*)(\tilde{\mathbf{Q}}^* \boldsymbol{w} - \alpha^2 \mathbf{U} \boldsymbol{a}) \,|^2]$$

$$= \sigma^2 \| \boldsymbol{u}_k - p_k \boldsymbol{u}_i \|^2, \tag{4-7}$$

where the second equality relies upon the fact that $\hat{\sigma} \in \{0, \sigma\}$, and we have substituted $n_j = \boldsymbol{u}_j^* \tilde{\mathbf{Q}}^* \boldsymbol{w} - \hat{\sigma}^2 \boldsymbol{u}_j^* \mathbf{U} \boldsymbol{a}$ from the definition of $\boldsymbol{n}$. From (4-7) we see that the noise variance is minimized when the term $p_k \boldsymbol{u}_i$ is the projection of $\boldsymbol{u}_k$ onto the subspace spanned by $\boldsymbol{u}_i$, so the k-th prediction coefficient is given by:

$$p_k = \boldsymbol{u}_k^* \boldsymbol{u}_i / \|\boldsymbol{u}_i\|^2, \; k \neq i. \tag{4-8}$$

Step 5: The remaining symbols are detected by quantizing the elements of $\boldsymbol{z}$ from (4-6):

$$\hat{a}_k = \text{dec}\{\, z_k \,\}, \; k \neq i. \tag{4-9}$$

Finally, the PDF detector's hard decision regarding a_k can be summarized succinctly as:

$$\hat{a}_k = \text{dec}\{ y_k - p_k (y_i - \hat{a}_i) \}. \tag{4-10}$$

Figure 4-1 shows the block diagram of the PDF detector after i and $\boldsymbol{p}$ have been calculated. Figure 4-2 describes a computationally efficient implementation of the noise-predictive PDF detector.

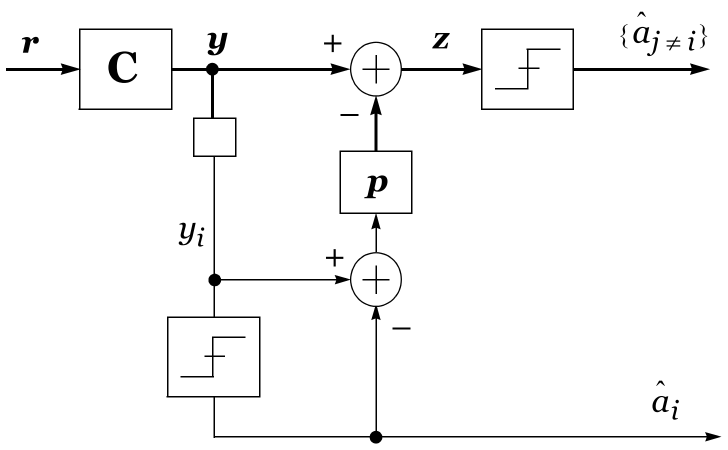

Figure 4-1. Noise-predictive partial DF detector.

```
Function    PartialDF
            Input: $\mathbf{H}, \boldsymbol{r}, \hat{\sigma}$;   Output: $\hat{\boldsymbol{a}}$
1.   $[\mathbf{Q}, \mathbf{L}] = \mathrm{QR}(\begin{bmatrix} \mathbf{H} \\ \hat{\sigma}\mathbf{I} \end{bmatrix})$
2.   $\mathbf{U} = (\mathbf{L}^{-1})^*$
3.   for $j = 1$ to $N$,   $e_j = \|\boldsymbol{u}_j\|^2$,   end
4.   $i = \underset{j \in \{1, 2, \dots N\}}{\arg\min}\, e_j$
5.   $\tilde{\mathbf{Q}}$ = First $M$ rows of $\mathbf{Q}$.
6.   $\boldsymbol{y} = \mathbf{U}^* \tilde{\mathbf{Q}}^* \boldsymbol{r}$
7.   $\hat{a}_i = \mathrm{dec}\{y_i\}$
8.   $n_i = y_i - \hat{a}_i$
9.   for $k \neq i$,
10.      $p_k = \boldsymbol{u}_k^* \boldsymbol{u}_i / e_i$
11.      $\hat{a}_k = \mathrm{dec}\{y_k - p_k n_i\}$
12.  end
```

Figure 4-2. Noise-predictive partial DF detector algorithm.

4.2. Performance Analysis

We now argue that word-error rate (WER) of the zero-forcing PDF detector converges to that of the zero-forcing BODF detector at high SNR. The key is that the error rate of the first symbol detected dominates the WER of both detectors. We begin by considering the probability of error for the first symbol compared to the probability of error for the remaining symbols. Let E_j represent the event of an error on the j-th symbol detected, so that $\mathbf{E} = \cup_{j=1}^{N} E_j$ represents the occurrence of a word error. For the two detectors, the probabilities of word error are given by the following expressions:

$$Pr[\mathbf{E}|\,\mathrm{BODF}] = Pr[E_1|\,\mathrm{BODF}] + Pr[\bar{E}_1\,|\,\mathrm{BODF}]Pr[\mathbf{E}|\,\bar{E}_1\,,\mathrm{BODF}], \tag{4-11}$$

$$Pr[\mathbf{E}|\,\mathrm{PDF}] = Pr[E_1|\,\mathrm{BODF}] + Pr[\bar{E}_1\,|\,\mathrm{BODF}]Pr[\mathbf{E}|\,\bar{E}_1\,,\mathrm{PDF}], \tag{4-12}$$

where $\bar{E}_1$ is the complement of E_1, and we used the fact that $Pr[E_1\,|\,\mathrm{PDF}] = Pr[E_1\,|\,\mathrm{BODF}]$.

In the absence of error propagation, the symbol-error rate of the j-th symbol of the BODF detector has diversity order $M - N + j$ [33], meaning that it decays asymptotically as $\mathrm{SNR}^{-(M-N+j)}$. In (4-11), this means that $Pr[E_1\,|\,\mathrm{BODF}]$ decays as $\mathrm{SNR}^{-(M-N+1)}$, and further that $Pr[\mathbf{E}\,|\,\bar{E}_1\,,\mathrm{BODF}]$ decays as $\mathrm{SNR}^{-(M-N+2)}$, as argued in Theorem 1 of [33]. Similarly, since $Pr[\mathbf{E}\,|\,\bar{E}_1\,,\mathrm{PDF}]$ behaves like the WER of a linear detector applied to an $M \times (N-1)$ channel, it also decays asymptotically as $\mathrm{SNR}^{-(M-N+2)}$. Therefore, the second terms in (4-11) and (4-12) converge to zero faster than the first terms:

$$\lim_{SNR \to \infty} \frac{Pr[\bar{E}_1|\mathrm{BDF}]Pr[\boldsymbol{E}|\bar{E}_1,\,\mathrm{BDF}]}{Pr[E_1|\mathrm{BDF}]} = 0 \quad , \tag{4-13}$$

$$\lim_{SNR \to \infty} \frac{Pr[\bar{E}_1|\mathrm{BDF}]Pr[\boldsymbol{E}|\bar{E}_1,\,\mathrm{PDF}]}{Pr[E_1|\mathrm{BDF}]} = 0 \quad . \tag{4-14}$$

In other words, the error rate of the first symbol dominates at high SNR. It follows that the WER of the zero-forcing PDF detector converges to that of the BODF detector at high SNR:

$$\lim_{SNR \to \infty} \frac{Pr[\boldsymbol{E}|\text{BDF}]}{Pr[\boldsymbol{E}|\text{PDF}]} = 1 \quad . \tag{4-15}$$

4.3. Complexity

We quantify the complexity of the proposed detector by counting the number of real multiplications per symbol period required to implement the algorithm described in Figure 4-2. Complex multiplications are counted as three real multiplications, and the squared absolute values of complex numbers are counted as two real multiplications. Although this complexity measure disregards additions, divisions, and square-roots it is still a reasonable measure of complexity since the number of multiplications dominates the overall complexity. We assume that the channel estimate is updated every L symbol periods. Thus, the total complexity per symbol period is the sum of the preprocessing complexity divided by L plus the core-processing complexity.

For the implementation of the PDF detector proposed in Figure 4-2, the preprocessing complexity includes lines 1-5, and 10. The core-processing complexity includes lines 6-8, and 11. Continuing our view of the PDF detector as an add-on to the linear filter, we consider only those computations it requires beyond the linear detector. Specifically, the linear detector requires lines 1-2 and 6 of Figure 4-2 whether or not the PDF additional processing is used. Therefore, the additional preprocessing complexity required by the

PDF detector are N^2 real multiplications in Line 3, and a maximum of $3N^2/2 - 5N/2 + 1$ real multiplications in Line 10 when $i = N$. The additional core-processing complexity of the PDF detector is only the $3(N-1)$ real multiplications needed at Line 11.

4.4. Numerical Results

In this section, we compare the performance and complexity of the MMSE versions of the linear, partial DF, and BODF detectors. We will show that the performance-complexity trade-off depends on the dimensions of the channel, as well as the size of the input alphabet. Although Section 4.2 predicts identical performance for the PDF and BODF detectors at high SNR, we will see that there can be a significant gap at practical SNR. However, even when the BODF detector significantly outperforms the PDF detector, the PDF detector still offers a way for the receiver to significantly improve upon the performance of the linear detector with a relatively small complexity increase. The SNR is measured as the received signal energy per signaling interval divided by the one-sided noise power spectral density at each receive antenna, divided by the number of bits per symbol, $SNR = E[\|\mathbf{H}\boldsymbol{a}\|^2] / (E[\|\boldsymbol{w}\|^2] \log_2|\mathcal{A}|)$. We assume that the receiver has perfect knowledge of the channel parameters $\mathbf{H}$ and σ^2. We measure performance as the SNR required to reach BER 10^{-3}, and the complexity as the maximum number of real multiplications per symbol period each detector may require.

The PDF detector has the greatest complexity reduction relative to the BODF detector over fast-fading channels. For example, Figure 4-3 demonstrates the performance versus complexity trade-off between the MMSE versions of the linear, PDF, and BODF detectors for $N \times N$ channels with 16- and 64-QAM inputs, assuming that the channel estimate is updated every symbol period ($L = 1$). Figure 4-3 clearly demonstrates that most of the BODF detector's performance improvement over the linear detector is also achieved by the PDF detector, but with only a fraction of the complexity increase. For example, upgrading from the linear to BODF detector when $N = 3$ and the input alphabet is 64-QAM reduces the necessary SNR by 3.9 dB, but requires 79 additional multiplications per symbol period. On the other hand, upgrading from the linear to PDF detector reduces the necessary SNR by 3.6 dB, while requiring only 22 additional multiplications per symbol period. Therefore, in terms of the additional complexity required beyond that of an existing linear detector, the PDF detector is 72% less complex than the BODF detector. In terms of an absolute performance-complexity trade-off, the PDF detector performs 0.3 dB worse than the BODF detector, but is 21% less complex.

We observe in Figure 4-3 that the PDF detector performs better relative to the BODF detector for larger QAM alphabets and smaller numbers of antennas, and decreases complexity more relative to the BODF detector as N increases. The performances gap between the two detectors is smaller for larger alphabets because they require operating at higher SNR, and Section 4.2 demonstrates that the performance of the PDF and BODF detectors will converge at high SNR. For the same reason the performance gap between the zero-forcing PDF and BODF detectors is significantly smaller than for the MMSE versions of these detectors.

4.5. Chapter Summary

The partial decision-feedback detector combines the strategies of the BODF detector and the linear detector. We have shown that when the goal is to upgrade the performance of the linear detector, while keeping complexity low, the PDF detector offers an attractive performance-complexity trade-off. Specifically, by feeding back only one decision, the PDF detector incurs a small performance loss relative to the BODF detector. In addition, the PDF detector is significantly less complex than the BODF detector. For example, for a 3-input 3-output Rayleigh-fading channel with 64-QAM inputs, the PDF detector is 21% less complex than the BODF detector, yet suffers only 0.3 dB of penalty in SNR.

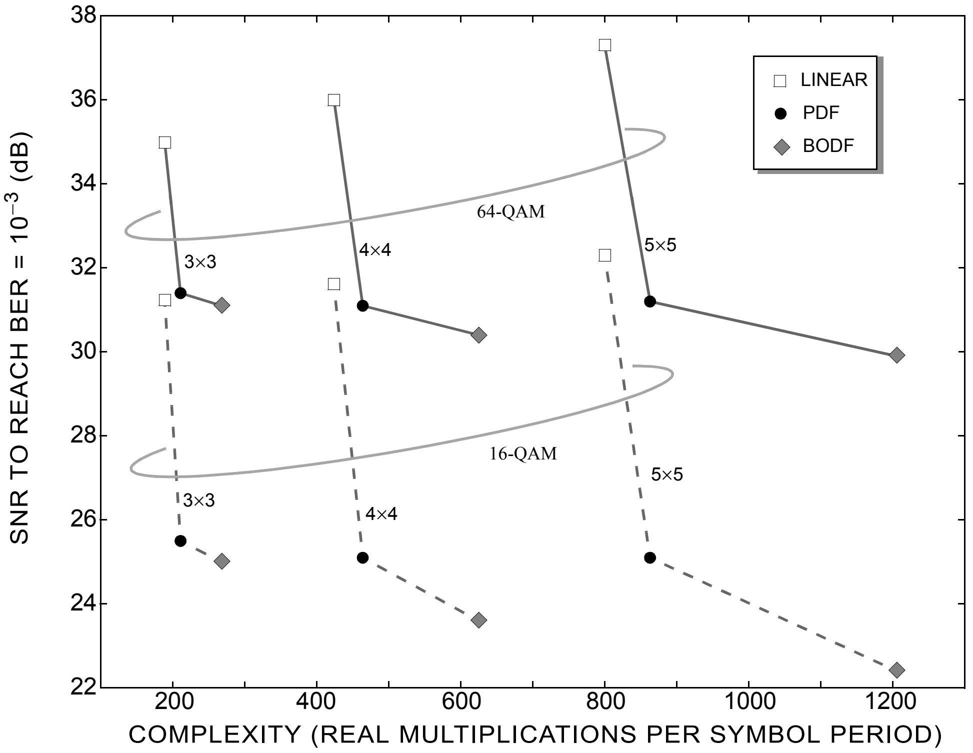

Figure 4-3. Performance versus complexity for the MMSE versions of the linear detector and the noise-predictive PDF, and BODF detectors. Results averaged over 10^5 N-input N-output Rayleigh-fading channels where $L = 1$.

CHAPTER 5

THE CHASE FAMILY OF DETECTION ALGORITHMS

The large gap in both performance and complexity between the maximum-likelihood (ML) and BLAST-ordered decision-feedback (BODF) detectors has motivated the search for alternatives. The sphere detector is a computationally-efficient implementation of the ML detector, and there has been extensive work to reduce its complexity as summarized in Section 2.5. There is an important class of reduced-complexity detectors called *list-based detectors* that adopt a two-step approach of first creating a list of candidate decision vectors, and second choosing the best candidate as its final decision. Examples of list-based detectors include [19][32][45][30]. The error-sensitive detector proposed in [19] starts by looking at the output of a linear detector, and flags some of the transmitted symbols as unreliable. It then enumerates the unreliable symbols while keeping the reliable symbols fixed. The space-time Chase decoder of [32] is similar, except that it identifies bits — not symbols — as being reliable or not. In [45], a technique that generates its list using multiple lattice reductions and multiple zero-forcing (ZF) BODF detectors was shown to closely approximate the ML detector. The parallel detector [30] generates its list by implementing a separate low-complexity detector for each possible value of the first symbol. Numerical results suggest that if the first symbol detected is chosen so as to approximately minimize the probability of error for the remaining symbols, then the parallel detector achieves full receive diversity.

This paper proposes a family of *Chase detectors*, which includes as special cases the ML, BODF [23], ML-BODF [11], parallel [30], and partial decision-feedback (PDF) detectors of Chapter 4. Thus, the Chase family provides a unified framework for comparing a variety of existing detectors. Furthermore, we propose the B-Chase detector as a new special case that performs well on fading channels. We will demonstrate that the B-Chase detector can approach ML performance with less complexity than previously reported detectors [13][11][30][46]. The B-Chase detector distinguishes itself from previous list-based detectors [19][32][45][30] in the unique way it builds its list. We will see that the B-Chase detector achieves better performance with significantly smaller candidate lists, leading to a favorable performance-complexity trade-off.

The remainder of this chapter is organized as follows. In Section 5.1 we introduce the Chase framework for defining detection algorithms, and show how existing detectors fit into the framework. In Section 5.2 we propose a new instance of the Chase detector family called the B-Chase detector. In Section 5.3, we describe a computationally efficient implementation of the B-Chase detector. In Section 5.4 we propose another new instance of the Chase detector family called the S-Chase detector. In Section 5.5 we present some performance and complexity numerical results, and in Section 5.6 we make concluding remarks.

5.1. Chase Detection: A General Framework

In this section we introduce the *Chase detector*, a general detection strategy for MIMO channels modeled by (2-1) that reduces to a variety of previously reported detectors as special cases. The Chase detector defines a simple framework for not only comparing existing MIMO detection algorithms but also proposing new ones. Specifically, a Chase detector is defined by five steps, as illustrated in Figure 5-1, and as outlined below:

Step 1. Identify $i \in \{1, \dots N\}$, the index of the first symbol to be detected.

Step 2. Generate a sorted list $[s_1, \dots s_q]$ of candidate values for the i-th symbol, defined as the q elements of the alphabet nearest to y_i, where $\boldsymbol{y} = (\mathbf{H}^*\mathbf{H} + \hat{\sigma}^2\mathbf{I})^{-1}\mathbf{H}^*\boldsymbol{r}$ is the output of either a ZF ($\hat{\sigma} = 0$) or MMSE ($\hat{\sigma}^2 = N_0$) linear filter.

Step 3. Generate a set of q residual vectors $\{\boldsymbol{r}_1, \dots \boldsymbol{r}_q\}$ by cancelling the contribution to $\boldsymbol{r}$ from the i-th symbol, assuming each candidate from the list is in turn correct:

$$\boldsymbol{r}_j = \boldsymbol{r} - \boldsymbol{h}_i s_j \,. \tag{5-1}$$

Step 4. Apply each of $\{\boldsymbol{r}_1, \dots \boldsymbol{r}_q\}$ to its own independent subdetector, which makes decisions about the remaining $N - 1$ symbols (all but the i-th symbol). Together with s_j, the j-th subdetector defines a *candidate* hard decision $\hat{\boldsymbol{a}}_j$ regarding the input $\boldsymbol{a}$.

Step 5. Choose as the final hard decision $\hat{\boldsymbol{a}}$ the candidate hard decision $\{\hat{\boldsymbol{a}}_1, \dots \hat{\boldsymbol{a}}_q\}$ that best represents the observation $\boldsymbol{r}$ in a minimum mean-squared-error sense:

$$\hat{\boldsymbol{a}} = \underset{\{\hat{\boldsymbol{a}}_1, \ldots \hat{\boldsymbol{a}}_q\}}{\operatorname{argmin}} \| \boldsymbol{r} - \mathbf{H}\hat{\boldsymbol{a}}_j \|^2 . \tag{5-2}$$

The Chase detector is roughly analogous to its namesake, the well-known Chase *algorithm* for soft decoding of binary error-control codes [10], but with the temporal dimension replaced by the spatial dimension. The analogy is loose, but still useful. The Chase algorithm begins by identifying the p least reliable bits of a received codeword, and enumerates all 2^p corresponding binary vectors while fixing the remaining more reliable bits. This is analogous to Steps 1 and 2, except in Step 1, only *one* symbol is identified instead of p, and in Step 2, only a *subset* of the most likely values are enumerated. The Chase algorithm decodes each of the 2^p binary vectors using a simple hard-decoding algorithm, producing a set of candidate hard decisions for the codeword. This is analogous to the cancellation and subdetection in Steps 3 and 4. Finally, the Chase algorithm chooses the candidate codeword that best matches the received observations in a way precisely analogous to that in Step 5.

To uniquely define an instance of the Chase detector requires that the following four parameters be specified:

- A strategy for selecting i in Step 1.
- A list length q for Step 2.

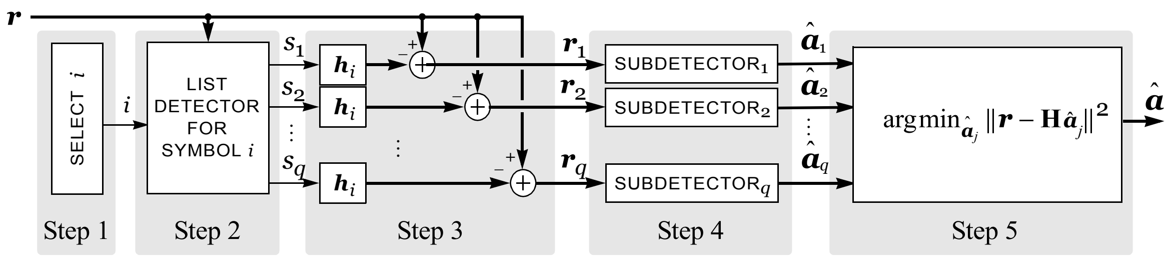

Figure 5-1. Block diagram of the Chase detector.

- A filter type, ZF or MMSE, for Step 2.
- A subdetector algorithm for Step 4.

Table 5-1 summarizes how the ML, BODF, PDF, and parallel detectors may be specified as Chase detectors using these four parameters. For example, the Chase detector reduces to the ML detector when the subdetectors are themselves ML detectors, and the list length is maximized. In this case, the choice of which symbol to detect first has no effect on performance. On the other hand, the Chase detector reduces to the BODF detector when the list length is one and the subdetectors are themselves BODF detectors. In this case, the choice of which symbol to detect first is critical to performance. The parallel detector is another Chase detector whose performance is highly sensitive to the choice of which symbol to detect first. The last row of Table 5-1 describes a new detector that will be proposed in the next section.

Table 5-1: Special cases of the Chase detector.

Detector	First-Symbol Index i	List Length q	Filter type, $\hat{\sigma}$	Subdetector
ML	any	$\|\mathcal{A}\|$	ZF	ML
BODF [22]	♦ BLAST_1	1	ZF or MMSE	BODF
PDF	♦ BLAST_1	1	ZF or MMSE	Linear
Parallel [30]	using (5-9)	$\|\mathcal{A}\|$	ZF	any
B-Chase	using (5-9) or (5-11)	$1 \le q \le \|\mathcal{A}\|$	ZF or MMSE	BODF
S-Chase	using (5-22)	$1 \le q \le \|\mathcal{A}\|$	ZF or MMSE	Sorted-QR DF

♦ The index BLAST_1 signifies the first index of the BLAST ordering [22].

5.2. The B-Chase detector: A New Chase Detector

In this section, we introduce the B-Chase detector, as summarized by the next-to-last row of Table 5-1. The B-Chase detector is defined simply as a Chase detector that uses BODF as a subdetector. The list length q can be any integer in the set $\{1, \dots |\mathcal{A}|\}$, and the filters can be ZF or MMSE. It remains to specify the key parameter, namely, the choice of which symbol to detect first. In the remainder of this section we describe two selection algorithms for selecting i in Step 1. Beforehand, we must derive the signal-to-noise ratio (SNR) for each symbol in the B-Chase detector.

5.2.1. The SNR Gain of a List Detector

A list detector makes an *error* when the actual transmitted symbol does not appear somewhere on the list. With this definition, increasing the length of the list leads to a decrease in the probability of error. (Indeed, a maximal list length of $|\mathcal{A}|$ ensures that the list detector *never* makes an error.) Effectively, increasing the list length leads to an SNR gain.

We demonstrate the effective SNR gain of a list detector using the 4-QAM alphabet $\{e^{\pm j\pi/4}, e^{\pm j3\pi/4}\}$ as an example. Without loss of generality, let $a = e^{j\pi/4}$ be the transmitted symbol. The input to the list detector will then be $y_i = a + n$, where n is noise. When the list length is one, the list detector reduces to the conventional decision device, and makes the correct decision when y_i is in the first quadrant of the complex plane. If we decompose $y_i = |y|e^{j\phi}$, this is equivalent to saying $|\phi - \pi/4| < \pi/4$, which happens with probability $1 - 2Q(\sqrt{SNR}) + Q^2(\sqrt{SNR})$, where $SNR = E[|a|^2]/E[|n|^2]$ is the SNR at the input to the list detector, and where $Q(x) = (2\pi)^{-1/2}\int_x^{\infty}\exp(-t^2/2)\,dt$. Figure 5-2 illustrates the decision regions for the list detector with list lengths q of two and three. A list detector with $q = 2$ will be *correct* when y_i lies in the half-plane $|\phi - \pi/4| < \pi/2$, which happens with probability $1 - Q(\sqrt{2SNR})$. Similarly, a list detector with $q = 3$ will be correct when $|\phi - \pi/4| < 3\pi/4$, which happens with probability $1 - Q^2(\sqrt{SNR})$. The probability of the list detector being *incorrect* can therefore be approximated as $Q(\sqrt{SNR})$, $Q(\sqrt{2SNR})$, and $Q(\sqrt{4SNR})$ for $q = 1$, 2, and 3, respectively.

In general, the probability of error for the list detector can be approximately written as $Q\left(\sqrt{\frac{3\gamma_q^2 SNR}{|\mathcal{A}| - 1}}\right)$, where γ_q^2 is the effective *SNR gain* of the list detector. The above 4-QAM example shows that the SNR gains of the list detector are $\gamma_1^2 = 1$, $\gamma_2^2 = 2$, and $\gamma_3^2 = 4$. Since the list detector cannot be incorrect when $q = 4$, the SNR gain of the full-length list detector is infinite, i.e. $\gamma_4^2 = \infty$. Similar analysis shows that the SNR gains for 16-QAM are $\gamma_2^2 = 2$, $\gamma_3^2 = 4$, $\gamma_8^2 = 8$, and $\gamma_{12}^2 = 13.6$; while for 64-QAM they are $\gamma_4^2 = 4$, $\gamma_8^2 = 8$, $\gamma_{16}^2 = 18.8$, $\gamma_{32}^2 = 38.7$, and $\gamma_{48}^2 = 58.1$.

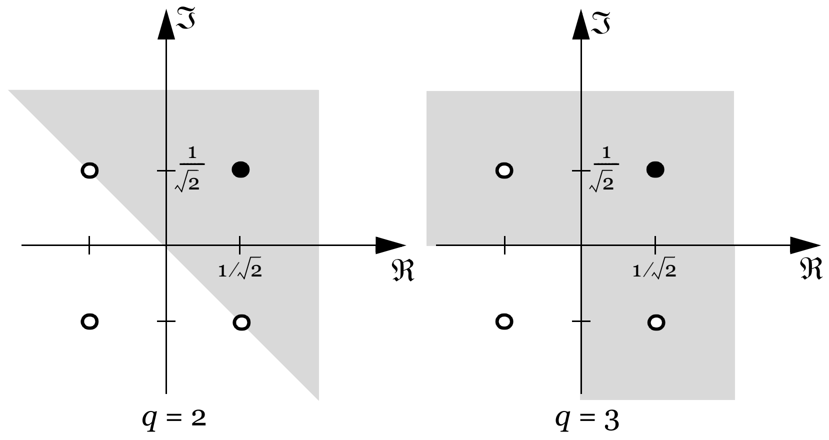

Figure 5-2. Decision regions (shaded) of the list detector for 4-QAM with different list lengths. When $a = e^{j\pi/4}$ is transmitted, the output of the list detector contains a if the output of the linear filter lies within the shaded region.

5.2.2. The SNR of the B-Chase Detector

In this subsection we quantify the SNR of the N symbols for the B-Chase detector. We begin by analyzing the output of the linear filter in Step 2 of the B-Chase detector, which provides the input to the list detector. First, consider the QR decomposition of the extended channel matrix [6][25]:

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \hat{\sigma}\mathbf{I} \end{bmatrix} = \mathbf{QL}, \tag{5-3}$$

where the columns of the $(M+N) \times N$ matrix $\mathbf{Q}$ are orthonormal, and where $\mathbf{L}$ is a lower triangular $N \times N$ matrix with positive and real diagonal elements. The bottom N rows of $\mathbf{Q}$ are the matrix $\hat{\sigma}\mathbf{L}^{-1}$ [6].

In terms of the QR decomposition (2-1), the linear filter of Step 2 can be written as $\mathbf{U}^*\mathbf{Q}^*$, where $\mathbf{U}^* = \mathbf{L}^{-1}$, and where the matrix $\mathbf{Q}$ is defined as the top M rows of $\mathbf{Q}$. The output of this linear filter is thus $\boldsymbol{y} = \mathbf{U}^*\mathbf{Q}^*\boldsymbol{r}$, which reduces to:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{a} - \hat{\sigma}^2\mathbf{U}^*\mathbf{U}\boldsymbol{a} + \mathbf{U}^*\mathbf{Q}^*\boldsymbol{w} \\ &= \boldsymbol{a} + \boldsymbol{n}, \end{aligned} \tag{5-4}$$

where we used the fact that $\mathbf{Q}^*\mathbf{H} = \mathbf{L} - \hat{\sigma}^2\mathbf{U}$. Although $\boldsymbol{n}$ contains both noise and residual ISI when $\hat{\sigma} \neq 0$, we continue to call it *noise*. Since $\hat{\sigma}^2 \in \{0, \sigma^2\}$, the noise variance of the i-th output of the forward filter is $E[|n_i|^2] = \sigma^2\|\boldsymbol{u}_i\|^2$, where $\boldsymbol{u}_i$ is the i-th column of $\mathbf{U}$. This can be seen from (5-4) since $\mathbf{Q}^*\mathbf{Q} + \hat{\sigma}^2\mathbf{U}\mathbf{U}^* = \mathbf{I}$.

The effective SNR for the symbol detected first, including the gain of the list detector, is:

$$SNR_1^{(i)} = \frac{\gamma_q^2}{\sigma^2\|\boldsymbol{u}_i\|^2}. \tag{5-5}$$

A more convenient expression for $SNR_1^{(i)}$, and for the SNRs of the remaining symbols, is defined by a QR decomposition of the extended channel matrix $\overline{\mathbf{H}}$ whose columns are permuted according to the detection order. Let $\Pi^{(i)}$ denote an $N \times N$ permutation matrix that arranges the columns of $\overline{\mathbf{H}}$ such that the i-th column comes first, and the remaining columns are arranged according to the BLAST ordering. Consider the QR decomposition:

$$\overline{\mathbf{H}}\,\Pi^{(i)} = \mathbf{Q}^{(i)}\mathbf{L}^{(i)}, \tag{5-6}$$

where the columns of the $(M+N) \times N$ matrix $\mathbf{Q}^{(i)}$ are orthonormal, and where $\mathbf{L}^{(i)}$ is a lower triangular $N \times N$ matrix with positive and real diagonal elements. The effective SNR for the symbol detected first is:

$$SNR_1^{(i)} = \frac{\gamma_q^2 (l_{1,1}^{(i)})^2}{\sigma^2}. \tag{5-7}$$

The final $N-1$ symbols in the B-Chase detector when the i-th symbol is detected first do not enjoy any list-detection gain. Therefore, their SNR can be expressed as:

$$SNR_k^{(i)} = \frac{(l_{k,k}^{(i)})^2}{\sigma^2}, k = 2, \ldots, N, \tag{5-8}$$

where $l_{k,k}^{(i)}$ is the k-th diagonal of $\mathbf{L}^{(i)}$.

5.2.3. B-Chase Selection

The choice of which symbol to detect first must balance two opposing goals. On the one hand, we want to choose i so that the SNR of the first symbol $SNR_1^{(i)}$ is high, so that the list detector is likely to be correct. Loosely speaking, $SNR_1^{(i)}$ is maximized by choosing the column of $\mathbf{H}$ that is *most orthogonal* to the remaining columns. On the other hand, we also want the subdetectors to see a well-conditioned channel so that the subdetector decisions are likely to be correct. Loosely speaking, this is accomplished by choosing the column of $\mathbf{H}$ that is *least* orthogonal to the remaining columns. We now describe two selection algorithms that strike a balance between these two opposing goals.

Selection Algorithm #1: The first selection algorithm we propose maximizes the minimum SNR of the symbols. The proposed selection algorithm can be succinctly defined using the SNR definitions given in (5-5) and (5-8) as follows:

$$i = \underset{k \in \{1, 2, \dots, N\}}{\arg\max} \; \min\{\gamma_q l_{1,1}^{(k)}, l_{2,2}^{(k)}, \dots, l_{N,N}^{(k)}\} . \tag{5-9}$$

When $q = 1$, so that $\gamma_q = 1$, this selection algorithm can be implemented by choosing the column of $\mathbf{U}$ with minimum norm, as proven in [22]. On the other hand, when the list length is maximized and $\gamma_q^2 = \infty$, the selection algorithm reduces to the parallel selection algorithm [30].

Implementing the selection algorithm (5-9) when $q > 1$ requires $\mathcal{O}(N^4)$ computations. This is because the QR decomposition (5-6) needs to be computed N times, where each decomposition involves computing the BLAST ordering of an $M \times (N-1)$ matrix.

Selection Algorithm #2: In order to avoid the large complexity of Selection Algorithm #1, we propose approximating the SNR of the symbols inside the subdetectors. First of all, if $q = 1$ we select the symbol with minimum noise variance, because this is optimal [22]. On the other hand, if the list length is maximal ($q = |\mathcal{A}|$), we select the symbol with the largest noise variance because the list detector has an infinite SNR gain to counteract the noise. When the list length is greater than one, but not maximal, we propose selecting the symbol which maximizes the minimum of $SNR_1^{(i)}$ and $SNR_2^{(i)}$. This approach is justified by the fact that the smallest SNR inside the subdetector is most often $SNR_2^{(i)}$, therefore $SNR_2^{(i)}$ serves as a rough approximation of the minimum SNR inside the subdetector. This SNR can be easily calculated from the matrix **U**:

$$SNR_2^{(i)} = \frac{1}{\sigma^2 \min_{j \neq i}\{\|\boldsymbol{u}_j\|^2 - |g_{j,i}|^2\}}, \tag{5-10}$$

where $g_{j,i} = \boldsymbol{u}_j^* \boldsymbol{u}_i / \|\boldsymbol{u}_i\|$. Selection Algorithm #2 can be summarized as follows:

$$i = \begin{cases} \underset{k \in \{1, \dots, N\}}{\arg\max} \|\boldsymbol{u}_k\|^2 & , q = |\mathcal{A}| \\ \underset{k \in \{1, \dots, N\}}{\arg\max} \min\left\{\frac{\gamma_q^2}{\|\boldsymbol{u}_k\|^2}, \frac{1}{\min_{j \neq k}\{\|\boldsymbol{u}_j\|^2 - |g_{j,k}|^2\}}\right\} & , \text{else} \end{cases}. \tag{5-11}$$

Note that if $q = 1$, (5-11) reduces to choosing the column of **U** with minimum norm.

5.3. Implementing the B-Chase Detector

Figure 5-3 gives the block diagram of the proposed implementation of the B-Chase detector. Figure 5-4 and Figure 5-5 give the pseudocode for a computationally efficient implementation of the B-Chase detector. In this section we describe the algorithm in detail. In Section 5.5, simulation results will show that the B-Chase detector achieves good performance with low complexity compared to other MIMO detectors.

Step 1: The first step towards implementing the B-Chase detector is to select the symbol to detect first according to (5-9) or (5-11). Selection Algorithm #1 can be implemented directly once the squares of the diagonal elements of $\mathbf{L}^{(i)}$ from (5-6) are known. We will calculate these without computing the QR decomposition of (5-6) directly. Observe that permuting the *columns* of $\overline{\mathbf{H}}$ by $\Pi^{(i)}$ corresponds to permuting the

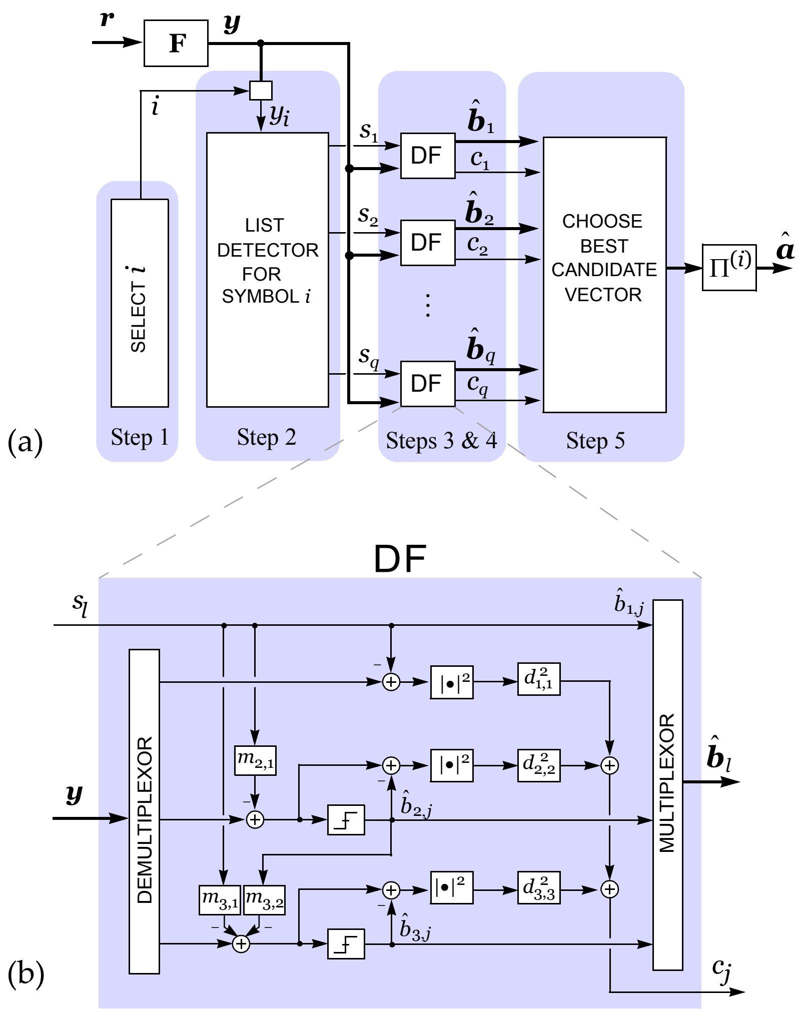

Figure 5-3. (a) Overall block diagram for the B-Chase detector.
(b) Block diagram for the DF subdetector when $N = 3$.

```
Function    BChase Detector.
            Input: $\overline{\mathbf{H}}, q, \mathcal{A}$        Output: $\hat{\boldsymbol{a}}$

1.  $[\mathbf{F}, \mathbf{M}, \Pi^{(i)}, \{d_{j,j}^2\}]$ = BChasePreprocessing($\overline{\mathbf{H}}$, $q$)
2.  $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$
3.  Sort the $q$ symbols in $\mathcal{A}$ nearest $y_1$ in order of
      increasing distance, yielding $\boldsymbol{s} = [\, s_1 \ldots s_q \,]$.
4.  $T = \infty$
5.  for $l = 1$ to $q$,
6.      $b_{1,l} = s_l$
7.      $c_l = |y_1 - s_l|^2 d_{1,1}^2$,
8.      for $k = 2$ to $N$,
9.          if $c_l < T$
10.             $x = y_k - \sum_{j=1}^{k-1} m_{k,j} \hat{b}_{j,l}$,
11.             $\hat{b}_{k,l}$ = dec{ $x$ },
12.             $c_l = c_l + |x - \hat{b}_{k,l}|^2 d_{k,k}^2$,
13.         end
14.     end
15.     if $c_l < T$
16.         $T = c_l$
17.         $f = l$
18.     end
19. end
20. $\hat{\boldsymbol{a}} = \Pi^{(i)} \hat{\boldsymbol{b}}_f$.
```

Figure 5-4. A computationally-efficient implementation of the B-Chase detector.

rows of $\mathbf{C} = \mathbf{U}^* \tilde{\mathbf{Q}}^*$ by $\Pi^{(i)*}$, where $\tilde{\mathbf{Q}}$ is the first M rows of $\mathbf{Q}^{(i)}$. As a result, the definitions of $\Pi^{(i)}$, $\tilde{\mathbf{Q}}$, and $\mathbf{L}^{(i)}$ given in (5-6) are equivalently defined by the following sorted-QR decomposition of $\mathbf{C}^*$:

$$\mathbf{C}^*\Pi^{(i)} = \tilde{\mathbf{Q}}^{(i)}\mathbf{U}^{(i)}, \tag{5-12}$$

where $\mathbf{U}^{(i)} = (\mathbf{L}^{(i)*})^{-1}$. This sorted-QR decomposition can be computed using the algorithm given in Appendix B using the syntax $[\mathbf{Q}^{(i)}, \mathbf{U}^{(i)}, \Pi^{(i)}] = \text{sortedQR}_{\text{UPPER}}(\mathbf{C}^*)$. It is important to note that $\Pi^{(i)}$ calculated in this way puts the final $N-1$ columns of $\overline{\mathbf{H}}$ in their BLAST ordering, as shown in Section 3.2. Finally, the squares of the diagonal elements of $\mathbf{L}^{(i)}$ are a by-product of this sorted-QR decomposition, and Selection Algorithm #1 can be implemented using (5-9).

Lines 3–8 of Figure 5-5 implement Selection Algorithm #1 in a less complex way by computing the sorted-QR decomposition of the lower triangular matrix $\mathbf{U}$, as we now explain. First, substituting the definition of $\mathbf{C}$ into (5-12) gives:

$$\begin{aligned}\mathbf{C}^*\Pi^{(i)} &= \tilde{\mathbf{Q}}\mathbf{U}\Pi^{(i)} \\ &= \tilde{\mathbf{Q}}\Theta^{(i)}\Theta^{(i)*}\mathbf{U}\Pi^{(i)},\end{aligned} \tag{5-13}$$

where $\Theta^{(i)}$ is a unitary matrix such that $\mathbf{U}^{(i)} = \Theta^{(i)*}\mathbf{U}\Pi^{(i)}$ is an upper triangular matrix with real and positive diagonals. Then by inspection we see that $\tilde{\mathbf{Q}}^{(i)} = \tilde{\mathbf{Q}}\Theta^{(i)}$. The matrices $\Theta^{(i)}$, $\mathbf{U}^{(i)}$, and $\Pi^{(i)}$ are simply defined by the sorted-QR decomposition of $\mathbf{U}$:

$$\mathbf{U}\Pi^{(i)} = \Theta^{(i)}\mathbf{U}^{(i)}. \tag{5-14}$$

As before, the squares of the diagonal elements of $\mathbf{L}^{(i)}$ are a by-product of this decomposition.

```
Function    BChasePreprocessing.
            Input: $\overline{\mathbf{H}}$ and $q$        Output: $\mathbf{F}$, $\mathbf{M}$, $\Pi^{(i)}$ and $\{d_{j,j}^2\}$

1.  $[\mathbf{Q}, \mathbf{L}]$ = QRdecomposition($\overline{\mathbf{H}}$)
2.  $\mathbf{U}$ = inverse of $\mathbf{L}^*$
3.  for $j = 1$ to $N$, $e_j = \sum_{k=1\ldots j} |u_{k,j}|^2$ , end
4.  for $k = 1$ to $N$,
5.      $[\Theta^{(k)}, \mathbf{U}^{(k)}, \Pi^{(k)}, \{(l_{j,j}^{(k)})^2\}_{j=1\ldots N}]$ = sortedQR$_{\text{UPPER}}$($\mathbf{U}, \boldsymbol{e}, k$);
6.      $S^{(k)} = \min\left(\gamma_q^2 (l_{1,1}^{(k)})^2, \{(l_{j,j}^{(k)})^2\}_{j\neq 1}\right)$
7.  end
8.  $i = \underset{k \in \{1, 2, \ldots N\}}{\text{argmax}} S^{(k)}$
9.  $\mathbf{D}^{-1}$ = diag( $\mathbf{U}^{(i)}$ )
10. $\tilde{\mathbf{Q}}$ = first $M$ rows of $\mathbf{Q}$
11. $\mathbf{F}$ = ( $\mathbf{D}^{-1}\Theta^{(i)*}$)$\mathbf{Q}^*$
12. $\mathbf{M}$ = ( $\mathbf{D}^{-1}\Theta^{(i)*}$)$\mathbf{L}\Pi^{(i)}$
13. for $j = 1$ to $N$, $d_{j,j}^2 = (l_{j,j}^{(i)})^2$, end
```

Figure 5-5. The preprocessing pseudocode for the proposed implementation of the B-Chase detector that uses Selection Algorithm #1.

Implementing Selection Algorithm #2 is considerably easier than Selection Algorithm #2. Once the $N(N-1)$ squared-magnitudes $\{|g_{j,k}|^2 | \ 1 \leq j \leq N, 1 \leq k \leq N, j \neq k\}$ are computed Selection Algorithm #2 can be implemented directly as given in (5-11). These squared magnitudes are computed as:

$$|g_{j,k}|^2 = |[\mathbf{U}^*\mathbf{U}]_{j,k}|^2 / \|\boldsymbol{u}_k\|^2, \tag{5-15}$$

where $[\bullet]_{j,k}$ is the element at the j-th row and k-th column of a matrix.

Before moving on to Step 2, we propose applying a front-end filter to the channel output that reduces the complexity of subsequent steps. Lines 9–11 of Figure 5-5 give the pseudocode for computing the front-end filter **F**, which is defined as follows:

$$\mathbf{F} = \mathbf{D}^{-1}\tilde{\mathbf{Q}}^{(i)*}, \tag{5-16}$$

where $\mathbf{D}$ is a diagonal matrix with $d_{j,j} = l^{(i)}_{j,j}$. Similar to (2-2), the output of this filter $\boldsymbol{y} = \mathbf{F}\boldsymbol{r}$ reduces to:

$$\boldsymbol{y} = \mathbf{M}\boldsymbol{b} + \boldsymbol{n}, \tag{5-17}$$

where $\mathbf{M} = \mathbf{D}^{-1}\mathbf{L}^{(i)}$ is an $N \times N$ lower-triangular matrix with ones along the diagonal, where $\boldsymbol{b} = \Pi^{(i)*}\boldsymbol{a}$ is a permuted version of the channel input, and the effective noise is $\boldsymbol{n} = \mathbf{F}\boldsymbol{w} - \hat{\sigma}^2\mathbf{D}^{-1}\mathbf{U}^{(i)}\boldsymbol{b}$. Line 12 of Figure 5-5 gives the pseudocode for computing $\mathbf{M}$, which will be needed to implement the subdetectors.

Step 2: After applying the front-end filter as shown in Line 2 of Figure 5-4 to compute $\boldsymbol{y}$ (5-17), the list detector simply generates a list $\boldsymbol{s} = [s_1, \dots s_q]$ of the q symbols in $\mathcal{A}$ nearest y_1.

Steps 3 and 4: It is convenient for implementation to merge steps 3 and 4. The result is q DF detectors whose first symbol decisions are hard-wired to distinct outputs of the list detector. Using the well-known decision-feedback process [16], the intersymbol interference can be cancelled from the k-th element of $\boldsymbol{y}$ as follows:

$$x_k = y_k - \sum_{j=1}^{k-1} m_{k,j}\hat{b}_{j,l}, \tag{5-18}$$

where $\hat{b}_{j,l}$ is the decision already made regarding b_j by the l-th subdetector. The l-th subdetector's estimate of the k-th element of $\boldsymbol{b}$ can be summarized succinctly as:

$$\hat{b}_{k,l} = \begin{cases} s_l & , k = 1 \\ dec\{x_k\} & , k > 1 \end{cases}, \tag{5-19}$$

where dec{z} is the symbol in $\mathcal{A}$ nearest z.

Step 5: In the fifth and final step, the B-Chase detector chooses its final decision as the subdetector's output which has the minimum cost. From (5-2), the cost of the l-th decision vector can be expressed as $c_l = \|\boldsymbol{r} - \mathbf{H}\Pi^{(i)}\hat{\boldsymbol{b}}_l\|^2$, which reduces to:

$$c_l = \|\mathbf{D}(\boldsymbol{y} - \mathbf{M}\hat{\boldsymbol{b}}_l)\|^2, \tag{5-20}$$

where $\hat{\boldsymbol{b}}_l$ is the decision vector produced by the l-th subdetector. For the case when $\hat{\sigma}^2 = \sigma^2$, (5-20) becomes an approximation due to the residual ISI.

A crucial piece of this low-complexity implementation is that the computations made inside the subdetectors can be reused to calculate the cost. Specifically, using (5-18) we can rewrite the cost expression (5-20) as:

$$c_l = \sum_{k=1}^{N} |x_k - \hat{b}_{k,l}|^2 d_{k,k}^2, \tag{5-21}$$

since $m_{k,k} = 1$. Using this expression, a cost *threshold* can be established with the cost of the first subdetector's decision, c_1. The cost calculation of subsequent subdetectors (5-21) as well as their decision feedback (5-18) can be aborted whenever this threshold is exceeded (see Line 9 of Figure 5-4). Furthermore, the threshold can be reduced each time a lower cost is found (see Line 15 of Figure 5-4). In any case, calculating the cost of a subdetector's decision vector requires at most only $\mathcal{O}(N)$ additional computations.

As presented here, the B-Chase algorithm implements the subdetectors in serial fashion. The B-Chase detector also lends itself to a parallel implementation since each of the subdetectors can operate independently as portrayed in Figure 5-3.

5.4. The S-Chase Detector

In this section, we introduce the *S-Chase* detector, as summarized by the last row of Table 5-1. The S-Chase detector implements multiple sorted-QR decision-feedback (DF) detectors [47] in parallel. A key problem for all DF detectors on fading channels is the minimal diversity gain for the first symbol detected; this leads to a large probability of error dominates the overall error rate. The proposed S-Chase detector overcomes this bottleneck by considering multiple possibilities for the first symbol, implementing a separate DF detector for each possibility, and choosing the best of the resulting candidate hard decision vectors.

A key benefit of considering multiple possibilities for the first symbol is that it drastically reduces the importance of optimizing the detection ordering. Indeed, we will see that a easily computed but suboptimal ordering is sufficient to achieve good performance. The preprocessing of the S-Chase detector consists only of a sorted-QR decomposition [47] modified to implement a simple heuristic to select the first symbol to detect as a function of the list length. This simple selection and ordering technique minimizes the preprocessing complexity of the S-Chase detector. In Section 5.5, simulation results will confirm that the S-Chase detector is a good choice when the channel experiences fast fading.

The S-Chase detector can be easily defined using the Chase framework established in Section 5.1. The S-Chase detector is defined simply as a Chase detector that uses sorted-QR DF as a subdetector. The list length q can be any integer in the set $\{1, \dots |\mathcal{A}|\}$, and the filters can be ZF or MMSE. It remains to specify the key parameter, namely, the choice of which symbol to detect first.

In order to minimize preprocessing complexity, the S-Chase detector uses a simple heuristic to make the symbol selection. The optimal solution to the selection problem for a given list length was solved in (5-9). The optimal solution when $q = 1$ is the symbol with the *largest* post-detection SNR, as found using the BLAST ordering. On the other hand, when q is maximal a good selection is the symbol with the *smallest* SNR (5-11). To keep complexity at a minimum, we propose a simple and low-complexity heuristic that does not require evaluation of the post-detection SNR values: *choose the index of the column of* $\mathbf{H}$ *with either the minimum or maximum norm, depending on the list length* q. This selection strategy can be summarized succinctly as follows:

$$i = \arg\max_{j \in \{1, \ldots, N\}} \|\boldsymbol{h}_j\|^m , \tag{5-22}$$

where $m = -1$ when $q > 3|A|/4$, and $m = 1$ when $q \le 3|A|/4$.

Once the first symbol to detect has been selected, we propose that the order of the remaining symbols be defined by the following sorted-QR decomposition [47]:

$$\overline{\mathbf{H}}\,\Pi = \mathbf{QDM} , \tag{5-23}$$

where the $N \times N$ permutation matrix Π represents the symbol ordering, $\mathbf{Q} = [\boldsymbol{q}_1, \ldots \boldsymbol{q}_N]$ is an $M \times N$ matrix with orthonormal columns, $\mathbf{D}$ is a diagonal matrix with real and positive diagonal elements, and $\mathbf{M}$ is a lower triangular matrix with ones along the diagonal. To preserve the selection made by (5-22), the first column of Π is the i-th column of the identity matrix. The final $N-1$ columns of Π are chosen according to the sorted-QR decomposition, which places weaker symbols later in the detection order. Implementing the selection heuristic (5-22) requires no additional floating-point operations because the QR decomposition already requires the column norms of $\overline{\mathbf{H}}$. Figure 5-6 gives the pseudocode for the preprocessing of the S-Chase detector, which computes Π, $\mathbf{Q}$, $\mathbf{D}$, and

$\mathbf{M}$. The core-processing of the S-Chase detector is identical to the B-Chase detector as given in lines 2–20 of Figure 5-4.

```
Function SChasePreprocessing
    Inputs: ($\overline{\mathbf{H}}$, $q$);  Outputs: ($\mathbf{F}$, $\mathbf{M}$, $\Pi$, $\{d_{j,j}^2\}$)
1.  for $j = 1$ to $N$,  $e_j = \sum_{k=1\ldots M} |\bar{h}_{k,j}|^2$,   end
2.  if $q < \frac{3}{4}|\mathcal{A}|$,
3.          $i = \arg\min\{e_j : j = 1 \ldots N\}$
4.  else
5.          $i = \arg\max\{e_j : j = 1 \ldots N\}$
6.  end
7.  [$\mathbf{Q}$, $\mathbf{G}$, $\Pi$, $\mathbf{D}^{-1}$, $\{d_{j,j}^2\}$] = SortedQR$_{\text{LOWER}}$($\overline{\mathbf{H}}$, $\boldsymbol{e}$, $i$);
8.  $\mathbf{M} = \mathbf{D}^{-1}\mathbf{G}$
9.  $\tilde{\mathbf{Q}}$ = first $M$ rows of $\mathbf{Q}$
10. $\mathbf{F} = \mathbf{D}^{-1}\mathbf{Q}^*$
```

Figure 5-6. Preprocessing for the S-Chase Detector. See Appendix B for SortedQR$_{\text{LOWER}}$ function.

5.5. Performance and Complexity Numerical Results

This section examines the performance and complexity of B-Chase and S-Chase detectors on Rayleigh-fading channels, assuming the channel parameters $\mathbf{H}$ and σ^2 are known to the receiver. In this section, we will compare the MMSE B-Chase and S-Chase detectors to the ZF and MMSE sphere detectors [13] whose initial radius corresponds to the mean-squared error of the output of the ZF and MMSE BODF detectors, respectively. In order to measure the performance-complexity trade-off of these sphere detectors, we measure their performance when they are forced to obey a complexity limit (this is the *truncated sphere detector* from Section 2.6.1). We also compare against the lattice-reduced MMSE BODF (LLL-BODF) and lattice-reduced MMSE linear (LLL-linear) detectors [34][46]. The last detector we compare against is the ML-BODF [11] detector which detects the first three symbols using ZF sphere detection, and the final symbol using ZF BODF detection. We will first give numerical results for the performance and complexity of these detectors individually, then jointly. We use B-Chase$^\star(q)$ to denote the B-Chase detector with list length q, selection algorithm (5-9), and $\hat{\sigma}^2 = \sigma^2$. Likewise, we use B-Chase(q) to denote the B-Chase detector with list length q, selection algorithm (5-11), and $\hat{\sigma}^2 = \sigma^2$. The MMSE versions of the parallel and BODF detectors are also included in the comparison, since they are the special cases B-Chase$^\star(|\mathcal{A}|)$ and B-Chase(1), respectively.

The B-Chase detector achieves near-ML performance for a variety of channel dimensions. To demonstrate this we performed simulations over N-input N-output Rayleigh-fading channels with 16-QAM inputs. Figure 5-6 shows the performance versus the number of antennas, where the SNR is measured as $E[\|\mathbf{H}\boldsymbol{a}\|^2] / (E[\|\boldsymbol{w}\|^2] \log_2|\mathcal{A}|)$. We see that B-Chase(16) achieves near-ML performance, with an SNR penalty that ranges from 0.5 dB to 1.0 dB as the number of antennas N increases from 2 to 6. Reducing the list length degrades performance, but B-Chase(4) performs at least as well as the LLL-BODF detector over the range of N from 2 to 6.

We now quantify the complexity of the B-Chase detector. The best complexity metric depends upon many variables that are specific to a particular implementation. We avoid the problem of defining the relative complexity of different floating-point operations by measuring complexity as the total number of real multiplications (RM) per bit. The squared absolute value of a complex number is counted as two RM, and each complex multiply is counted as three RM. Since the number of divisions and square-roots is small compared to the number of multiplications, the main drawback of counting only the multiplications is that it neglects the contribution to the complexity of the addition operations. However, this is a reasonable simplification since multiplications are generally more complex to implement than additions. Another important point is that the multiplication of a floating point number by a constellation point is counted as an *addition* since the constellation points are just scaled integers [7]. This means that implementing interference cancellation (5-18) is multiply-free.

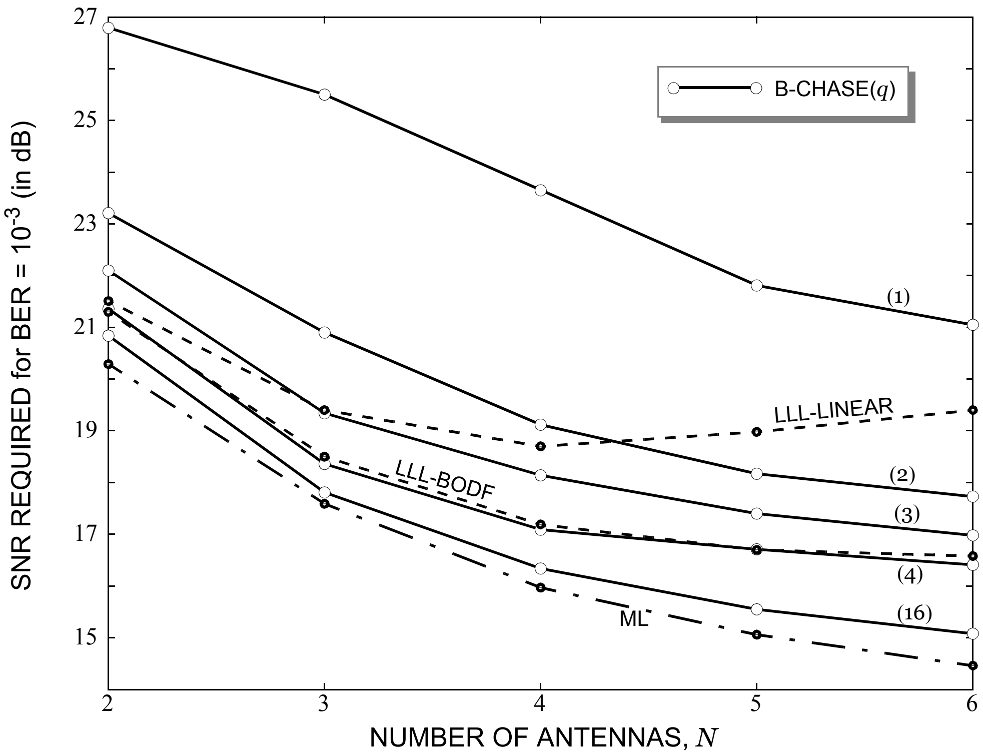

Figure 5-6. SNR required versus number of antennas for various detectors. Results are averaged over 10^5 Rayleigh-fading N-input N-output channels with 16-QAM inputs.

The number of computations required by the detectors we compare is a random variable that depends on the channel and noise. Using the *average* complexity as the basis for comparison may be too optimistic. We measure complexity as the maximum number of real multiplications required by a detector since systems are often designed to handle the worst-case scenario.

The preprocessing used to implement the B-Chase⋆ detector is described in Figure 5-5, where the N sorted-QR decompositions dominate the preprocessing complexity. On the other hand, the most complex part of the preprocessing used to implement the B-Chase detector is the QR decomposition of the extended channel matrix in Line 1 of Figure 5-5. The preprocessing complexities of the MMSE sphere, LLL-BODF, and LR-linear detectors are higher than that of the B-Chase detector. Although the preprocessing for the MMSE sphere detector is essentially the same as that of B-Chase(1), it is more complex because it uses the real channel model which doubles the channel dimensions. The LLL-BODF detector requires the same preprocessing as the MMSE sphere detector in addition to LLL lattice reduction.

Figure 5-4 describes the core-processing of both the B-Chase⋆ and B-Chase detectors. When $q = 1$ it requires only $3MN$ RM since lines 7 and 12 can be skipped. Otherwise it requires a maximum of $3MN + 3qN$ RM. We assume that the channel estimate is updated every L symbol periods. As a result, the *total complexity*, as measured by real multiples per bit, is related to the preprocessing complexity C_{pre} and core-processing complexity C_{core} by:

$$\text{COMPLEXITY} = \frac{C_{core} + C_{pre}/L}{N\log_2|\mathcal{A}|}. \tag{5-24}$$

We now investigate the performance-complexity trade-off of the B-Chase detectors for a 4-input 4-output Rayleigh-fading channel with 16-QAM inputs. Figure 5-7 illustrates the performance versus core complexity trade-off, where performance is measured by the SNR required to reach BER = 10^{-3}, and complexity is measured in real multiplications per bit (RM/bit). This comparison applies to channels that are changing very slowly since comparing only the core-processing complexity is equivalent to assuming the channel never changes ($L = \infty$). The sphere detector achieved BER = 10^{-3} with only 16.0 dB of SNR, but its complexity was as high as 282 RM/bit. B-Chase⋆(16) sacrifices 0.4 dB of performance in order to reduce complexity by 94%, from 282 RM/bit to 18 RM/bit. The truncated sphere detector whose complexity limit was 41 RM/bit also suffered a 0.4 dB performance penalty even though it differed from the sphere detector only 0.1% of the time. The MMSE truncated sphere detector required about twice as much complexity as B-Chase⋆(16) to achieve the same performance. At the low-complexity end of the spectrum, B-Chase⋆(2) outperforms the BODF detector (B-Chase⋆(1)) by 4.4 dB, while increasing the complexity by 25%, from 3 RM/bit to 4 RM/bit. Clearly, the B-Chase detector exhibits a better performance-complexity trade-off than just enforcing a complexity limit on the sphere detector via truncation, or using the ML-DF detector. In addition, by simply adjusting the list length parameter, the B-Chase⋆ detector provides a simple and effective way to trade complexity for performance. The LLL-BODF detector has the same core-processing complexity as the BODF detector at 3 RM/bit, and its performance falls short of the ML detector by only about 1.2 dB. The B-Chase detector outperforms the LLL-BODF detector by up to 0.8 dB when $q = 16$, but it also increases the complexity significantly.

One important dimension to the performance-complexity trade-off not captured in Figure 5-7 is the dependence of the performance on the target BER. In systems with more powerful error correcting codes, a target BER of 10^{-3} could be lower than necessary. Figure 5-8 demonstrates the performance-complexity trade-off when the target BER is 10^{-2}. The complexity of B-Chase⋆ and the LLL-BODF detector remain exactly the same. The performance gap between B-Chase⋆ and optimal performance shrunk slightly after increasing the target BER, and B-Chase⋆(16) still had a better performance-complexity trade-off than the MMSE truncated sphere detector. On the other hand, the performance gap between the ML and LLL-BODF detectors grew, enough that the LLL-BODF detector was outperformed by B-Chase⋆(2), which in turn required 25% more complexity.

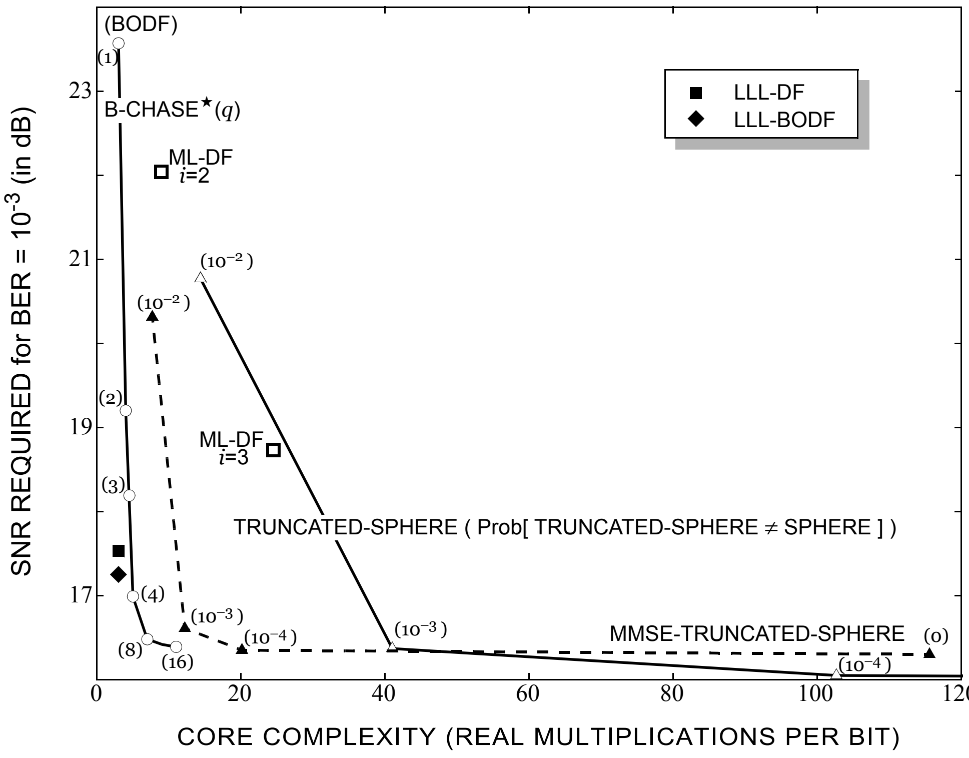

Figure 5-7. Performance-complexity trade-off averaged over 10^5 4-input 4-output Rayleigh-fading channels that are changing slowly ($L = \infty$) with 16-QAM inputs with target BER 10^{-3}.

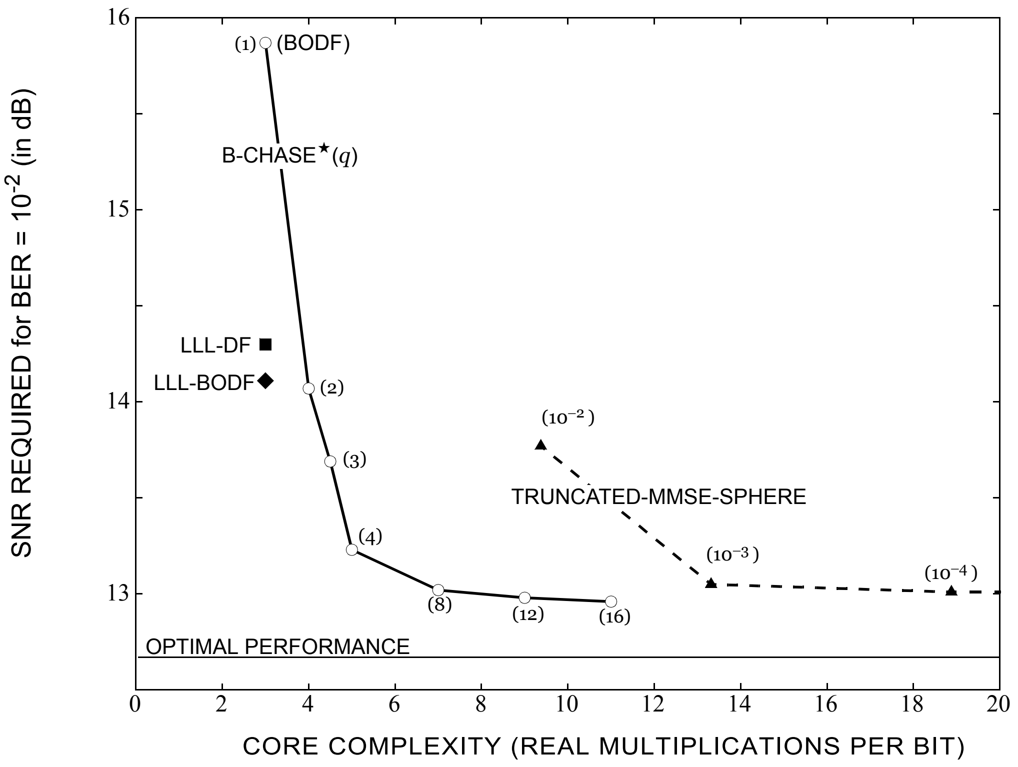

Figure 5-8. Performance-complexity trade-off averaged over 10^5 4-input 4-output Rayleigh-fading channels that are changing slowly ($L = \infty$) with 16-QAM inputs and target BER 10^{-2}.

An important dimension of the complexity comparison is not represented in either Figure 5-7 or Figure 5-8 because they disregard how quickly the channel changes. In contrast to Figure 5-7 and Figure 5-8, Figure 5-9 shows the performance-complexity trade-off when the channel changes quickly ($L = 4$). For fast-changing channels, minimizing the preprocessing complexity is just as important as minimizing the core-processing complexity. In this scenario, B-Chase is a better choice than B-Chase★. In fact, B-Chase is significantly less complex than B-Chase★, and it sacrifices very little performance. In particular, B-Chase(16) and B-Chase★(16) achieve roughly the same performance, but B-Chase(16) is 23% less complex. The S-Chase detector also has a very favorable performance-complexity trade-off. For example, S-Chase(12) outperforms B-Chase(1) by more than 6 dB, but increases complexity by only 4% from 13.2 RM/bit to 13.8 RM/bit. Due to the high worst-case complexity of the LLL algorithm, the LLL-DF and LLL-BODF detectors are not well-suited for fast changing channels as illustrated in Figure 5-9.

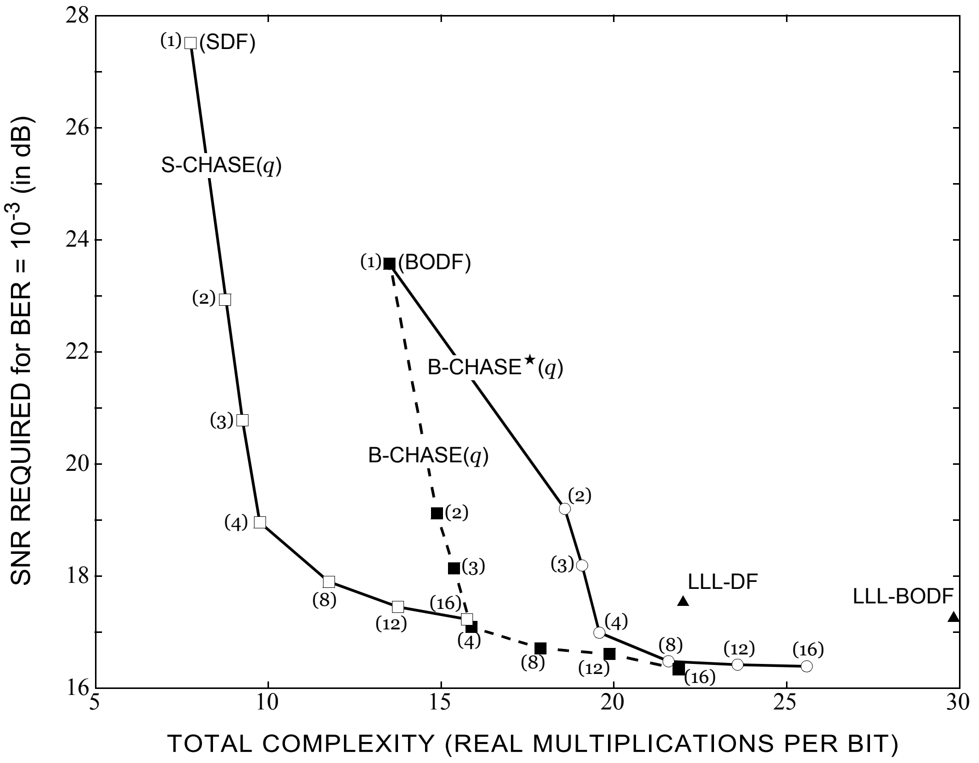

Figure 5-9. Performance-complexity trade-off averaged over 10^5 4-input 4-output Rayleigh-fading channels that are changing quickly ($L = 4$) with 16-QAM inputs and target BER 10^{-3}.

5.6. Chapter Summary

The Chase family of detection algorithms for MIMO channels is a combination of a list detector and a parallel bank of subdetectors. The general Chase detector reduces to a variety of existing MIMO detectors as special cases. Based on the Chase framework, we proposed the B-Chase and S-Chase detectors that can trade performance for reduced complexity by modifying the list length.

Using efficient implementations and a new selection algorithm, the B-Chase detector achieves near-ML performance with low complexity. For example, on a slowly-changing 4-input 4-output Rayleigh-fading channel whose inputs are uncoded 16-QAM, the B-Chase(16) detector fell 0.4 dB short of the ML detector while reducing complexity by 94%. B-Chase(16) achieved the same performance as the MMSE sphere detector with roughly half the complexity. At the low end of the complexity spectrum, the B-Chase(2) detector outperformed the MMSE BODF detector by 4.4 dB while increasing the core complexity by 25%. For a quickly-changing 4-input 4-input Rayleigh-fading channel with uncoded 16-QAM inputs, the S-Chase(12) outperformed the BODF detector by about 6 dB with only a small complexity increase.

CHAPTER 6

Reducing Complexity of the Lattice-Aided DF Detector

The underlying limitation of the performance of the decision-feedback (DF) detector is the high probability of error for the first symbol. This thesis has already described several ways of improving this limitation. Section 2.2 introduced the notion of using the detection order of the symbols to improve the system bottleneck. Chapter 5 introduced the idea of combining a list detector with DF detection to strengthen the most error-prone symbol. The lattice-aided decision-feedback (LA-DF) detector presented in Section 2.4 can also be interpreted as a means of improving the performance of the DF detector.

Using the LLL algorithm to create a more orthogonal effective channel and aid the DF detector has been shown to achieve near-ML performance at high SNR [34][44][46] (See Figure 5-6). Unfortunately, the complexity of the LLL algorithm can be quite large for some channels. To make the LA-DF detector more practical for fast-fading channels we propose a new Double-Sorted (DOS) lattice-reduction algorithm. Combining DOS lattice-reduction with DF creates a new MIMO detector we call the DOS-DF detector. The DOS-DF detector achieves almost the same performance as the LLL-BODF detector, but its worst-case preprocessing complexity is less by roughly half.

6.1. Lattice-Aided Decision-Feedback Detection

As presented in Section 2.4, the LA-DF detector creates an effective channel matrix model:

$$\tilde{\boldsymbol{r}} = \tilde{\mathbf{H}}\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{w}}, \tag{6-1}$$

where the effective channel matrix is $\tilde{\mathbf{H}} = \overline{\mathbf{H}}\mathbf{T}$, and where $\mathbf{T}$ can be any unimodular matrix. The LA-DF detector first makes its decisions regarding each element of $\tilde{\boldsymbol{b}}$ using decision-feedback. Next, the LA-DF detector converts its estimate of $\tilde{\boldsymbol{b}}$, labelled as $\hat{\boldsymbol{b}}$, into a decision vector regarding the channel input $\boldsymbol{a}$.

Section 2.4 described the LA-DF detector using the conventional-DF approach. Here we will describe the LA-DF detector using the recursive-DF approach. The approaches are functionally equivalent, but as pointed out in Section 2.1 the recursive-DF approach is less complex when ordering is involved.

The general QR decomposition used to define the LA-DF detector in (2-21) is written as:

$$\overline{\mathbf{H}}\mathbf{T} = \mathbf{QDM}, \tag{6-2}$$

where $\mathbf{Q}$ is an $M \times N$ matrix with orthonormal columns, where $\mathbf{D}$ is an $N \times N$ diagonal matrix with diagonal elements that are positive and real, and where $\mathbf{M}$ is a lower triangular matrix with ones along the diagonal. In describing the recursive-DF implementation, it is convenient to label the columns of the effective channel, $\tilde{\mathbf{H}} = [\tilde{\boldsymbol{h}}_1, \ldots, \tilde{\boldsymbol{h}}_N]$.

We point out that the matrix $\mathbf{M}$ is not required to implement the LA-DF detector. Instead, the LA-DF detector will use the forward filter $\mathbf{F} = \mathbf{D}^{-1}\mathbf{Q}^*$ one row at a time to make its decisions, and the matrix $\tilde{\mathbf{H}}$ for cancelling interference. The decision regarding the first symbol is made by multiplying the first row of $\mathbf{F}$ times the effective channel output:

$$\hat{b}_1 = \lceil \boldsymbol{f}_1 \tilde{\boldsymbol{r}} \rfloor, \tag{6-3}$$

where $\lceil y \rfloor = \lceil \mathrm{Re}\{y\} \rfloor + \sqrt{-1}\lceil \mathrm{Im}\{y\} \rfloor$ independently rounds each part of y to the nearest integer. Next, the interference due to the first symbol is cancelled from the effective channel output:

$$\tilde{\boldsymbol{r}}_1 = \tilde{\boldsymbol{r}} - \tilde{\boldsymbol{h}}_1 \hat{b}_1 . \tag{6-4}$$

The LA-DF detector assumes that the first decision is correct, and makes the decision regarding the second symbol:

$$\hat{b}_2 = \lceil \boldsymbol{f}_2 \tilde{\boldsymbol{r}}_1 \rfloor . \tag{6-5}$$

The process continues, allowing the LA-DF detector to be defined by the following pair of recursive equations $k = 1, \dots, N$:

$$\hat{b}_k = \lceil \boldsymbol{f}_k \tilde{\boldsymbol{r}}_{k-1} \rfloor \tag{6-6}$$

$$\tilde{\boldsymbol{r}}_k = \tilde{\boldsymbol{r}}_{k-1} - \tilde{\boldsymbol{h}}_k \hat{b}_k , \tag{6-7}$$

where $\tilde{\boldsymbol{r}}_0 = \tilde{\boldsymbol{r}}$, and $\boldsymbol{f}_k$ is the k-th row of $\mathbf{F}$.

Once the decision vector is calculated, it is converted to a decision regarding the channel input:

$$\hat{\boldsymbol{a}} = dec\{(\mathbf{T}\hat{\boldsymbol{b}} + \boldsymbol{s})2c\} , \tag{6-8}$$

where $dec\{\boldsymbol{x}\}$ returns the element of $\mathcal{A}$ nearest each element of $\boldsymbol{x}$, and where $\boldsymbol{s} = 0.5(1 + \sqrt{-1})[1, \dots, 1]^T$.

6.2. Double-Sorted Lattice-Reduction

The role of lattice reduction is to make detection easier by creating an effective channel whose columns are more orthogonal than the original. Choosing the matrix **T** is critical to the performance of the LA-DF detector. If $\mathbf{T} = \mathbf{I}$, then the LA-DF detector performs slightly worse than the DF detector. If **T** is equal to the BLAST permutation matrix, then the LA-DF detector performs slightly worse than the BLAST-ordered decision-feedback (BODF) detector. However, if **T** is calculated using the LLL algorithm, the resulting LA-DF detector is called the LLL-DF detector and it achieves near-ML performance at high SNR [34][44][46].

In this section we present DOS lattice reduction as a new way to compute **T**. It is based on the idea that combining LLL lattice reduction with a BODF detector is overkill. DOS lattice reduction will prove to have a small worst-case complexity while maintaining the good performance of the LLL algorithm. DOS lattice reduction computes **T** in three steps: an initial sorting, a weak-Gramm-Schmidt (WGS) reduction, and a final sorting.

The DOS detector begins by computing the *lower-triangular* sorted-QR decomposition [47] of $\overline{\mathbf{H}}$:

$$\overline{\mathbf{H}}\Pi_1 = \mathbf{Q}_1\mathbf{G}, \tag{6-9}$$

where Π_1 is a permutation matrix that is often identical to the BLAST-ordering permutation of $\overline{\mathbf{H}}$, Appendix B gives the pseudocode for the sorted-QR decomposition. Stopping with this initial sort, $\mathbf{T} = \Pi_1$, the LA-DF detector could not beat the sorted-QR DF detector of [47]. Instead, the DOS detector prepares for the final sort by computing the WGS reduction [34] of **G**, also known as size reduction, which yields:

$$\mathbf{G} = \mathbf{L}_1\mathbf{W}^{-1}, \tag{6-10}$$

where $\mathbf{W}^{-1}$ is a unimodular lower triangular matrix with ones on the diagonal, and $\mathbf{L}_1$ is also a lower triangular matrix. Figure 6-2 gives the pseudocode for the WGS reduction algorithm, which by definition computes $\mathbf{W}^{-1}$ to minimize the column norms of $\mathbf{L}_1$.

Combining (6-9) and (6-10) we see that, in effect, the WGS reduction leads to a new QR decomposition:

$$\bar{\mathbf{H}}\Pi_1\mathbf{W} = \mathbf{Q}_1\mathbf{L}_1. \tag{6-11}$$

If we were to stop here, with $\mathbf{T} = \Pi_1\mathbf{W}$, the resulting LA-DF detector still cannot beat the sorted-QR DF detector of [47] because the diagonals of $\mathbf{L}_1$ and $\mathbf{G}$ are the same. However, the WGS reduction has created a matrix $\bar{\mathbf{H}}\Pi_1\mathbf{W}$ whose columns have not yet been sorted. To achieve the best performance, the DOS-DF detector computes the BLAST ordering of the matrix $\bar{\mathbf{H}}\Pi_1\mathbf{W}$. This last step can also be viewed as a QR decomposition:

$$(\bar{\mathbf{H}}\Pi_1\mathbf{W})\Pi_2 = \mathbf{QL}, \tag{6-12}$$

where Π_2 is the BLAST permutation matrix of $\bar{\mathbf{H}}\Pi_1\mathbf{W}$. This is obviously a special case of the general QR decomposition (6-2) when we adopt the following as the lattice-reduction matrix:

$$\mathbf{T} = \Pi_1\mathbf{W}\Pi_2. \tag{6-13}$$

```
Function   WGS-Reduction.
           Input: $\mathbf{G}$      Output: $\mathbf{W}$, $\mathbf{W}^{-1}$, $\mathbf{L}$
1.  $\mathbf{W} = \mathbf{I}_{N\times N}$ ; $\mathbf{W}^{-1} = \mathbf{I}_{N\times N}$ ; $\mathbf{L} = \mathbf{G}$
2.  for $j = N - 1$ downto 1,
3.    for $i = j + 1$ to $N$
4.        $\mathbf{W}^{-1}_{i,j} = \lceil l_{i,j}/g_{i,i} \rfloor$
5.        for $n = i$ to $N$
6.          $w_{n,j} = w_{n,j} - \mathbf{W}^{-1}_{i,j}\, w_{n,i}$
7.          $l_{n,j} = l_{n,j} - \mathbf{W}^{-1}_{i,j}\, l_{n,i}$
8.        end
9.    end
10. end
```

Figure 6-1. The weak-Gramm-Schmidt reduction algorithm.

Figure 6-2 gives the pseudocode for a computationally efficient implementation for the DOS-DF detector, where the functions sortedQR$_{\text{LOWER}}$ and sortedQR$_{\text{UPPER}}$ are given in Appendix B. Table 6-1 gives the number of multiplications required by this algorithm line-by-line. The BLAST permutation matrix Π_2 is computed using the upper-triangular sorted-QR decomposition of $\mathbf{U}_1 = (\mathbf{L}_1{}^*)^{-1}$:

$$\mathbf{U}_1\Pi_2 = \Theta\mathbf{U}, \tag{6-14}$$

where Θ is an $N \times N$ unitary matrix, and $\mathbf{U}$ is a lower-triangular matrix with real and positive diagonals. After substituting (6-14) into (6-11) and simplifying we obtain:

$$\overline{\mathbf{H}}\mathbf{T} = \mathbf{Q}_1\Theta\mathbf{L}, \tag{6-15}$$

where $\mathbf{L} = \Theta^*\mathbf{L}_1\Pi_2$. Finally, comparing to (6-2) and (6-15), we see that $\mathbf{Q} = \mathbf{Q}_1\Theta$.

```
Function    DOS-DF.
                 Input: $\overline{\mathbf{H}}$, $\boldsymbol{r}$, $\hat{\sigma}$, $c$     Output: $\hat{\boldsymbol{a}}$
1.  $[\mathbf{Q}_1, \mathbf{G}, \Pi_1] = \text{sortedQR}_{\text{LOWER}}(\overline{\mathbf{H}})$
2.  $[\mathbf{W}, \mathbf{W}^{-1}, \mathbf{L}_1] = \text{WGS-Reduction}(\mathbf{G})$
3.  $\mathbf{G}^{-1} = (\text{bottom } N \text{ rows of } \mathbf{Q}_1) / \hat{\sigma}$
4.  $\mathbf{L}_1{}^{-1} = \mathbf{W}^{-1}\mathbf{G}^{-1}$
5.  $[\Theta, \mathbf{U}, \Pi_2] = \text{sortedQR}_{\text{UPPER}}((\mathbf{L}_1{}^{-1})^*)$
6.  $\mathbf{T} = \Pi_1\mathbf{W}\Pi_2,$
7.  $\tilde{\mathbf{H}} = \overline{\mathbf{H}}\mathbf{T}$
8.  $\mathbf{D}^{-1} = \text{diag}(\mathbf{U})$
9.  $\mathbf{F} = \mathbf{D}^{-1}\Theta^*\mathbf{Q}_1{}^*$
10. $\boldsymbol{s} = 0.5(1+\sqrt{-1})[1, \ldots, 1]^T$
11. $\tilde{\boldsymbol{r}}_0 = \begin{bmatrix} \boldsymbol{r}/(2c) \\ 0 \end{bmatrix} - \overline{\mathbf{H}}\boldsymbol{s}$
12. for $k = 1$ to $N$,
13.     $\hat{b}_k = \lceil \boldsymbol{f}_k\tilde{\boldsymbol{r}}_{k-1} \rfloor$
14.     $\tilde{\boldsymbol{r}}_k = \tilde{\boldsymbol{r}}_{k-1} - \tilde{\boldsymbol{h}}_k\hat{b}_k$
15. end
16. $\hat{\boldsymbol{a}} = dec\{\mathbf{T}(\hat{\boldsymbol{b}} + \boldsymbol{s})2c\}$
```

Figure 6-2. The DOS-DF detector algorithm.

6.2.1. Performance Analysis

To understand how the DOS-DF detector outperforms the BODF detector, consider a 2×2 example. The BODF detector performs poorly when either $g_{1,1}^2$ or $g_{2,2}^2$ (from (6-9)) is small since $\min(g_{2,2}^2, g_{1,1}^2)$ dictates performance. The DOS detector improves the bottleneck of the BODF detector. To see this we first observe that $\min(l_{1,1}^2, l_{2,2}^2)$ dictates the performance of the DOS detector since it makes an error only when $\hat{\boldsymbol{b}} \neq \tilde{\boldsymbol{b}}$. The diagonals of the matrix $\mathbf{L}$ in (6-12) can be written as:

$$l_{1,1}^2 = \max\left(g_{1,1}^2, \frac{g_{1,1}^2 g_{2,2}^2}{g_{1,1}^2 + |e|^2 g_{2,2}^2}\right), \tag{6-16}$$

$$l_{2,2}^2 = \min(g_{2,2}^2, g_{1,1}^2 + |e|^2 g_{2,2}^2), \tag{6-17}$$

where $e = g_{2,1}/g_{2,2} - \lfloor g_{2,1}/g_{2,2} \rceil$ is the rounding error, and $|e|^2 \leq 1/2$. Using these expressions we show that $\min(l_{1,1}^2, l_{2,2}^2) \geq \min(g_{1,1}^2, g_{2,2}^2)$ by considering the matrix $\mathbf{L}$ for the cases $\Pi_2 = \mathbf{I}$ and $\Pi_2 \neq \mathbf{I}$.

First, if the final sort does not change the order, then $\Pi_2 = \mathbf{I}$, and $g_{j,j}^2 = l_{j,j}^2$. In this case, the DOS-DF and BODF detectors have the exact same performance limitation and *achieve full diversity*. From (6-16), $g_{1,1}^2 > g_{2,2}^2(1 - |e|^2)$ which implies that $g_{1,1}^2 > g_{2,2}^2/2$, and therefore $Pr(\hat{\boldsymbol{b}} \neq \tilde{\boldsymbol{b}})$ decays as SNR^{-2} [33]. Second, if the final sort changes the order

Table 6-1: Complexity of the DOS-DF preprocessing from Figure 6-2.

	Real Multiplications
Line 1	$3MN^2 + N^3 + MN + N^2 - N$
Line 2	$N^2 - N$
Line 5	$3N^3 + 0.5N^2 - 0.5N - 1$
Line 8	$3MN^2 + 1.5N^3 + 2.5N^2$
Total (M=N)	$11.5N^3 + 5N^2 - 3.5N - 1$

$\Pi_2 \neq \mathbf{I}$, then we know that $g_{1,1}^2 + |e|^2 g_{2,2}^2 < g_{2,2}^2$ which implies that $g_{1,1}^2 < g_{2,2}^2$. In this case, the ratio of the minimum SNR of the DOS-DF and BODF detectors, γ, can have one of two values:

$$\gamma = \begin{cases} \dfrac{g_{2,2}^2}{g_{1,1}^2 + |e|^2 g_{2,2}^2}, & if\ \ l_{1,1}^2 < l_{2,2}^2 \\ \dfrac{g_{1,1}^2 + |e|^2 g_{2,2}^2}{g_{1,1}^2}, & else \end{cases} . \tag{6-18}$$

Obviously, $\gamma \geq 1$, which means that the DOS-DF detector always performs at least as good as the BODF detector. Furthermore, γ can be large, in which case the DOS-DF significantly outperforms the BODF detector.

The reason for the BODF detector's suboptimal performance is the fact that $g_{1,1}^2$ is small too often. Therefore, to quantify how the DOS-DF detector improves the performance bottleneck consider their SNR ratio γ as $g_{1,1}^2$ approaches zero and $|e|^2 > 0$, which can occur when $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are nearly colinear. In this case, $l_{1,1}^2 < l_{2,2}^2$ since $l_{1,1}^2$ also tends to zero, and γ approaches the following limit:

$$\lim_{g_{1,1}^2 \to 0} \gamma = \frac{1}{|e|^2} . \tag{6-19}$$

Therefore, while the BODF detector *always* has poor performance when $g_{1,1}^2$ is small, the DOS-DF detector always has an SNR *at least* 3 dB better because $|e|^2 \leq 1/2$.

6.3. Numerical Results

In this section, we present numerical results that demonstrate that the proposed DOS-DF detector achieves an attractive performance-complexity trade-off over Rayleigh-fading channels. In all simulations we assume that the receiver knows the channel parameters $\mathbf{H}$ and σ. We compare the proposed DOS-DF detector to the MMSE versions of the BODF [23], LLL-DF [34][46] and LLL-BODF [46] detectors. We facilitate the comparison of these detectors by using the same pseudocode implementation as far as possible. To implement the LLL lattice reduction we modified the LLL algorithm given in [46] to handle complex channel matrices as suggested in [34]. Then to implement the LLL-BODF detector we used the pseudocode in Figure 6-2 except that the WGS reduction is replaced by the LLL lattice reduction (Appendix C). To implement the LLL-DF detector we used the pseudocode of Figure 6-2 except the LLL lattice reduction again replaced the WGS reduction, and line 5 was omitted leaving $\Theta = \mathbf{I}$, $\mathbf{U} = (\mathbf{L}_1^{-1})^*$, and $\Pi_2 = \mathbf{I}$.

In order to compare the performance-complexity trade-offs of these detectors, we measure the complexity as the number of multiplications involving two real floating-point numbers. This is a reasonable and convenient metric since these operations generally dominate the overall complexity [7], and it allows us to avoid the problem of defining the relative complexity of different arithmetic operations.

The core-processing for all four detectors considered here is roughly the same (lines 11–16 of Figure 6-2). The main difference is that the preliminary scaling and shifting (2-18) and mapping $\hat{\boldsymbol{b}}$ to the QAM alphabet (6-8) are unnecessary for the BODF detector. Since the complexity of these operations is negligible compared to the rest of the core-processing the complexity comparison reduces to comparing only preprocessing complexities. The DOS-DF detector is more complex than the BODF detector because the forward filter $\mathbf{F}$ has larger dimensions, and the WGS reduction requires $N^2 - N$ real floating-point multiplications. The complexity of the LLL algorithm varies widely depending upon the channel, but since practical systems must be prepared to implement the algorithm in any case we measure its maximum complexity.

Figure 6-3 quantifies the performance-complexity trade-off of different LA-DF detectors as measured over 10^5 4–input 4–output Rayleigh-fading channels with 16-QAM inputs. The performance was measured as the SNR required to reach BER 10^{-3}. The worst-case preprocessing complexity of each detector is shown as measured in real multiplications. The ML curve does not represent its true performance-complexity trade-off since it requires much more *core processing* than the other detectors shown, but it gives a performance reference. The DOS-DF detector is 14% more complex than the BODF detector, but at the same time it improves performance by 6 dB. Due to the LLL algorithm, the LLL-DF detector was up to 35% more complex than the DOS-DF detector while performing slightly *worse*. The LLL-BODF detector was nearly twice as complex as the DOS-DF detector, and performed only 0.3 dB better. Figure 6-3 also shows that using the real channel model to implement the LLL-DF and LLL-BODF detectors does not improve performance, but it does increase complexity substantially.

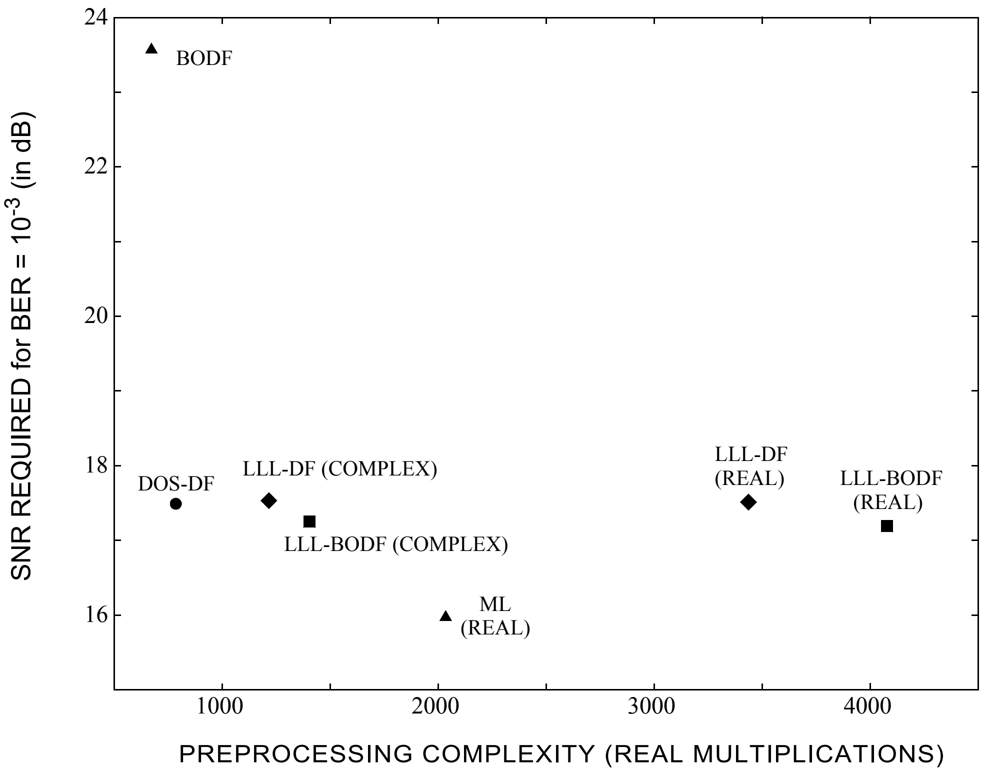

Figure 6-3. Performance versus preprocessing complexity trade-off averaged over 10^5 4-input 4-output Rayleigh-fading channels with 16-QAM inputs and target BER 10^{-3}.

6.4. Chapter Summary

We have proposed a new kind of lattice-aided DF detector called the DOS-DF detector. It is based on a new lattice reduction technique that sandwiches a WGS procedure between two sorting procedures. This new detector was shown to dramatically outperform the BODF detector with only a small increase in complexity. For example, over a 4-input 4-output Rayleigh-fading channel with 16-QAM inputs, the DOS detector outperformed the BODF detector by 6 dB while requiring 14% more preprocessing complexity, and the same core-processing complexity. In the same setting, the LLL-BODF detector needed as much as twice the preprocessing complexity of the DOS-DF detector to perform only 0.3 dB better.

CHAPTER 7

CONCLUSION

Since increasing the bandwidth of a communication system is rarely an option due to physical or legal constraints, future communication systems must use the available spectrum more efficiently in order to increase throughput. In wireless communications spectral efficiency can be increased by using multiple transmit and receive antennas. However, while the capacity of these MIMO channels increases linearly with the number of antennas, the complexity of detection increases exponentially. The practical implication of this is that receivers require vastly more computational power in MIMO systems. Suboptimal detectors can be used to reduce the complexity of the receiver, but they perform worse since they require more transmit power to successfully communicate than the optimal detector. In this thesis, we have proposed MIMO detection strategies and algorithms that can be used to manage the performance-complexity trade-off for MIMO channels.

The BLAST-ordered decision-feedback (BODF) detector has become somewhat of a standard low-complexity detector. We have shown that the complexity of the BODF detector can be further reduced by using noise prediction. The noise-predictive implementation of the BODF detector also makes it easy to upgrade from an existing linear detector to a BODF detector by adding some simple processing.

For high-speed applications where complexity is at a premium, sometimes even the BODF detector is too complex. For this scenario, we propose the partial decision-feedback (PDF) detector. It achieves nearly the same performance as the BODF detector, while requiring nearly the same complexity as the linear detector. For example, for a 3-input 3-output Rayleigh-fading channel with 64-QAM inputs, the PDF detector needs 21% fewer computations than the BODF detector, and performs only 0.3 dB worse.

The family of Chase detectors defines a framework that includes many MIMO detectors as special cases. This framework helps to understand how various MIMO detectors are related to each other, and also provides a means to propose new detectors. We have proposed two new Chase detectors called the B-Chase and S-Chase detectors. The B-Chase detector allows the receiver to implement either the BODF detector or achieve near-ML performance by changing a single parameter. This convenient structure allows the receiver to manage the performance-complexity trade-off depending upon the required performance and available computational resources. For fast-fading channels, the S-Chase detector achieves an even better performance-complexity trade-off than the B-Chase detector. An important strength of the B-Chase and S-Chase detectors is that they require relatively low complexity to achieve near-ML performance. For example, on a slowly-changing 4-input 4-output Rayleigh-fading channel whose inputs are uncoded 16-QAM, one version of the B-Chase detector fell only 0.4 dB short of the ML detector while reducing complexity by 94%.

For slow-fading channels, one way to achieve near-ML performance with low complexity is to use lattice reduction to improve the performance of the decision-feedback (DF) detector. We show how to make lattice-reduced detectors practical for fast-fading channels by combining a new low-complexity lattice-reduction technique called double-sorted (DOS) lattice reduction with the DF detector. The resulting DOS-DF detector achieves near-ML performance and requires near-BODF complexity.

In future research, a MIMO detector with an even better performance-complexity trade-off may be possible by improving the performance of the DOS-DF detector. The DOS-DF detector has already been shown to achieve an attractive performance-complexity trade-off. However, it is still outperformed by the B-Chase detector, and it does not provide any way for the receiver to improve performance by increasing complexity. The B-Chase detector improved upon the BODF detector by making use of a list detector. Similarly, incorporating list detection with the DOS-DF detector should improve performance while increasing complexity at a modest rate.

Future research should also consider the impact of imperfect channel estimation at the receiver. In this thesis we have assumed that the receiver has perfect knowledge of the channel matrix and the noise variance. In practice, errors in estimating these parameters could impact the relative performance of detection algorithms.

Another practical aspect of the overall MIMO communication system not treated in this thesis is the interaction between the MIMO detector and an error-control code. In practice, a powerful error-control code can dramatically improve performance. Many of these error-control codes require the detector to provide bit decisions and the degree of certainty regarding those decisions, this is often referred to as *soft information*. In this thesis we have not addressed this topic directly. However, the Chase detectors compute a list of candidates that can be used to generate soft information [27]. Future research should determine the most practical and efficient way to combine low-complexity MIMO detectors with error-control codes.

MIMO communication systems are gaining momentum in industry as new techniques continue to be proposed to address the practical issues involved. Various companies in industry are building and selling new MIMO systems that leverage the new technology to increase the speed and range of wireless local-area networks. Recent products are already boasting data rates up to 200 Megabits per second by using MIMO technology. Such network speeds will take the wireless office one step closer to reality.

APPENDIX A

BLAST-Ordering Algorithm

This algorithm is a modified version of the original BLAST ordering algorithm [22]. This version is less complex because it operates on the lower triangular matrix **U** instead of the pseudoinverse of **H**. The two algorithms give identical outputs.

```
Function        BLAST
                    Input: $\mathbf{H}, \boldsymbol{r}$        Output: $\hat{\boldsymbol{a}}$

1.    $\mathbf{H}_0 = \mathbf{H},\ \Pi = \mathbf{0}_{N\mathrm{x}N},$
2.    $\boldsymbol{p} = [\ 1, 2, \ldots, N\ ]$
3.    for $j = 1$ to $N$,
4.        $[\mathbf{Q}, \mathbf{R}\ ] = \mathrm{QR}(\mathbf{H}_{j-1})$
5.        $\mathbf{U} = (\mathbf{R}^{-1})^*$
6.        $i = \underset{k\,=\,1 \text{ to } N-j+1}{\arg\min} \|\boldsymbol{u}_k\|^2$
7.        Put a 1 in the $p_i$-th row and $j$-th column of $\Pi$.
8.        Delete $i$-th element of $\boldsymbol{p}$.
9.        $\mathbf{H}_{j-1} = \mathbf{H}_j$ with $i$-th column removed.
10.       $\boldsymbol{w}_j = \boldsymbol{u}_i{}^*\mathbf{Q}^*$
11.   end

12.   $\boldsymbol{r}_0 = \boldsymbol{r},$
13.   $\mathbf{H} = \mathbf{H}\Pi$
14.   for $j = 1$ to $N$,
15.       $y_j = \boldsymbol{w}_j \boldsymbol{r}_{j-1}$
16.       $\hat{a}_j = \mathrm{dec}(y_j)$
17.       $\boldsymbol{r}_j = \boldsymbol{r}_{j-1} - \boldsymbol{h}_j \hat{a}_j$
18.   end
19.   $\hat{a}_k = \Pi \hat{a}_k$
```

Figure A-1. The BLAST-ordered decision-feedback (BODF) detector using a modification of the original sorting algorithm.

APPENDIX B

SORTED-QR DECOMPOSITION

Here we give the lower and upper triangular versions of the sorted-QR decompositions. The inputs ***e*** and *i* are optional. The decomposition has the form:

$$\mathbf{H}\Pi = \mathbf{QG}, \tag{A-1}$$

where **H** is an $M \times N$ matrix, **Q** is an $M \times N$ matrix with orthonormal columns, **G** is lower triangular $N \times N$ matrix with real and positive diagonals, and Π is an $N \times N$ permutation matrix.

The optional input parameter ***e*** is useful when the sorted-QR decomposition is to be computed multiple times for matrices whose column norms are the same. The optional input parameter *i* allows the last column chosen to be specified. The two outputs $\{u_{k,k}\}$ and $\{d_{k,k}^2\}$ are useful by-products of this decomposition. They are related to the diagonals of the output matrix **G**; $u_{k,k} = 1/g_{k,k}$ and $d_{k,k}^2 = g_{k,k}^2$. The matrix **D** whose diagonals are $\{d_{k,k}\}$, is just the matrix created by taking the diagonal elements of **G**. The matrix **U** is therefore equal to $\mathbf{D}^{-1}$.

```
Function    SortedQR_LOWER.
            Input: ($\mathbf{H}$, $\boldsymbol{e}$, $i$);        Outputs: ($\mathbf{Q}$, $\mathbf{G}$, $\Pi$, $\mathbf{U}$, $\{d_{k,k}^2\}$)

1.   $\Pi = \mathbf{I}_{N \times N}$
2.   $\mathbf{Q} = \mathbf{H}$
1.   $\mathbf{G} = \mathbf{0}_{N \times N}$
2.   if $\boldsymbol{e}$ is not input
3.       for $j = 1$ to $N$, $e_j = \sum_{k=1 \ldots M} |q_{k,j}|^2$ , end
4.   end
5.   if $i$ is not input
6.       $t = 0$
7.   else
8.       $t = e_i$
9.       $e_i = 0$
10.  end
11.  for $k = N$ downto 1
12.      if $k > 1$
13.          $i = \arg\min\{ e_j : j = 1, \ldots k , e_j > 0\}$
14.          Swap the $i$-th and $k$-th columns of $\mathbf{Q}$, $\mathbf{G}$, and $\Pi$.
15.          Swap the $i$-th and $k$-th elements of $\boldsymbol{e}$.
16.      else
17.          $e_k = e_k + t$
18.      end
19.      $d_{k,k}^2 = e_k$
20.      $g_{k,k} = \sqrt{e_k}$
21.      $u_{k,k} = 1/g_{k,k}$
22.      $\boldsymbol{q}_k = \boldsymbol{q}_k u_{k,k}$
23.      for $j = k - 1$ downto 1
24.      $g_{k,j} = \boldsymbol{q}_k^* \boldsymbol{q}_j$
25.      $\boldsymbol{q}_j = \boldsymbol{q}_j - g_{k,j} \boldsymbol{q}_k$
26.      $e_j = e_j - |g_{k,j}|^2$
27.    end
28.  end
```

Figure B-1. The lower-triangular sorted-QR decomposition.

Function SortedQR$_{\text{UPPER}}$.
Input: $(\mathbf{H}, \boldsymbol{e}, i)$; Outputs: $(\mathbf{Q}, \mathbf{G}, \Pi, \mathbf{U}, \{d^2_{k,k}\})$

```
1.  $\Pi = \mathbf{I}_{N \times N}$
2.  $\mathbf{Q} = \mathbf{H}$
1.  $\mathbf{G} = \mathbf{0}_{N \times N}$
2.  if $\boldsymbol{e}$ is not input
3.      for $j = 1$ to $N$, $e_j = \sum_{k=1 \ldots M} |q_{k,j}|^2$ , end
4.  end
5.  if $i$ is not input
6.      $t = 0$
7.  else
8.      $t = e_i$
9.      $e_i = 0$
10. end
11. for $k = 1$ to $N$
12.     if $k < N$
13.         $i = \arg\min\{ e_j : j = k, \ldots, N, e_j > 0\}$
14.         Swap the $i$-th and $k$-th columns of $\mathbf{Q}$, $\mathbf{G}$, and $\Pi$.
15.         Swap the $i$-th and $k$-th elements of $\boldsymbol{e}$.
16.     else
17.         $e_k = e_k + t$
18.     end
19.     $d^2_{k,k} = e_k$
20.     $g_{k,k} = \sqrt{e_k}$
21.     $u_{k,k} = 1/g_{k,k}$
22.     $\boldsymbol{q}_k = \boldsymbol{q}_k u_{k,k}$
23.     for $j = k + 1$ to $N$
24.         $g_{k,j} = \boldsymbol{q}_k{}^* \boldsymbol{q}_j$
25.         $\boldsymbol{q}_j = \boldsymbol{q}_j - g_{k,j} \boldsymbol{q}_k$
26.         $e_j = e_j - |g_{k,j}|^2$
27.     end
28. end
```

Figure B-2. The upper-triangular sorted-QR decomposition.

APPENDIX C

LLL LATTICE-REDUCTION ALGORITHM

The LLL lattice-reduction algorithm given in [46] operates on an upper-triangular real matrix **R**. Here we give the lower-triangular version of this algorithm that operates on a complex matrix **R**. Note that the parameter δ used here corresponds to $\sqrt{\delta}$ as used in [46]

Function LLL.
Input: $(\mathbf{Q}, \mathbf{R}, \Pi, \delta)$; Outputs: $(\tilde{\mathbf{Q}}, \tilde{\mathbf{R}}, \mathbf{T})$

```
1.  $\tilde{\mathbf{Q}} = \mathbf{Q}$;     $\tilde{\mathbf{R}} = \mathbf{R}$;   $\mathbf{T} = \Pi$
2.  $N$ = number of columns in $\mathbf{R}$
3.  $i = N - 1$
4.  while $i \geq 1$
5.      for $j = i+1$ to $N$
6.          $\mu = \lceil \tilde{r}_{j,i} / \tilde{r}_{j,j} \rfloor$
7.          for $k = j$ to $N$
8.              $\tilde{r}_{k,i} = \tilde{r}_{k,i} - \mu \tilde{r}_{k,j}$
9.              $t_{k,i} = t_{k,i} - \mu t_{k,j}$
10.         end
11.     end
12.     $\alpha = \tilde{r}_{i+1,i}$
13.     $\beta = \tilde{r}_{i,i}$
14.     $n^2 = |\alpha|^2 + \beta^2$
15.     if $\delta \tilde{r}_{i+1,i+1} > n$
16.         Swap the $(i+1)$-th and $i$-th columns of $\mathbf{T}$ and $\tilde{\mathbf{R}}$.
17.         $\Theta = \mathbf{I}_{N \times N}$
18.         $\alpha = \alpha / n$;   $\beta = \beta / n$
19.         $\Theta_{i,i} = -\alpha$;   $\Theta_{i,i+1} = \beta$;
            $\Theta_{i+1,i} = \beta$;   $\Theta_{i+1,i+1} = \alpha^*$;
20.         $\tilde{\mathbf{R}} = \Theta \tilde{\mathbf{R}}$
21.         $\tilde{\mathbf{Q}} = \tilde{\mathbf{Q}} \Theta^*$
22.         $i = min(i + 1, N - 1)$;
23.     else
24.         $i = i - 1$
25.     end
26. end
```

Figure C-1. The LLL lattice-reduction algorithm for lower-triangular matrices.

REFERENCES

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. on Inf. Th.*, vol. 48, no. 8, Aug. 2002.

[2] C. Belfiore, and J. Park, "Decision-feedback equalization," *Proc. IEEE*, vol. 67, no. 8, pp. 1143-1156, Aug. 1979.

[3] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Trans. on Sig. Proc.*, vol. 51, no. 7, pp. 1722-1730, July 2003.

[4] A. Benjebbour, and S. Yoshida, "Novel semi-adaptive ordered successive receivers for MIMO wireless systems," *IEEE Int. Symp. Personal Indoor Mobile Radio Commun.*, vol. 2, pp. 582-586, Sept. 2002.

[5] A. Bhargave, R. J.P. de Figueiredo, and T. Eltoft, "A detection algorithm for the V-BLAST system," *IEEE Global Telecommun. Conf.*, vol. 1, pp. 494-498, Nov. 2001.

[6] R. Böhnke, D. Wübben, V. Kühn, and K. Kammeyer, "Reduced complexity MMSE detection for BLAST architectures," *IEEE Global Commun. Conf.*, vol. 4, pp. 2258-2262, Dec. 2003.

[7] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fischtner, H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566-1577, July 2005.

[8] R. T. Causey, and J. R. Barry, "Blind multiuser detection using linear prediction," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 8, pp. 1702-1710.

[9] A. Chan, and I. Lee, "A new reduced-complexity sphere decoder for multiple antenna systems," *IEEE Conf. on Commun.*, vol. 1, pp. 460-464, May 2002.

[10] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Info. Th.*, pp. 170-182, Jan. 1972.

[11] W. J. Choi, R. Negi, and J. Cioffi, "Combined ML and DFE decoding for the V-BLAST system," *IEEE Int. Conf. on Commun.*, pp. 1243-1248, June 2000.

[12] T. Cui, and C. Tellambura, "Approximate ML detection for MIMO systems using multistage sphere decoding," *IEEE Sig. Proc. Letters*, vol. 12, no. 3, Mar. 2005.

[13] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. on Info. Th.*, vol. 49, no. 10, Oct. 2003.

[14] M. O. Damen, A. Chkief, and J. C. Belfiore, “Lattice code decoder for space-time codes,” *IEEE Commun. Letters*, pp. 161-163, May 2000.

[15] A. Duel-Hallen, “Equalizers for multiple input/multiple output channels and PAM systems with cyclostationary input sequences,” *IEEE J. Sel. Areas Commun.*, vol. 10, no. 3, pp. 630-639, Apr. 1992.

[16] A. Duel-Hallen, “Decorrelating decision-feedback multiuser detector for synchronous code-division multiple-access channel,” *IEEE Trans. on Commun*., vol. 41, no. 2, pp. 285-290, Feb. 1993.

[17] A. Duel-Hallen, “A family of multiuser decision-feedback detectors for asynchronous code-division multiple-access systems,” *IEEE Trans. on Commun.*, vol. 43, no. 2/3/4, pp. 421-434, Feb./Mar./Apr. 1995.

[18] Y. L. C. de Jong, T. J. Willink, “Iterative tree search detection for MIMO wireless systems,” *IEEE Trans. on Commun.*, vol. 53, no. 6, pp. 930-935, June 2005.

[19] J. H. Y. Fan, R. D. Murch, and W. H. Mow, “Near Maximum Likelihood Detection Schemes for Wireless MIMO Systems,” *IEEE Trans. on Wireless Commun*., vol. 3, no. 5, pp. 1427-1430, Sept. 2004.

[20] R. F. H. Fischer, and C. Windpassinger, “Real versus complex-valued equalization in V-BLAST systems,” *Electronic Letters*, vol. 39, no. 5, pp. 470-471, Mar. 2003.

[21] G. J. Foschini, and M. J. Gans, “On limits of wireless communications in a fading environment when using multiple antennas,” *Wireless Personal Commun.*, vol. 6, pp. 311-335, Mar. 1998.

[22] G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, “Simplified processing for wireless communication at high spectral efficiency,” *IEEE J. Select. Areas Commun.*, vol. 17, no. 11, pp. 1841-1852, Nov. 1999.

[23] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, “Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture,” *Electronic Letters*, vol. 35, no. 1, pp. 14-16, Jan. 1999.

[24] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[25] B. Hassibi, “An efficient square-soot algorithm for BLAST,” *IEEE Conf. on Acoustics, Speech, and Signal Proc.*, vol. 2, pp. 737-740, June 2000.

[26] B. Hassibi, and H. Vikalo, “On the complexity of sphere decoding,” *Asimolar Conf. on Sig., Sys. and Computers*, vol. 2, pp. 1051-1055, Nov. 2001.

[27] B. M. Hochwald, and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. on Commun.*, vol. 51, no. 3, Mar. 2003.

[28] J. W. Kang, and K. B. Lee, "Layered structure ML detection scheme for MIMO systems," *Korean Domestic Conf.*, 2002.

[29] R. Kannan, "Improved algorithms for integer programming and related lattice problems," *Proc. ACM Symp. Theory of Computing*, pp. 193-206, Apr. 1983.

[30] Y. Li, and Z. Luo, "Parallel detection for V-BLAST system," *IEEE Conf. on Commun.*, vol. 1, pp. 340-344, May 2002.

[31] L. Lovasz, *An Algorithmic Theory of Numbers, Graphs, and Convexity*, Philadelphia, PA: SIAM 1986.

[32] D. J. Love, S. Hosur, A. Batra, and R. W. Heath, "Chase decoding for space-time codes," *IEEE Vehicular Tech. Conf.*, vol. 3, pp. 1663-1667, Nov. 2004.

[33] N. Prasad, M. K. Varanasi, "Analysis of decision-feedback detection for MIMO Rayleigh fading channels and optimum allocation of transmitter powers and QAM constellations," *Allerton Conf. Commun., Control, and Comp.*, Univ. of IL., Oct. 2001.

[34] W. H. Mow, "Universal lattice decoding: principle and recent advances," *Wireless Commun. and Mobile Computing, Special issue on coding and its applications in wireless CDMA systems*, vol. 3, no. 5, Aug. 2003, pp. 553-569.

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

[36] C. P. Schnorr, and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181-191, 1994.

[37] D. Seethaler, H. Artés, and F. Hlawatsch, "Dynamic nulling-and-cancelling with near-ML performance for MIMO communication systems," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 777-780, May 2004.

[38] K. Su, and I. J. Wassell, "A new ordering for efficient sphere decoding," *IEEE Int. Conf. on Commun.*, vol. 3, pp. 1906-1910, May 2005.

[39] D. Tse, and P. Viswanath, *Fundamentals of Wireless Communications*, Cambridge Univ. Press, June 2005.

[40] M. K. Varanasi, and B. Aazhang, "Near-optimum detection in synchronous code-division multiple-access systems," *IEEE Trans. on Commun.*, vol. 39, no. 5, pp. 725-736, May 1991.

[41] S. Verdú, *Multiuser Detection*, Cambridge University Press, 1998.

[42] E. Viterbo, and J. Boutros, “A universal lattice code decoder for fading channels,” *IEEE Trans. on Inf. Th.*, vol. 45, no. 5, July 1999.

[43] W. K. Wai, C. Y. Tsui, and R. S. Cheng, “A low complexity architecture of the V-BLAST system,” *IEEE Wireless Commun. and Networking Conf. (WCNC)*, vol. 1, pp. 310-314, Sept. 2000.

[44] C. Windpassinger, and R. F. H. Fischer, “Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction,” *Proc. IEEE Inf. Th. Workshop* (*ITW*), pp. 345-348, April 2003.

[45] C. Windpassinger, L. H.-J. Lampe, and R. F. H. Fischer, “From lattice-reduction-aided detection towards maximum-likelihood detection in MIMO systems,” *Int. Conf. on Wireless and Optical Commun.*, pp. 144-148, July 2003.

[46] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, “Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction,” *IEEE Conf. on Commun.*, vol. 2, pp. 798-802, June. 2004.

[47] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, “Efficient algorithm for decoding layered space-time codes,” *Electronic Letters*, vol. 37, no. 22, pp.1348-1350, Oct., 2001.

[48] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, “MMSE extension of V-BLAST based on sorted QR decomposition,” *IEEE Vehicular Tech. Conf.*, vol. 1, pp. 508-512, Oct. 2003.

[49] H. Yao, and G. W. Wornell, “Lattice-reduction-aided detectors for MIMO communication systems,” *Global Telecommun. Conf.*, vol. 1, pp. 424-428, Nov. 2002.

[50] W. Zha, and S. Blostein, “Modified decorrelating decision-feedback detection of BLAST space-time system,” *IEEE Conf. on Commun.*, vol. 1, pp. 335-339, May 2002.

[51] W. Zhao, and G. B. Giannakis, “Sphere decoding algorithms with improved radius search,” *IEEE Commun. and Networking Conf.*, vol. 4, pp. 2290-2294, March 2004.

[52] H. Zhu, Z. Lei, and F. Chin, “An improved square-root algorithm for BLAST,” *IEEE Trans. on Sig. Proc.*, vol. 11, no. 9, pp. 772-775, Sept. 2004.

[53] E. Zimmerman, W. Rave, and G. Fettweis, “On the complexity of sphere decoding,” *Int. Symp. on Wireless and Pers. Multimedia Commun.*, Sept. 2004.

VITA

Deric Waters was born in Wellington, TX in 1977 and grew up in the small farming community of Samnorwood, Texas. He left home as the valedictorian of the class of 1995 to pursue a college education at Texas Tech University as a presidential scholar. In December 1999, he received bachelor of science degrees in Electrical Engineering and Computer Science with the designation of magna cum laude. He worked as an engineer in the Geolocation and Signal Exploitation division at Southwest Research Institute in San Antonio, Texas before beginning his graduate studies in the Fall of 2000. While earning his master of science degree from the Georgia Institute of Technology, he spent two years at their Georgia Tech Lorraine campus in Metz, France. As part of a dual degree program, he earned a French engineering certificate by passing the senior year at a French engineering school called l'Ecole Supérieure d'Ingenieurs de Marseille. In the Summer of 2002, he was awarded a master of science degree in Electrical and Computer Engineering from the Georgia Institute of Technology with a specialization in Telecommunications. During 2002-2005, he completed the research included in this dissertation under the advisement of Dr. John Barry. Finally, in December 2005 Mr. Waters received his PhD in Electrical and Computer Engineering from the Georgia Institute of Technology.